ANÁLISE DO DESEMPENHO E DESENVOLVIMENTO DE APLICAÇÕES MOBILES, UTILIZANDO O FRAMEWORK FLUTTER E A BIBLIOTECA REACT NATIVE

Lucas Orestes Fabris, [1] Luciano Antunes [2]

Resumo: Considerando o fato de que o framework Flutter e a biblioteca React Native são as tecnologias predominantes atualmente no mercado de desenvolvimento mobile, foi realizado a análise das duas ferramentas, tendo como métricas o processo de desenvolvimento e o desempenho das duas tecnologias, aplicadas em dois aplicativos semelhantes, primeiro utilizando o framework Flutter e o segundo utilizando a biblioteca React Native. Estudos com temas semelhantes também vem sendo conduzidos atuamente, um deles se concentrou na comparação de desempenho de aplicativos criados com o framework Flutter e biblioteca React Native em relação às abordagens de desenvolvimento nativo para Android e iOS. Essas pesquisas contribuiram significativamente para o campo do desenvolvimento de aplicativos móveis e auxiliam na seleção adequada de frameworks para abordagens multiplataforma.
Dentre os motivos que levaram a comparar estas tecnologias está a gratuidade de ambas, por serem as mais utilizadas no mercado de desenvolvimento e apresentar a comunidade possiveis alternativas para desenvolvimento mobile. Os resultado obtidos a partir deste estudo indicam que o framework Flutter demonstrou um alto grau de organização, eficiência no processo de desenvolvimento e ótimas métricas de desempenho. Em contrapartida, o React Native se destacou como uma escolha mais consolidada e amadurecida, mas que não teve bons resultados de desempenho.

Palavras-chave: Desenvolvimento de aplicações. Flutter. React Native.

[1]Lucas Orestes Fabris. lucasfabris28@hotmail.com
[2]Luciano Antunes. luc@unesc.net

ABSTRACT: Considering the fact that the Flutter framework and the React Native library are the predominant technologies in the mobile development market today, an analysis of the two tools was carried out, using the development process and the performance of the two technologies as metrics, applied to two similar applications, the first using the Flutter framework and the second using the React Native library.
Studies on similar topics are also currently being conducted, one of which focused on comparing the performance of applications created with the Flutter framework and the React Native library in relation to native development approaches for Android and iOS. This research has made a significant contribution to the field of mobile application development and helps in the proper selection of frameworks for cross-platform approaches.
Among the reasons for comparing these technologies is that they are both free, are the most widely used in the development market and present the community with possible alternatives for mobile development. The results obtained from this study indicate that the Flutter framework has demonstrated a high degree of organization, efficiency in the development process and excellent performance metrics. On the other hand, React Native stood out as a more consolidated and mature choice, but one that did not have good performance results.

Keywords: Flutter. React Native. Application development

1 INTRODUÇÃO

O mercado de dispositivos móveis está em constante crescimento, com cerca de 7.26 bilhões de usuários diariamente (Turner, 2023), gastando em média 4 a 5 horas de uso diário (Perez, 2022). Sendo Android e iOS os sistemas operacionais mais utilizados no mundo, detém juntos 98% dos usuários (StatCounter, 2023), sendo o principal foco para as empresas de desenvolvimento mobile.

Entretanto cada sistema possui sua particularidade, contendo diferentes linguagens de programação nativas, o que acaba demandando que o desenvolvimento de aplicações para os ambos os sistemas seja feito separadamente. Aplicativos nativos são aqueles desenvolvidos utilizando a linguagem de programação específica do sistema operacional escolhido (Smutný, 2012).

Como cada plataforma possui diferentes linguagens de programação é necessário desenvolver uma versão para cada plataforma, o que duplica o esforço, custo e tempo de desenvolvimento. Além do desenvolvimento inicial, a manutenção dos dois projetos também requer um esforço dobrado. Neste contexto, surgiram as aplicações híbridas, onde é realizado o desenvolvimento de apenas uma aplicação que pode ser utilizada em dispositivos com diferentes sistemas operacionais (Prezotto, 2016). Essa prática facilita o desenvolvimento inicial e reduz a manutenção necessária, evitando a necessidade de desenvolver para cada sistema operacional. Isso é possível graças aos frameworks de desenvolvimento de aplicações híbridas, que são responsáveis por empacotar o código-fonte para diferentes plataformas, permitindo que sejam instalados em diversos dispositivos (Prezotto, 2016).

O custo para a criação de cada aplicativo é alto, considerando o custo dos recursos, tempo, manutenção e evolução. Portanto, manter dois ou mais aplicativos é uma tarefa custosa (Lima, 2019). Neste sentido duas tecnologias vêm sendo muito utilizadas para o desenvolvimento de aplicações multiplataforma, o framework Flutter e a biblioteca React Native. Ambas as tecnologias são open source de desenvolvimento híbrido para aplicativos mobile. De acordo com a plataforma (GitHub, 2023), o React Native conta com 105 mil stars, e o Flutter com 146 mil stars (Flutter, 2023a), demonstrando desta forma a grande utilização dessas duas tecnologias atualmente (Statista, 2021).

