CARINE CASA GRANDE

ANÁLISE COMPARATIVA ENTRE O DESEMPENHO DE APLICAÇÕES MOBILE DESENVOLVIDAS EM FLUTTER E REACT NATIVE

Londrina
2021

CARINE CASA GRANDE

ANÁLISE COMPARATIVA ENTRE O DESEMPENHO DE APLICAÇÕES MOBILE DESENVOLVIDAS EM FLUTTER E REACT NATIVE

Trabalho de Conclusão de Curso apresentado ao Curso de Ciência da Computação do Centro Universitário Filadélfia - UniFil, como requisito parcial à obtenção do título de Bacharel em Ciência da Computação.

Orientadora: Profª. Simone Sawasaki Tanaka.

Londrina
2021

CARINE CASA GRANDE

ANÁLISE COMPARATIVA ENTRE O DESEMPENHO DE APLICAÇÕES MOBILE DESENVOLVIDAS EM FLUTTER E REACT NATIVE

Trabalho de Conclusão de Curso apresentado ao Curso de Ciência da Computação do Centro Universitário Filadélfia - UniFil, como requisito parcial à obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA

_____
Orientadora: Profª Simone Sawasaki Tanaka
Centro Universitário Filadélfia - UniFil

_____
Prof(a). Dr(a). Componente da Banca
Centro Universitário Filadélfia - UniFil

_____
Prof(a). Dr(a). Componente da Banca
Centro Universitário Filadélfia - UniFil

Cidade, ____de _____de _____.

GRANDE, Carine Casa. **Análise comparativa entre o desempenho de aplicações mobile desenvolvidas em Flutter e React Native**. Trabalho de Conclusão de Curso Ciência da Computação – Centro Universitário Filadélfia - UniFil, Londrina, 2021.

RESUMO

Atualmente, existem diversos sistemas operacionais (SO) para dispositivos móveis, sendo o iOS e o Android os mais utilizados. Entretanto, como os dois possuem linguagens nativas diferentes, sendo o Swift e o Objective-C para o primeiro e Java e Kotlin para o segundo, alguns *frameworks* podem ser utilizados para desenvolver de forma híbrida um aplicativo, isto é, um único código compila para funcionar em SO diferentes. Este trabalho visa analisar e comparar o desempenho dos aplicativos híbridos criados em Flutter e em React Native, dois dos principais *frameworks* que possuem tal objetivo e, para isso, uma aplicação móvel de aparência semelhantes entre si, será desenvolvida e executará diferentes algoritmos de ordenação, ao mesmo em que é contabilizado a porcentagem de bateria e o tempo de execução dos mesmos. Considerando diversas pesquisas já efetuadas comparando ambos *frameworks*, é possível que o Flutter obtenha os melhores resultados tanto no iOS quanto no Android.

Palavras-chave: Flutter. React Native. Desenvolvimento Híbrido. iOS. Android. *Framework*.

GRANDE, Carine Casa. **Comparative analysis between the performance of mobile applications developed in Flutter and React Native**. Trabalho de Conclusão de Curso Ciência da Computação – Centro Universitário Filadélfia - UniFil, Londrina, 2021.

ABSTRACT

Currently there are several operational systems for mobile devices, with iOS and Android being the most used. However, as the two have different native languages, being Swift and Objective-C for the first and Java and Kotlin for the second, some frameworks can be used to hybridly develop an application, that is, a single code compiles to run on different operating systems. This work aims to analyze and compare the performance of hybrid applications created in Flutter and React Native, two of the main frameworks that have this objective and, for that, a mobile application with similar appearance will be developed and will run different sorting algorithms, at the same time the percentage of battery and the running time of the same are counted. Considering several surveys that compared both frameworks, it's possible that Flutter gets the best results on both iOS and Android.

Keywords: Flutter. React Native. Hybrid Development. iOS. Android. Framework.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

CPU	Unidade Central de Processamento
CSS	Cascading Style Sheets
HTML	HyperText Markup Language
IBGE	Instituto Brasileiro de Geografia e Estatística
JS	JavaScript
SO	Sistemas Operacionais
UI	Interfaces de Usuário
W3	World Wide Web
WWW	World Wide Web
XML	Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

O uso dos aparelhos móveis tem crescido cada vez mais, um exemplo disso é o resultado das pesquisas do Instituto Brasileiro de Geografia e Estatística (IBGE) de 2019, que mostraram que 94% das residências brasileiras possuíam telefone celular e que das pessoas com mais de 10 anos 81% possuíam o dispositivo para uso pessoal. Assim sendo, a criação de aplicações móveis é uma área que tem crescido consideravelmente.