Dentre os motivos que justificaram a escolha de comparar as duas tecnologias em questão, destaca-se a gratuidade integral de ambas as ferramentas. Essa característica singular facilita a realização de estudos aprofundados e o desenvolvimento detalhado das respectivas comparações, com o objetivo de apresentar à comunidade de desenvolvedores e à indústria de software possíveis alternativas na esfera do desenvolvimento móvel. Nesse contexto, foram adotadas métricas criteriosas para a análise do desempenho e do processo de desenvolvimento de cada framework.

Foram conduzidas pesquisas com temas semelhantes, se concentrando em aspectos relacionados ao desenvolvimento multiplataforma e à avaliação de desempenho de aplicativos móveis. Uma dessas pesquisas teve como objetivo avaliar as tecnologias disponíveis para o desenvolvimento multiplataforma (Mota, 2020). Um estudo propôs um processo abrangente para a comparação de desempenho de aplicações móveis (Prezotto, 2016). Além disso, outro estudo se concentrou na comparação de desempenho de aplicativos criados com o framework Flutter e biblioteca React Native em relação às abordagens de desenvolvimento nativo para Android e iOS(Paim; Júnior, 2022). Essas pesquisas contribuem significativamente para o campo do desenvolvimento de aplicativos móveis e auxiliam na seleção adequada de frameworks para abordagens multiplataforma.

Com o objetivo de determinar a tecnologia de desenvolvimento multiplataforma mais adequado, este estudo visa comparar as características do framework Flutter e a biblioteca React Native. A análise abrangerá uma variedade de aspectos, incluindo a análise de desempenho avaliando o consumo de CPU, consumo de memória RAM, número de linhas de código e FPS. O processo de desenvolvimento abrangerá: tempo de execução, documentação disponível, curva de aprendizagem e a comunidade.

2 MATERIAIS E MÉTODOS

O desenvolvimento do aplicativo foi realizado em um computador com as seguintes especificações: 16GB de memória RAM, processador Intel Core i3-8100 de 4 núcleos a 3,6GHz e uma placa de vídeo ZOTAC GeForce GTX 1060 de 6GB. O sistema operacional utilizado foi o Windows 10 Pro.

Para a codificação, utilizamos a IDE Visual Studio Code, uma ferramenta amplamente adotada por desenvolvedores de software, conhe-

cida por suas diversas funcionalidades que simplificam o processo de desenvolvimento de aplicações, incluindo sugestões de código e formatação automática. Nos testes e na depuração dos aplicativos, empregamos um modelo de smartphone com sistema operacional Android.

O celular selecionado foi o Xiaomi Redmi Note 11 Pro 5G, com 128GB de armazenamento e executando o Android 13, equipado com 6GB de RAM. A escolha desse modelo levou em consideração a ampla utilização do Android como plataforma em dispositivos móveis. Foram desenvolvidas duas aplicações: uma utilizando o framework Flutter e a segunda utilizando a biblioteca React Native.

2.1 DESENVOLVIMENTO DA APLICAÇÃO COM FLUTTER

Para o desenvolvimento da primeira aplicação, optamos por utilizar o framework multiplataforma Flutter, na versão 3.72.0 (Flutter, 2023b). A escolha do Flutter como plataforma de desenvolvimento foi motivada por sua ampla popularidade e eficácia na criação de aplicativos voltados para diversas plataformas, incluindo os sistemas operacionais Android e iOS.

Além disso, para configurar o ambiente de desenvolvimento, foram instaladas as extensões relevantes para a linguagem de programação Dart e o framework Flutter no Visual Studio Code. Essas extensões proporcionam uma experiência de desenvolvimento mais eficiente, simplificando a codificação e a depuração de aplicativos Flutter.

2.2 DESENVOLVIMENTO DA APLICAÇÃO COM REACT NATIVE

O desenvolvimento da segunda aplicação adotou o uso da biblioteca React Native, que é amplamente empregada no desenvolvimento de aplicações multiplataforma, na versão 0.72 (ReactNative, 2023). Para o desenvolvimento do projeto, optamos pela utilização da versão React Native CLI, adequada para aplicações mais complexas presentes no mercado. Em ambos os casos, utilizamos o gerenciador de pacotes Chocolatey para facilitar a instalação de pacotes. No React Native, o Chocolatey foi empregado para instalar o NodeJS e o Java 11.

Vale ressaltar que, como parte da configuração do ambiente de desenvolvimento, optamos por instalar a versão 11 do Java. Essa decisão

foi tomada considerando a possível ocorrência de problemas de compatibilidade que poderiam surgir ao utilizar versões mais recentes do Java. A compatibilidade é uma preocupação crítica para garantir que tanto a aplicação Flutter quanto a React Native sejam executadas de forma estável e consistente em todas as plataformas-alvo.

2.3 MÉTRICAS UTILIZADAS

Uma das principais áreas de investigação deste estudo foi o processo de desenvolvimento das aplicações, que incluiu a análise de diversas métricas e fatores. Com o objetivo de apresentar e compreender os conhecimentos necessários para o uso eficaz das tecnologias apresentadas.