Atualmente, existem dois principais sistemas operacionais para dispositivos móveis no mercado: o Android e o iOS, sendo importante desenvolver aplicativos que rodem em ambos para atender a maior parte dos usuários.

O Android foi criado em 2003 pela Android Inc. e comprado pela Google LLC em 2005 e para desenvolver nativamente para este sistema, podem ser utilizadas as linguagens Java ou Kotlin. Já o iOS foi lançado em 2007 pela empresa Apple Inc. e suas linguagens nativas são o Objective-C ou o Swift (linguagem criada pela própria Apple para o desenvolvimento nos seus sistemas operacionais e também no Linux).

De modo a facilitar o desenvolvimento e a manutenção dos aplicativos foram criados *frameworks* que permitem o desenvolvimento híbrido, isto é, um único código que compila para diversos sistemas. React Native e Flutter são dois dos mais conhecidos, o primeiro criado pelo Facebook em 2015 e utilizando a linguagem JavaScript e o segundo lançado em 2017 pelo Google, tendo como base o Dart.

Este trabalho visa analisar e comparar o desempenho dos aplicativos híbridos criados em Flutter e em React Native, de modo a determinar o melhor *framework* para tais aplicações.

2 PROBLEMÁTICA DA PESQUISA E METODOLOGIA

Escolher corretamente qual linguagem ou *framework* será utilizado para desenvolver um sistema é essencial para que o mesmo forneça uma melhor experiência para o usuário. Assim sendo, este trabalho visa escolher entre Flutter e React Native como *frameworks* de desenvolvimento híbrido com base principalmente o desempenho dos mesmos durante a execução de alguns algoritmos de ordenação.

Uma aplicação de interface simples será criada e nela poderá ser selecionado a quantidade de elementos que serão ordenados e o algoritmo utilizado. Durante a execução, será contabilizado o tempo que levará e também o consumo de bateria, sendo posteriormente apresentada ao usuário. Os dados obtidos serão analisados e comparados entre os aplicativos tanto no iOS quanto no Android.

3 RESULTADOS ESPERADOS

Em uma pesquisa de Demedyuk e Tsybulskyi (2020), a comparação entre Flutter, React Native e linguagem nativa apontou que o primeiro obteve resultados superiores ao segundo em relação ao consumo da Unidade Central de Processamento (CPU) e de memória.

Lima (2019) aponta que o React Native é mais indicado para aplicativos mais complexos, e que, assim como publicado por Demedyuk e Tsybulskyi (2020), o Flutter apresentou menores valores médios para consumo de memória e uso do processador.

Embora existam diversas pesquisas comparando os *frameworks*, nenhuma foi encontrada utilizando algoritmos de ordenação. Tais algoritmos são ótimos para analisar o desempenho da linguagem por exigirem capacidade de processamento e memória.

É esperado que durante a execução dos algoritmos o Flutter apresente os melhores resultados tanto no iOS quanto no Android, isso porque nas pesquisas já realizadas o mesmo obteve melhores resultados tanto em CPU quanto em consumo de memória.

3.1 LIMITAÇÕES DO TRABALHO

Será comparado somente Flutter e React Native, tanto no iOS quanto no Android, sendo deixado de lado outros *frameworks* ou linguagens nativas, assim como outros sistemas operacionais. O Flutter foi escolhido por conta do seu potencial de crescimento, mesmo sendo na linguagem Dart, já o segundo foi escolhido por ser o *framework* de JavaScript para desenvolvimento híbrido mais utilizado.

4 REVISÃO DA LITERATURA

Ao desenvolver um software, independente da plataforma em que o mesmo será utilizado ou sua função, o desenvolvedor investe tempo, conhecimento e, consequentemente, dinheiro.

De acordo com o StatCounter (2021), juntos, os sistemas iOS e Android foram utilizados por mais de 98% dos usuários de dispositivos móveis no ano de 2020 e, considerando tal número, entende-se a necessidade de pensar os aplicativos para ambos os sistemas.

Entretanto, para desenvolver aplicativos de forma nativa, são utilizadas linguagens de programação diferentes para cada sistema: o iOS utiliza o Objective-C ou o Swift enquanto o Android utiliza o Java ou o Kotlin. Assim, o custo para criar aplicativos que sejam suportados em sistemas diferentes são caros, além de que quando é necessário realizar correções ou atualizações, mais de um código fonte deve ser revisado.