2.3.1 MÉTRICAS DE DESENVOLVIMENTO

Foi avaliado o número de linhas de código necessário para a implementação de funções em ambas as tecnologias. Esse procedimento permitiu uma análise aprofundada da eficiência do processo de desenvolvimento em cada uma delas. O objetivo dessa análise foi identificar qual das linguagens de programação se mostra mais acessível, tanto no desenvolvimento inicial quanto na manutenção do código, enfatizando a clareza da estrutura para leitura e revisão. Isso, por sua vez, visa aprimorar a compreensão tanto de programadores novatos quanto de profissionais experientes.

Avaliou-se a curva de aprendizado associada a cada uma das tecnologias apresentadas. Esta avaliação envolveu a observação do tempo necessário para que um desenvolvedor adquira proficiência em cada uma das tecnologias em análise. Também foi analisado o período necessário para que um desenvolvedor iniciante alcance um nível de familiaridade e conforto com a tecnologia. Levamos em consideração os recursos disponíveis para aprendizado, abrangendo elementos como tutoriais, cursos, literatura especializada e outros materiais adicionais, contemplando a clareza de cada linguagem de programação e a estrutura da tecnologia em questão, com o objetivo de identificar e influência desses fatores na curva de aprendizado.

Outro componente essencial desta pesquisa consistiu na análise da documentação associada a cada uma das tecnologias em estudo.

A qualidade e a abrangência da documentação desempenham um papel crucial no sucesso de um projeto de desenvolvimento de software. Os critérios utilizados para a avaliação da documentação incluíram a clareza e organização, a disponibilidade de exemplos de código e tutoriais para simplificar o processo de aprendizado, a inclusão de informações abrangentes sobre das funcionalidades e recursos da tecnologia, e a frequência de atualizações da documentação, garantindo a sua conformidade com as versões mais recentes da tecnologia em questão.

Foi realizada à avaliação das comunidades relacionadas a cada uma das tecnologias examinadas com o objetivo de compreender a dinâmica da interação dos usuários no contexto de dúvidas e resolução de problemas durante o processo de desenvolvimento de aplicações baseadas em cada tecnologia. A participação ativa de uma comunidade de desenvolvedores pode desempenhar um papel crucial no sucesso de um projeto. A análise da comunidade considerou a interação e o envolvimento dos seus membros em diferentes plataformas, como fóruns, grupos de discussão, redes sociais, entre outros. Os critérios estabelecidos para essa avaliação incluíram a dimensão da comunidade, ou seja, o número de desenvolvedores ativos que dela fazem parte, bem como a atividade da comunidade, medida a partir do volume de perguntas e respostas geradas, as quais refletem a participação e o auxílio mútuo entre os membros.

2.3.2 MÉTRICAS DE DESEMPENHO

Por fim, também analisamos as métricas de desempenho das aplicações desenvolvidas. Para avaliação de desempenho, utilizamos um recurso disponibilizado pela empresa Xiaomi, empresa desenvolvedora do celular Redmi Note 11 Pro, onde os testes foram realizados. Esse recurso é chamado 'Consumo de Energia' e permite medir a quantidade de memória RAM utilizada, a taxa de quadros por segundo (FPS) e o uso da CPU.

Também utilizamos o aplicativo 'Resource Monitor Mini', disponível na Play Store, para auxiliar na avaliação de desempenho, analisando o uso da CPU e a quantidade de memória RAM disponível.

Neste contexto, avaliamos o uso da CPU, que mede a porcentagem de tempo em que a CPU do dispositivo é utilizada para executar tarefas. Um alto e constante uso da CPU pode indicar uma carga intensa de processamento, e dependendo do contexto, pode resultar em diminuição

do desempenho, atrasos ou lentidão no funcionamento do smartphone.

Foi avaliado o Frames Por Segundo (FPS), uma métrica amplamente utilizada em smartphones, especialmente em jogos e aplicativos gráficos, para medir a fluidez e a qualidade da exibição de imagens em movimento. O FPS representa a quantidade de quadros renderizados por segundo. Um FPS mais alto indica uma experiência de uso mais suave e responsiva para o usuário, garantindo uma visualização fluida e sem interrupções.

Foi avaliado o consumo de memória RAM, uma métrica importante para a avaliação de desempenho em smartphones. Essa métrica mede a quantidade de memória RAM utilizada pelo sistema operacional ou por aplicativos específicos. Um alto consumo de RAM pode resultar em lentidão do sistema e ter um impacto negativo no desempenho geral, especialmente quando a memória disponível se aproxima de sua capacidade máxima. Portanto, é essencial monitorar e gerenciar o consumo de RAM para garantir um funcionamento fluido do smartphone e evitar problemas de desempenho.

Foi verificado o tempo de execução em smartphones refere-se à quantidade de tempo necessária para completar uma tarefa ou operação específica, como a abertura do aplicativo. Essa métrica é usada para avaliar a eficiência e o desempenho geral de um dispositivo. Um tempo de execução menor indica maior eficiência e desempenho aprimorado. Vários fatores influenciam o tempo de execução, incluindo o processador, a quantidade de memória RAM e otimizações de software. Avanços nos componentes de hardware e melhorias no software desempenham um papel fundamental na redução do tempo de execução em smartphones, proporcionando uma experiência mais ágil e responsiva para os usuários.