Visando solucionar o problema, foram criados *frameworks* que permitem o desenvolvimento híbrido de aplicativos, isto é, um único código fonte que é compilado para diversos sistemas.

O JavaScript (JS) é uma das linguagens que possuem diversos *frameworks* com tal objetivo. Conforme definição do MDN Web Docs (2021), o JS é uma linguagem de programação de script orientada a objetos que faz parte do World Wide Web (também conhecido como WWW ou W3), que é um sistema que permite o acesso às informações apresentadas em formato digital pela internet (MARTINS, 2008).

Dentre os *frameworks* para o desenvolvimento de aplicativos híbridos que utilizam o JS como base, o Ionic, o NativeScript e o React Native são os mais conhecidos.

O Ionic foi lançado em 2013 e utiliza o HTML (HyperText Markup Language), o CSS (Cascading Style Sheets) e o JS (ANDRADE, 2020) e também pode utilizar o TypeScript, que é superconjunto de JS criado pela Microsoft que possui tipagem estática e orientação à objetos (MELO, 2021). Além disso, o Ionic é baseado no Apache Cordova, plataforma para desenvolvimento de aplicativos móveis que opera em cima de WebView (DEVMEDIA, 2017) que, por sua vez, é um componente do

sistema operacional que permite que os conteúdos da web sejam exibidos dentro de um aplicativo (SALUTES, 2021).

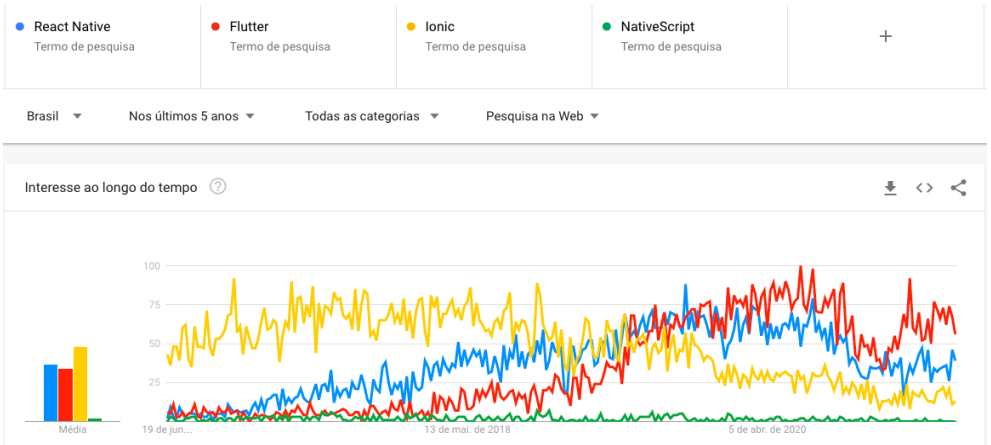
Assim como o Ionic, o NativeScript utiliza o JS ou o TypeScript e o CSS, entretanto, como linguagem de marcação, é utilizado o XML (Extensible Markup Language). Ele não é baseado em uma WebView e sim aproveita o tempo de execução com acesso nativo à API.

Já o React Native foi criado em 2015 pelo Facebook e é uma extensão do React, uma biblioteca para criação de interfaces de usuário (UI) que também foi criada pelo Facebook, em 2011 (ROVEDA, 2016). Este é o *framework* JS mais utilizado para o desenvolvimento híbrido de aplicativos, sendo utilizado por empresas conhecidas, como Facebook, Skype, Instagram e UberEats (CANGUÇU, 2020).

Há também os *frameworks* que não utilizam o JavaScript e um exemplo deste é o Flutter. Desenvolvido pelo Google e lançado em 2017, utiliza Dart uma linguagem de programação fortemente tipada criada também pelo Google e lançada em 2011 (GUEDES, 2020). Mesmo que seja um framework novo, o Flutter tem crescido consideravelmente, sendo utilizado em grandes empresas como o seu criador e também empresas como Alibaba, Nubank (GUEDES, 2020) e iFood (MATIAS, 2020).

Conforme mostrado na Figura 1, a busca pelos termos React Native e Flutter no Google nos últimos 5 anos tem crescido consideravelmente. Em contrapartida, o Ionic teve uma queda nas buscas, e isto se deve ao sucesso e crescimento dos seus concorrentes.

Figura 1 — Busca pelos termos React Native, Flutter e Ionic nos últimos 5 anos.



Fonte: Google Trends (2021)

React Native e Flutter possuem diversas diferenças, sendo que uma delas é em questão da interface do usuário: o primeiro é baseado em componentes nativos enquanto o segundo utiliza widgets proprietários (CANGUÇU, 2019).