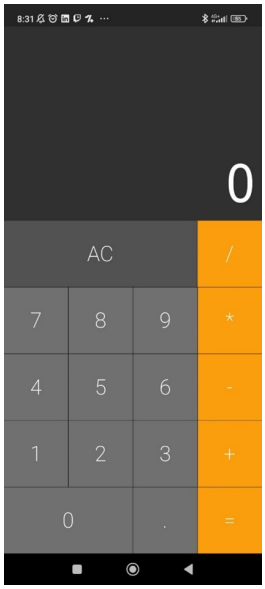
Ao analisar esses aspectos, este estudo visa fornecer informações para desenvolvedores e equipes de projeto que precisam tomar decisões sobre a escolha entre React Native e Flutter para o desenvolvimento de aplicações multiplataforma. Os resultados obtidos ajudarão a identificar as vantagens e desvantagens de cada abordagem, contribuindo para desenvolvimento de aplicações móveis mais eficiente e eficaz .

2.4 INTERFACE DA APLICAÇÃO

Com o objetivo de atingir a meta de criar duas aplicações semelhantes para possibilitar a comparação das métricas definidas para ambas,

desenvolvemos duas calculadoras para conduzir os testes. Durante o processo de desenvolvimento, optamos por um design simplificado,uma vez que nosso foco principal não era a estética, mas sim a análise das métricas relevantes. Como resultado, cada aplicativo apresenta uma única tela com botões numerados de 1 a 9, símbolos de operações e um visor para exibir os resultados das operações, como pode ser visto na Figura 1.

Figura 1 - Design escolhido para os dois Apps.



Fonte: Elaborado pelo autor (2023).

3 DISCUSSÃO E RESULTADOS

Este capítulo apresenta os resultados obtidos a partir dos experimentos realizados, divididos pelas métricas de desempenho e desenvolvimento que foram previamente apresentadas.

3.1 TEMPO DE EXECUÇÃO

O Tempo de Execução de uma aplicação é uma métrica que representa o intervalo de tempo desde o momento em que o usuário seleciona o ícone do aplicativo até o instante em que ele pode começar a interagir com as funções da aplicação.

Para obter os resultados, avaliamos o tempo de execução das duas aplicações. Utilizamos um cronômetro para medir o tempo e executamos a mesma aplicação várias vezes, garantindo que o cache do celular fosse limpo para obter os melhores resultados possíveis. Ao analisar a média de tempo de cada uma.

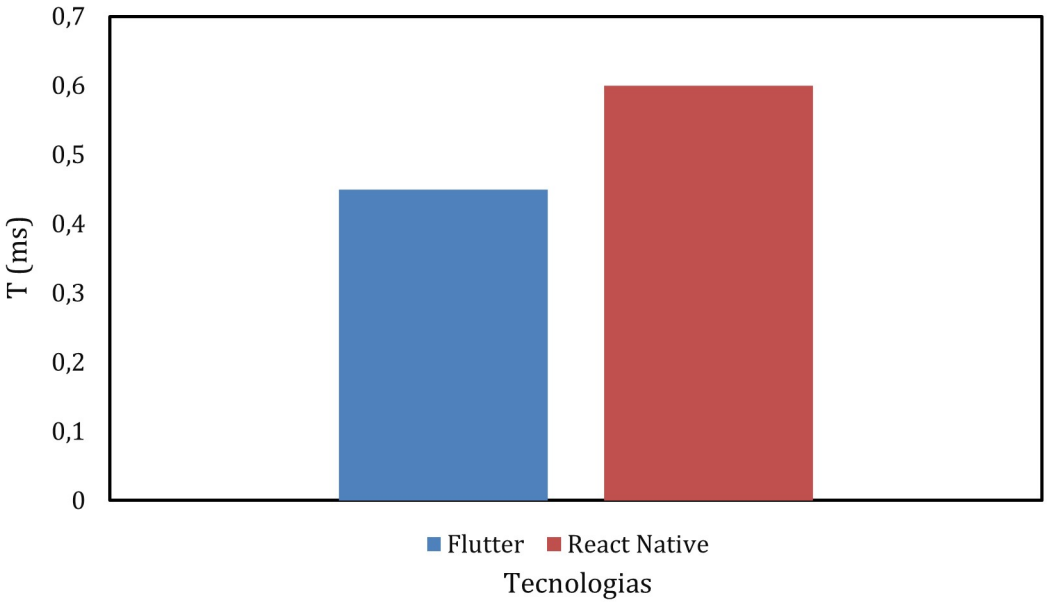
Conforme observado na Figura 3, apresentamos as implementações da aplicação em Flutter e da aplicação em React Native. Os valores demonstram que a inicialização da aplicação usando React Native foi mais lenta, com 0,60 ms, enquanto a aplicação em Flutter inicializou em 0,45 ms, resultando que o Flutter obteve um tempo de execução mais rápido.