Nos aplicativos desenvolvidos em React Native, a aparência deste botão será diferente nos sistemas iOS e Android, pois ele considerará as configurações nativas do sistema, enquanto nos aplicativos desenvolvidos em Flutter o mesmo terá a aparência igual independente do sistema.

Na Figura 2, por exemplo, é possível visualizar a aparência de um botão do aplicativo do Skype, que é desenvolvido em React Native, no sistema operacional Android e na Figura 3, a aparência é a dos botões do mesmo aplicativo no sistema operacional iOS. No primeiro, as bordas são bem quadradas, enquanto o segundo são arredondados, além de utilizarem a fonte do sistema.

Figura 2 — Botão do aplicativo Skype no sistema operacional Android.

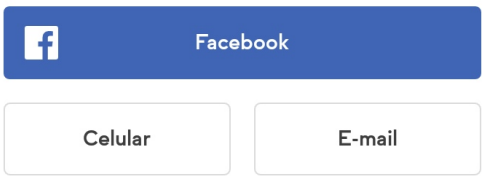
Fonte: O Autor (2021)

Figura 3 — Botão do aplicativo Skype no sistema operacional iOS.

Fonte: O Autor (2021)

Já na Figura 4, vemos um botão utilizado no aplicativo iFood, que é desenvolvido em Flutter, sendo igual ao botão da Figura 5, que é do mesmo aplicativo mas no sistema operacional iOS. A única diferença entre eles é o tamanho do mesmo, mas isso se dá por conta da resolução das telas dos dispositivos.

Figura 4 — Botão do aplicativo iFood no sistema operacional Android.



Fonte: O Autor (2021)

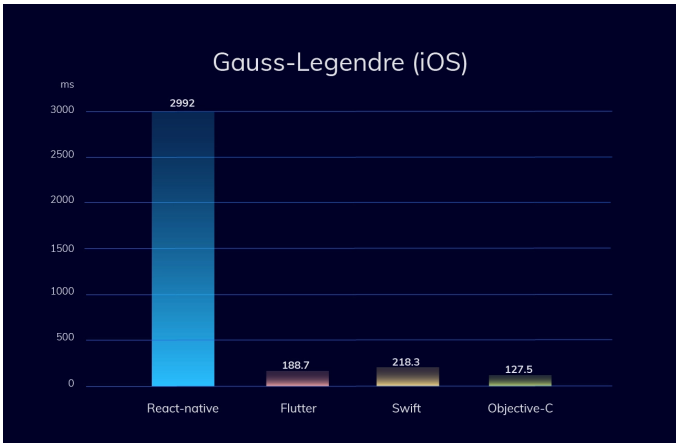
Figura 5 — Botão do aplicativo iFood no sistema operacional iOS.

Fonte: O Autor (2021)

Mas não é somente na interface do usuário que o Flutter e o React Native divergem: a forma como cada um se comunica com funções nativas também mudam. O aplicativo desenvolvido em Flutter é compilado para código nativo em tempo de compilação, enquanto o React Native faz esse processo em tempo de execução (GUEDES, 2020). Além disso, o React Native acessa os recursos nativos do dispositivo através do JavaScript, enquanto com o Flutter isto é feito de forma nativa (ANDRADE, 2020).

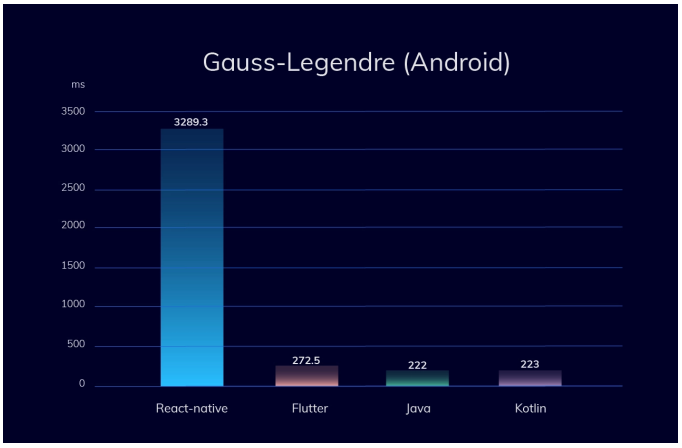
Por conta desta característica do Flutter, ele tende a obter os melhores resultados quando comparado com o React Native em questão de desempenho. Como já citado anteriormente, os testes de Demedyuk e Tsybulskyi (2020) apontam que o Flutter obteve um desempenho significativamente superior ao do React Native ao executar algoritmos que exigem memória e CPU, como é possível observar nas Figuras 6, 7, 8 e 9.

Figura 6 — Teste de memória com o algoritmo Gauss-Legendre no iOS.



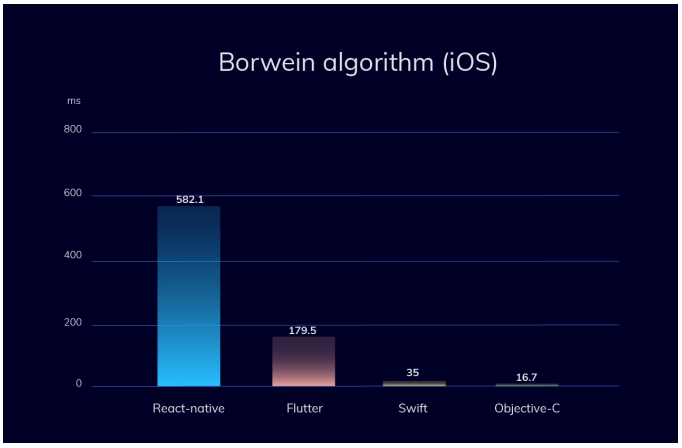
Fonte: Demedyuk e Tsybulskyi (2020)

Figura 7 — Teste de memória com o algoritmo Gauss-Legendre no Android.



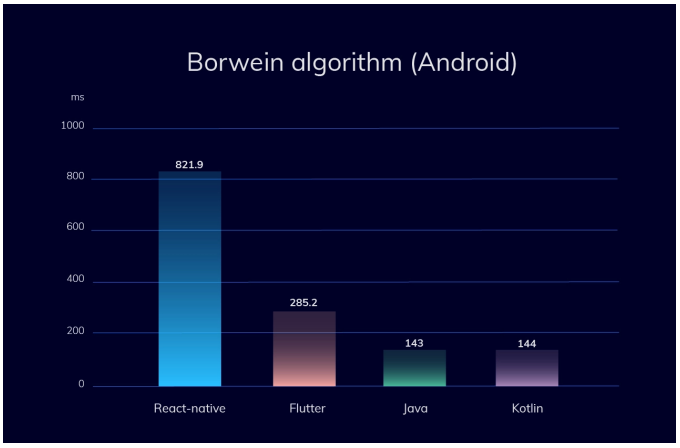
Fonte: Demedyuk e Tsybulskyi (2020)

Figura 8 — Teste de CPU com o algoritmo Borwein no iOS.



Fonte: Demedyuk e Tsybulskyi (2020)

Figura 9 — Teste de CPU com o algoritmo Borwein no Android.



Fonte: Demedyuk e Tsybulskyi (2020)

É necessário considerar também a linguagem de programação por trás do *framework*: o JavaScript já é consolidado no mercado, possui anos de evolução e uma comunidade de desenvolvedores enorme que podem ajudar a solucionar problemas ou ensinar coisas novas. Por outro lado, o Flutter utiliza Dart, uma linguagem nova e que, embora venha crescendo por conta do Flutter, ainda não possui uma comunidade muito grande.

REFERÊNCIAS

ANDRADE, A. P. **O que é o React Native?**. Disponível em: https://www.treinaweb.com.br/blog/o-que-e-o-react-native/. Acesso em: 18 jun. 2021.

ANDRADE, A.P. **O que é Flutter?**. Disponível em: https://www.treinaweb.com.br/blog/o-que-e-flutter. Acesso em: 18 jun. 2021.

ANDRADE, A.P. **O que é Ionic?**. Disponível em: https://www.treinaweb.com.br/blog/o-que-e-ionic. Acesso em: 18 jun. 2021.

CANGUÇU, R. **React Native no Desenvolvimento de Aplicativos Famosos**. Disponível em: https://codificar.com.br/react-native-no-desenvolvimento-de-aplicativos-famosos/. Acesso em: 18 jun. 2021.

CANGUÇU, R. **React Native vs. Flutter**. Disponível em: https://codificar.com.br/react-native-vs-flutter/. Acesso em: 18 jun. 2021.

DEMEDYUK, I; TSYBULSKYI, N. **Flutter vs Native vs React-Native: Examining Performance**. Disponível em: https://inveritasoft.com/blog/flutter-vs-native-vs-react-native-examining-performance. Acesso em: 22 mai. 2021.