Outro trabalho semelhante obteve resultados semelhantes em suas comparações de tempo de execução. A pesquisa realizada por Paim e Júnior (2022) chegou a resultados em que sua aplicação em Flutter iniciou mais rápido do que em React Native. No caso, a aplicação em Flutter iniciou em 1138 ms, enquanto a aplicação em React Native iniciou em 1141 ms, apresentando pouca diferença entre ambas tecnologias, mas com o Flutter iniciando ligeiramente mais rápido.

3.2 FPS

FPS é uma métrica amplamente utilizada em smartphones, especialmente em jogos e aplicativos gráficos, para medir a fluidez e a qualidade da exibição de imagens em movimento. Vale ressaltar que o celular

Figura 2 - Tempo de execução



Fonte: Elaborado pelo autor (2023).

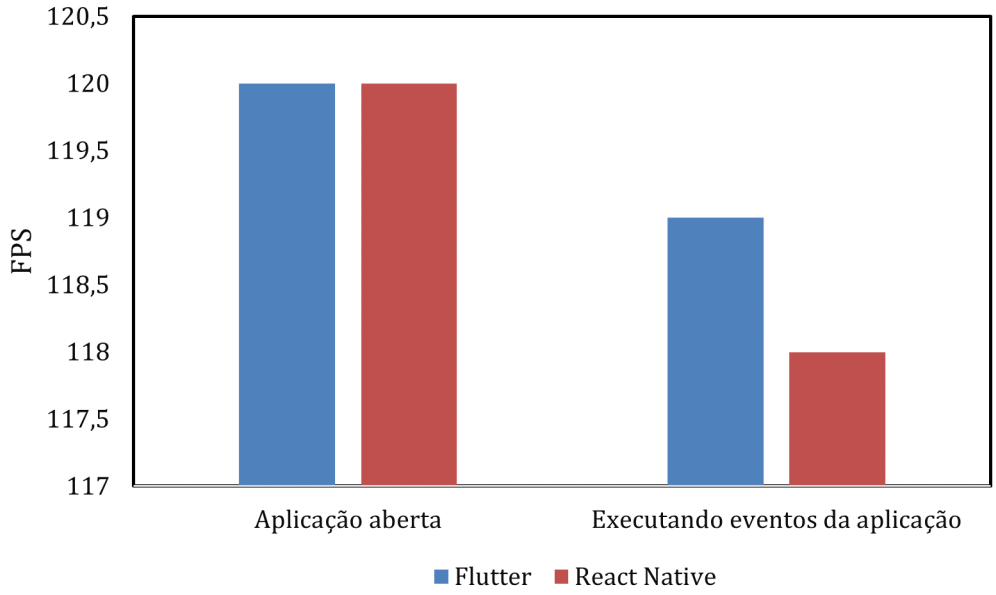
usado para os testes possui uma taxa de atualização de 120 Hz, o que significa que o máximo de FPS que pode ser atingido é 120 FPS.

Os resultados obtidos nesta análise foram obtidos utilizando o recurso disponibilizado pela Xiaomi, a empresa desenvolvedora do celular Redmi Note 11 Pro, onde os testes foram realizados. Esse recurso é chamado 'Consumo de Energia' e permite medir diversas métricas, incluindo o consumo de energia.

Ambas as aplicações desenvolvidas apresentaram uma média de 120 FPS enquanto a aplicação estava apenas aberta. No entanto, ao executar algumas funções das aplicações, como clicar em botões e realizar animações, observamos que a média de FPS da aplicação em Flutter obteve uma média de 119 FPS, enquanto a aplicação em React Native teve uma média de 118 FPS. Isso ocorreu ao pressionar os botões várias vezes seguidas. Portanto, podemos concluir que a aplicação em Flutter obteve uma média de FPS ligeiramente melhor no cenário de aplicação apresentado, como ilustrado na Figura 4.

O trabalho de Mota (2020) obteve resultados semelhantes no processo de execução da aplicação e execução de eventos, considerando

Figura 3 - Média de FPS



Fonte: Elaborado pelo autor (2023).

que seus testes atingiam no máximo 60 FPS. Pode-se observar que a aplicação em Flutter teve a melhor média de FPS, atingindo uma média de 52,5 FPS após 10 testes, enquanto a aplicação em React Native não manteve uma média tão alta de FPS, apresentando uma média de 38,8 FPS após 10 testes. Isso indica que a aplicação em Flutter mantém uma média de FPS mais alta e mais estável durante a execução, enquanto a aplicação desenvolvida em React Native teve uma média menor e mostrou maior oscilação no FPS.