DEVMEDIA. **Xamarin, Ionic e Cordova: Conheça o que são e as principais diferenças**. Disponível em: https://www.devmedia.com.br/xamarin-ionic-e-cordova-conheca-o-que-sao-e-as-principais-diferencas/37690. Acesso em: 18 jun. 2021.

GUEDES, M. **O que é Dart?**. Disponível em: https://www.treinaweb.com.br/blog/o-que-e-dart. Acesso em: 18 jun. 2021.

GUEDES, M. **React Native ou Flutter: por qual começar?**. Disponível em: https://www.treinaweb.com.br/blog/react-native-ou-flutter-por-qual-comecar/. Acesso em: 18 jun. 2021.

LIMA, F. F. D. Avaliação de *frameworks* para o desenvolvimento de aplicações híbridas. **Universidade Federal do Pampa**, Alegrete, v. 1, n. 1, p. 1-119, jun./2019.

APÊNDICES

APÊNDICE A

Fichas de Leitura

Autor:	Botella, Federico Escribano, Pedro Peñalver, Antonio	**Ano:**	2016
Título:	*Selecting the Best Mobile Framework for Developing Web and Hybrid Mobile Apps*	**Qualis:**	
Resenha: Este artigo de Botella, Escribano e Peñalver escrito em 2016 apresenta um estudo para selecionar a melhor estrutura para o desenvolvimento de aplicativos móveis, sendo que o resultado aponta para o fato de que os aplicativos híbridos são as melhores escolhas para desenvolver um aplicativo móvel. O estudo foi aplicado no Android, iOS e Windows Phone (sistema operacional que já chegou ao seu fim) e o aplicativo de amostra foi desenvolvido em Ionic *Framework*.			

Autor:	Küpper, Steffen Rausch, Andreas Andelfinger, Urs	**Ano:**	2018
Título:	*Towards the Systematic Development of Hybrid Software Development Processes*	**Qualis:**	
Resenha: Este artigo escrito em 2018 por Küpper, Rausch e Andelfinger fornece uma visão geral dos componentes dos métodos necessários e delineia como as abordagens de desenvolvimento híbridos podem ser implantadas em nível organizacional e adaptado ao nível do projeto.			

Autor:	Nunkesser, Robin	**Ano:**	2018
Título:	*Beyond Web/Native/Hybrid: A New Taxonomy for Mobile App Development*	**Qualis:**	

Resenha: Escrito em 2018 por Nunkesser, este artigo propõe uma nova taxonomia para o desenvolvimento de aplicativos móveis, sendo dividido os apps em seis categoria, sendo alguns deles destacados abaixo:

- Aplicativos endêmicos: desenvolvidos nas linguagens nativas dos sistemas operacionais (Java e Kotlin para Android e Swift e Objective-C para iOS)
- Aplicativos web: todos os principais sistemas operacionais para dispositivos móveis oferecem suporte a HTML, CSS e JavaScript.
- Aplicativos web híbridos: o Apache Cordova, por exemplo, permite criar aplicações híbridas para diferentes plataformas mobile com base no componente WebView, ou seja, este funciona como browser, mas sem barra de endereço ou botões para os usuários.
- Aplicativos de pontes híbridos: *frameworks* como Flutter e React Native também utilizam os motores JavaScript mas também permitem o uso de elementos endêmicos da interface do usuário ao invés de HTML.

Autor:	Hjort, Elin	**Ano:**	2020
Título:	*Evaluation of React Native and Flutter for cross-platform mobile application development*	**Qualis:**	

Resenha: Este estudo de 2020 realizado por Hjort avalia o Flutter e o React Native com o objetivo de determinar qual deles é mais adequado para os aplicativos móveis desenvolvidos pela Gambit (empresa finlandesa de desenvolvimento de softwares). A avaliação foi feita nos sistemas operacionais Android e iOS, considerando que são os mais usados. Nos resultados, o React Native apresentou os melhores resultados, embora ambas provaram ser opções viáveis para o desenvolvimento.

Autor:	Brito, Hugo Gomes, Anabela Santos, Álvaro Bernardino, Jorge	**Ano:**	2018

Título:	*JavaScript in mobile applications: React native vs ionic vs NativeScript vs native development*	**Qualis:**	

Resenha: Este artigo brasileiro escrito em 2018 por Brito, Gomes, Santos e Bernardino faz uma análise comparativa entre aplicações móveis nativas e híbridas baseadas em JavaScript, escritas em React Native, NativeScript e Ionic. Os resultados apontaram que o React Native apresenta os melhores resultados em quase todos os critérios analisados, tendo ainda a vantagem no desenvolvimento híbrido em relação ao nativo. Os critérios levados em consideração foram:

- Aprendizagem e qualidade na documentação.
- Custo de desenvolvimento.
- Emuladores e depuração.
- Tempo de resposta e velocidade.
- Reconhecimento comercial.
- Reutilização de código e trabalho em equipe.
- Manutenções e atualizações.