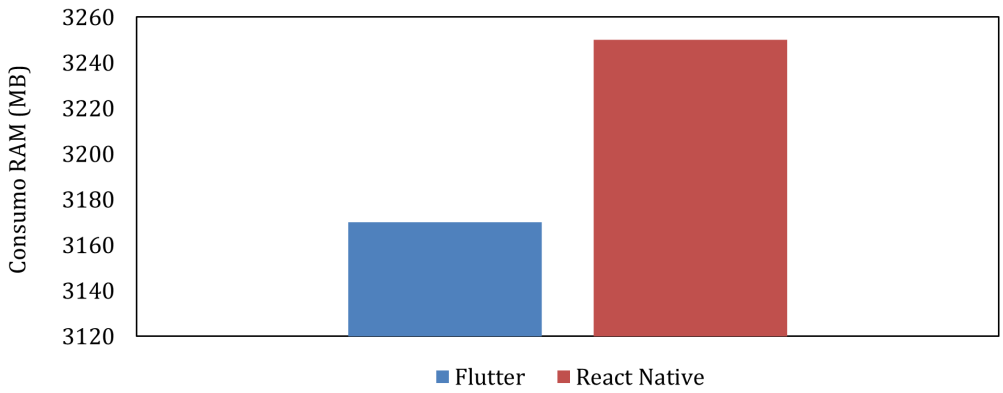
3.3 MEMÓRIA RAM

Para comparar o consumo de memória RAM em cada aplicação, utilizamos o recurso fornecido pela empresa Xiaomi, que já estava disponível no celular de teste, chamado 'Consumo de Energia'. Isso nos permitiu monitorar o consumo de memória RAM.

Conforme ilustrado na Figura 5, observamos uma pequena diferença no consumo de memória RAM. A aplicação em Flutter consumiu 3170 MB de RAM, enquanto a aplicação em React Native consumiu 3250 MB de

RAM, o que resultou em um consumo ligeiramente maior de memória RAM pela aplicação em React Native.

Figura 4 - Consumo de memória RAM



Fonte: Elaborado pelo autor (2023).

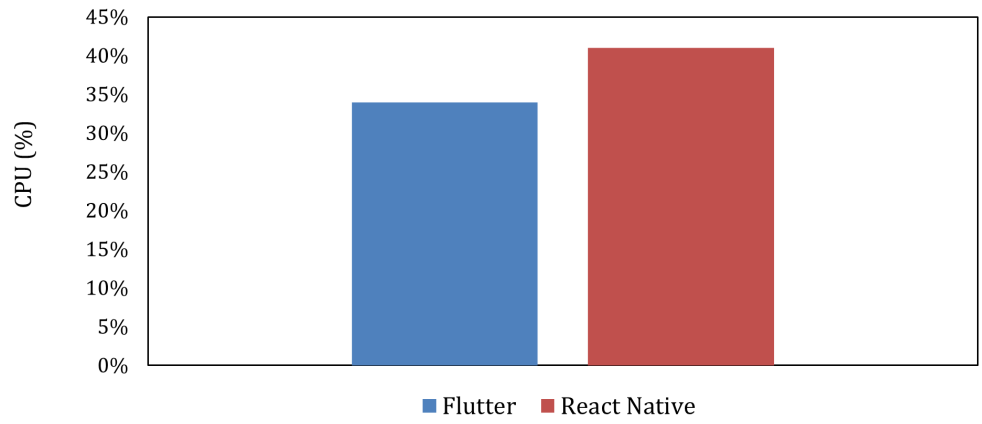
O trabalho desenvolvido por Paim e Júnior (2022) chegou ao resultado em que a aplicação desenvolvida em React Native teve uma média de consumo de memória RAM maior, atingindo 5%, enquanto a aplicação desenvolvida em Flutter obteve uma média de 4%, o que resultou em um consumo menor de memória RAM. Portanto, podemos concluir que a aplicação desenvolvida em Flutter consome uma quantidade menor de memória RAM em comparação com a aplicação desenvolvida em React Native.

3.4 CPU

Para realizar a comparação e monitoramento da CPU das aplicações, utilizamos um aplicativo auxiliar chamado ’Resource Monitor Mini’, que fornece informações sobre a porcentagem de CPU utilizada por cada aplicação.

Conforme ilustrado na Figura 6, observamos que o consumo médio de CPU das duas aplicações em Flutter atingiu uma média de uso de CPU de 34%, enquanto o consumo da aplicação em React Native foi maior, atingindo 41%. Isso resultou em um consumo de CPU maior por parte da aplicação em React Native.

Figura 5 - Consumo de CPU



Fonte: Elaborado pelo autor (2023).

O estudo desenvolvido por Paim e Júnior (2022) obteve resultados semelhantes, onde a média de CPU da aplicação desenvolvida em Flutter foi de 20%, o que é menor do que a média da aplicação desenvolvida em React Native, que teve uma média de 25% de uso da CPU. Isso leva à conclusão de que o Flutter apresentou resultados mais satisfatórios no quesito uso de CPU.

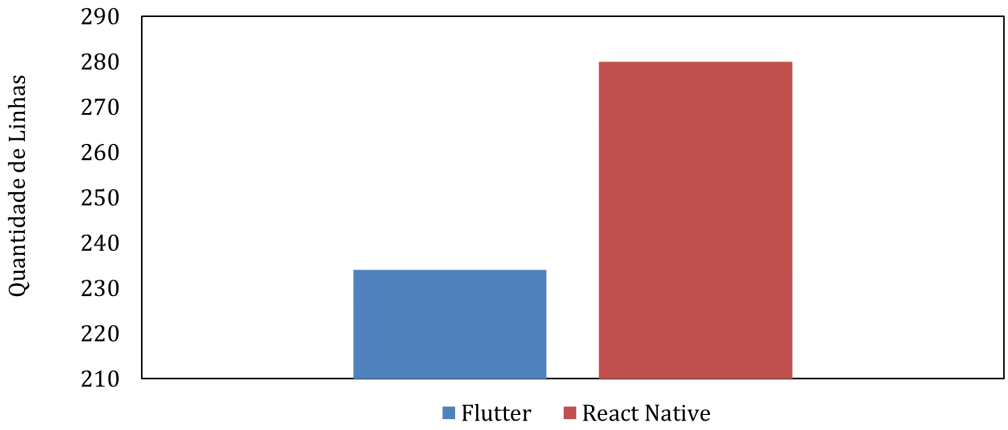
3.5 LINHAS DE CÓDIGO

A quantidade total de linhas de código de cada projeto foi avaliada com o objetivo de observar qual tecnologia exige um maior esforço de desenvolvimento, o que, por sua vez, pode influenciar o tempo necessário para a manutenção de cada projeto.

Como observado na Figura 7, podemos ver que para desenvolver a aplicação em React Native foram necessárias 280 linhas de código, enquanto a aplicação em Flutter exigiu 230 linhas. Isso leva à conclusão de que a ferramenta React Native requer uma quantidade maior de código para o desenvolvimento. Pode-se concluir que o React Native teve um requisito de codificação mais extenso em comparação com o Flutter para a criação das respectivas aplicações.

O estudo realizado por Silva (2021) chegou a uma conclusão semelhante, onde sua aplicação em React Native exigiu o desenvolvimento

Figura 6 - Linhas de Código



Fonte: Elaborado pelo autor (2023).

de 594 linhas de código, levando em consideração que ele desconsiderou a contagem de linhas utilizadas no CSS. Já o desenvolvimento em Flutter necessitou de 469 linhas de código. Isso leva à conclusão de que o desenvolvimento em React Native é mais verboso, enquanto o Flutter exige menos esforço de codificação

3.6 CURVA DE APRENDIZADO

Durante o processo de curva de aprendizagem, avaliamos as principais dificuldades encontradas em cada uma das tecnologias e o tempo necessário para desenvolver em cada uma delas.

3.6.1 FLUTTER

O processo inicial de desenvolvimento da aplicação em Flutter foi simples e de fácil entendimento. Seguindo a documentação e instalando os recursos necessários, o processo foi tranquilo. No entanto, o processo de estilização no Flutter foi mais desafiador, pois tudo se baseia em seus Widgets, o que adicionou uma camada de aprendizado. Com tempo e prática, as coisas se tornaram mais fáceis, e a ajuda da documentação e pesquisas na internet foi fundamental para desenvolver e estruturar uma boa aplicação

3.6.2 REACT NATIVE

O processo inicial de desenvolvimento da aplicação em React Native foi um pouco mais confuso, pois desde o início havia duas opções de modelos de aplicação. Poderíamos optar por utilizar o Expo Go, uma ferramenta que visa facilitar o processo de desenvolvimento de aplicativos, ou o React Native CLI, que contém as versões mais atualizadas do React Native, permitindo o desenvolvimento de aplicações mais estruturadas e sempre atualizadas para evitar problemas. Para nossa aplicação, optamos pela versão CLI.

Durante o processo de instalação, é crucial prestar muita atenção em todas as versões que devem ser instaladas, seguindo a documentação do React, a fim de evitar problemas futuros. Após a instalação, o processo de aprendizado é simples, e o desenvolvimento de layouts é intuitivo e direto. É possível definir arquivos globais que podem ser chamados e aplicados por padrão em todas as telas, o que torna a manutenção mais simples e fácil.

3.6.3 RESULTADO DA COMPARAÇÃO

O React Native adota uma abordagem de estilização semelhante à usada na web, empregando HTML e CSS. Enquanto isso, o Flutter utiliza a linguagem Dart e possui um método de estilização distinto. No entanto, o Flutter demanda que o desenvolvedor se familiarize com a personalização do layout, o que pode aumentar um pouco a dificuldade para aprender. Ao dominar as propriedades de layout e possuir um conhecimento básico de CSS e HTML, a criação de interfaces de usuário no React Native se torna relativamente simples, não exigindo um amplo conhecimento por parte do desenvolvedor. Conclui-se que o React Native é uma ferramenta mais fácil de aprender, especialmente para iniciantes.

O estudo desenvolvido por Silva (2021) observou que, devido ao fato do React Native utilizar HTML e CSS, a curva de aprendizagem se torna mais suave, especialmente para desenvolvedores com conhecimento prévio em desenvolvimento web, pois proporciona familiaridade com a sintaxe e permite um progresso rápido no domínio do framework. Por outro lado, o Flutter possui uma curva de aprendizagem mais íngreme, uma vez que requer o conhecimento da linguagem Dart, que não é amplamente uti-

lizada, e, portanto, exige uma dedicação extra para aprender os aspectos da linguagem.