Autor:	Lima, Fernando Fortunato de	**Ano:**	2019
Título:	Avaliação de *frameworks* para o desenvolvimento de aplicações híbridas	**Qualis:**	

Resenha: Este artigo de 2019 escrito por Lima, tem como objetivo realizar a comparação entre *frameworks* de desenvolvimento híbrido. Foram desenvolvidos mini-aplicações (Bluetooth, camera, GPS e memória) em Flutter, Ionic, NativeScript-Vue.js e React Native e uma bateria de testes foi executada nos sistemas operacionais Android e iOS para obter as médias de tempo de execução, consumo de memória e uso do processador. Os resultados apontam, que o React Native é o *framework* que apresentou menor tempo de desenvolvimento e é o mais indicado para aplicações complexas, enquanto o Ionic se mostrou ser o menos aconselhado para aplicações complexas ou com acesso às funcionalidades nativas do dispositivo. O Flutter apresentou os menores valores médios dos testes de consumo de memória e uso do processador e o NativeScript-Vue.js apresentou os melhores tempos médios para o iOS, menos para a mini-aplicação Bluetooth.

Autor:	Neves, Jonathan Junior, Vilmar Mendes	**Ano:**	2020
Título:	Uma análise comparativa entre Flutter e React Native como *frameworks* para desenvolvimento híbrido de aplicativos mobile: estudo de caso visando produtividade	**Qualis:**	
Resenha: Este artigo de 2020 escrito por Neves e Junior analisou e comparou a produtividade de dois *frameworks* voltados para o desenvolvimento híbrido de aplicativos mobile: o Flutter e o React Native. Os resultados apontaram que ambos os *frameworks* são boas escolhas, a depender do nível de conhecimento do desenvolvedor: caso o mesmo possua conhecimento da linguagem JavaScript, o React Native é a melhor escolha, considerando que o é um *framework* de tal linguagem. Já, caso o desenvolvedor possua bons conhecimentos de POO (Programação Orientada à Objeto) o Flutter se mostra uma boa escolha, pois mesmo sendo em Dart, uma linguagem nova, é de fácil aprendizado.			

Autor:	Wu, Wenhao	**Ano:**	2018
Título:	*React Native vs Flutter, Cross-platforms mobile application frameworks*	**Qualis:**	
Resenha: Este trabalho publicado em 2018 por Wu, estuda algumas características importantes do React Native e do Flutter. Os estudos mostram que o React Native é a melhor escolha para iniciar um aplicativo multiplataforma, por conta da forte comunidade que o *framework* possui, entretanto, aposta como o Flutter tem um futuro brilhante por conta da consistência e organização da sintaxe e no nível do SDK.			

Autor:	Stender, Simon Åkesson, Hampus	**Ano:**	2020
Título:	*Cross-platform Framework Comparison: Flutter & React Native*	**Qualis:**	

Resenha: Esta tese desenvolvida em 2020 por Stender e Åkesson verifica qual *framework* é mais eficiente quando se trata de gestão de recursos e comparações gerais. Foi criado dois aplicativos semelhantes e realizado testes de desempenho nos mesmos para poder comparar os resultados. Os resultados apontam que o Flutter teve uma leve vantagem em relação ao React Native, mas que a diferença não é tão notável.

Autor:	Rieger, Christoph Majchrzak, Tim A.	**Ano:**	2019
Título:	*Towards the definitive evaluation framework for cross-platform app development approaches*	**Qualis:**	
Resenha: Este artigo de 2019 escrito por Rieger e Majchrzak apresenta um estudo que avalia vários *frameworks* e os compara com aplicativos web e desenvolvimento nativo. Os resultados apontam que desenvolvimento de plataforma cruzada (híbridos) tem tido muito progresso. Para avaliar, foram consideradas quatro perspectivas: infraestrutura, desenvolvimento, aplicativo e uso.			

Autor:	Rahman, Ashikur	**Ano:**	2020
Título:	*A Comparative Study of Hybrid Mobile Application Development*	**Qualis:**	
Resenha: Escrito em 2020 por Rahman, este artigo discute os prós e contras do desenvolvimento híbrido das aplicações mobile, apresenta e compara diferentes ferramentas, estrutura e também o desempenho dos aplicativos em comparação com o desenvolvimento nativo. Por mais que seja claro como os aplicativos nativos ainda são melhores quando visto na perspectiva de desempenho, o desenvolvimento híbrido não está muito atrás, considerando que é mais barato para desenvolver e é mais fácil de dar manutenção.			

Autor:	Presa, Marcus Käld	**Ano:**	2021

	Svensson, Oskar		
Título:	*React Native and native application development: A comparative study based on features & functionality when measuring performance in hybrid and native applications*	**Qualis:**	
Resenha: Este estudo publicado em 2021 por Presa e Svensson analisa o desempenho de aplicativos desenvolvidos em Kotlin (linguagem nativa para o Android) e React Native (*framework* de JavaScript utilizado para o desenvolvimento híbrido de aplicativos). Os resultados apontam que, como já era de se esperar, o aplicativo desenvolvido nativamente apresentou melhor desempenho.			

Autor:	Kuitunen, Mika	**Ano:**	2019
Título:	*Cross-Platform Mobile Application Development with React Native*	**Qualis:**	
Resenha: Este artigo publicado em 2019 por Kuitunen teve como objetivo descobrir se as tecnologias de desenvolvimento híbrido são viáveis para as aplicações móveis em termos de experiência, tanto do desenvolvedor quanto do usuário. A conclusão aponta para o fato de que, embora sejam uma opção prática, não está claro o quanto essas tecnologias possuem uma vantagem sobre o desenvolvimento nativo.			

Autor:	Olsson, Matilda	**Ano:**	2020
Título:	*A Comparison of Performance and Looks Between Flutter and Native Applications: When to prefer Flutter over native in mobile application development*	**Qualis:**	
Resenha: Publicado em 2020 por Olsson, este estudo analisa como o Flutter (*framework* da linguagem Dart utilizada para o desenvolvimento de aplicativos híbridos) se comporta em relação aos aplicativos nativos. A conclusão é de que o Flutter é melhor para ser utilizado em aplicações pequenas e médias e que possui potencial de crescimento para superar as desvantagens em relação aos aplicativos nativos.			

Autor:	Jagiello, Jakub	**Ano:**	2019
Título:	*Performance comparison between React Native and Flutter*	**Qualis:**	
Resenha: Publicado em 2019 por Jagiello, este estudo compara o desempenho dos dois *frameworks* de desenvolvimento híbrido mais utilizados até então: o Flutter e o React Native. Para medir o desempenho, foi considerado o número de quadros perdidos por um determinado tempo para duas aplicações diferentes, sendo que os resultados revelam que embora a diferença não fosse significativa, o React Native perdeu menos frames.			

Autor:	Fentaw, Awel Eshetu	**Ano:**	2020
Título:	*Cross platform mobile application development: a comparison study of React native vs Flutter*	**Qualis:**	
Resenha: Esta tese publicada em 2020 por Fentaw apresenta um estudo comparativo de dois aplicativos móveis multiplataformas desenvolvidos com Flutter e React Native. Os resultados mostram que uma das principais vantagens de usar o React Native é a forte comunidade de desenvolvedores que o *framework* possui, além de utilizar JavaScript, que por um lado é de fácil aprendizado e de outro pode ter efeitos negativos no desempenho. O Flutter, embora utilize Dart, que é uma linguagem pouco usada, possui diversos widgets de interface do usuário e que funcionam perfeitamente em diversas plataformas. Em geral, ambos os *frameworks* estão sob melhorias constantes, o que sugere que poderão suportar cargas mais próximas do desenvolvimento nativo do que apresentam agora.			

Autor:	Blokland, Erik	**Ano:**	2019
Título:	*An Empirical Evaluation of the User Interface Energy Consumption of React Native and Flutter*	**Qualis:**	
Resenha: Este estudo publicado em 2019 por Blokland compara o uso de energia durante a execução de aplicações desenvolvidas em Flutter e React Native. Os			

resultados não foram claros, entretanto foi possível identificar certas ações na interface do usuário que utilizavam mais ou menos energia.

Autor:	Schuler, Andreas Anderst-Kotsis, Gabriele	**Ano:**	2019
Título:	*Examining the Energy Impact of Sorting Algorithms on Android: An Empirical Study*	**Qualis:**	
Resenha: Este estudo publicado em 2019 por Schuler e Anderst-Kotsis é examinado a energia consumida durante a execução de 12 algoritmos de ordenação e o impacto resultante quando utilizado com diferentes tipos de dados. Os resultados mostram que a escolha do tipo de dado junto com o algoritmo pode ter um impacto significativo no perfil de energia de um aplicativo.			