3.7 DOCUMENTAÇÃO

A qualidade e a abrangência da documentação desempenham um papel crucial no sucesso de um projeto de desenvolvimento de software. Os critérios utilizados para avaliar a documentação incluíram a clareza e organização, a disponibilidade de exemplos de código e tutoriais para facilitar o aprendizado, a inclusão de informações abrangentes sobre funcionalidades e recursos da tecnologia, e a frequência de atualizações da documentação para garantir a compatibilidade com as versões mais recentes da tecnologia em questão.

3.7.1 FLUTTER

A documentação do Flutter é um dos pontos fortes do framework. O processo de aprendizado e o início do desenvolvimento são claros e intuitivos. A documentação explica de forma precisa como configurar o framework em todas as plataformas e fornece um passo a passo para a criação do primeiro projeto.

Além disso, a documentação fornece todos os recursos necessários para que o desenvolvedor compreenda como utilizar widgets, plugins e realizar o deploy da aplicação. Tudo é explicado de forma clara e acessível a todos os tipos de desenvolvedores. Para iniciantes, a documentação serve como uma base sólida para iniciar os estudos do framework, eliminando a necessidade de assistência externa, pois é abrangente e completa.

3.7.2 REACT NATIVE

A documentação do React Native, inicialmente, pode parecer um pouco confusa devido à falta de organização, o que pode confundir os iniciantes que estão se familiarizando com a biblioteca. Além disso, encontrar respostas para possíveis dúvidas pode ser um desafio, uma vez que os exemplos frequentemente estão desatualizados.

A resolução de problemas acaba sendo um desafio, uma vez

que encontrar soluções na própria documentação pode ser difícil. Muitas vezes, os desenvolvedores precisam recorrer a fontes externas, como sites e fóruns, para obter ajuda, o que pode ser um fator desestimulante para aqueles que desejam aprender mais sobre essa biblioteca.

3.7.3 RESULTADO DA COMPARAÇÃO

O Flutter possui uma documentação excelente e altamente organizada, com detalhes abrangentes sobre seus widgets, facilitando sua localização. Por outro lado, o React Native carece de uma documentação tão organizada e, em alguns casos, pode estar desatualizada, o que representa uma desvantagem para a biblioteca, tornando o Flutter superior nesse aspecto.

O estudo realizado por Silva (2021) chegou a conclusão que o uso da documentação oficial do Flutter é mais satisfatória, pois é detalhada e simples, o que facilita o entendimento e resolução de problemas, tornando-a adequada para o público em geral. Por outro lado, a análise da documentação do React Native resultou na constatação de que a documentação oficial não é tão satisfatória, uma vez que foi necessário realizar consultas fora da documentação para esclarecer dúvidas e resolver problemas.

3.8 COMUNIDADE

Foi realizada uma avaliação das comunidades relacionadas a cada uma das tecnologias examinadas com o objetivo de compreender a dinâmica da interação dos usuários no contexto de dúvidas e resolução de erros durante o processo de desenvolvimento de aplicações com base em cada tecnologia.

3.8.1 FLUTTER

A comunidade do Flutter é ampla e ativa, oferecendo uma variedade de recursos online. Ela inclui uma diversidade de vídeos que apresentam a tecnologia, tutoriais tanto gratuitos quanto pagos e uma ampla gama de cursos que abrangem desde questões básicas até avançadas. Fóruns

estão disponíveis para auxiliar na correção de problemas dos desenvolvedores e para apresentar as atualizações disponíveis na tecnologia.

Os membros da comunidade frequentemente desenvolvem clones de interfaces de aplicativos populares e compartilham seus projetos utilizando a tecnologia. Apesar de o Flutter ser uma tecnologia relativamente recente, lançada em maio de 2017, ele está em constante crescimento, contando com uma comunidade ativa e uma abundância de recursos disponíveis online.

3.8.2 REACT NATIVE

A comunidade do React Native é, igualmente, grande e ativa, oferecendo uma variedade de conteúdos disponíveis para o seu público. Inclui tutoriais, tanto gratuitos e cursos pagos, além de uma ampla gama de cursos que abrangem desde questões básicas até avançadas. Fóruns estão disponíveis para auxiliar na correção de problemas dos desenvolvedores e para apresentar as atualizações disponíveis na tecnologia. Ao pesquisar sobre a tecnologia, é possível encontrar diversos resultados e projetos desenvolvidos pela comunidade.

O fato de a linguagem de desenvolvimento ser o JavaScript e a semelhança com o ReactJS facilitam a resolução de erros, uma vez que as soluções para problemas semelhantes no ReactJS podem ser aplicadas no React Native. Essa afinidade entre as tecnologias contribui para uma transição mais suave e eficiente para os desenvolvedores que já têm experiência com o ReactJS.

3.8.3 RESULTADO DA COMPARAÇÃO

Tanto o React Native quanto o Flutter possuem comunidades ativas, mas o React Native, por sua vez, apresenta uma quantidade maior de conteúdo disponível. Isso significa que os desenvolvedores que utilizam o React Native têm mais chances de encontrar soluções para problemas ou funcionalidades na internet. O Flutter, sendo um framework relativamente novo, está com seu conteúdo online em constante crescimento, e o número de recursos disponíveis para ele aumenta diariamente.

Com base nos dados desta pesquisa, é evidente que o Flutter, como um dos frameworks mais recentes, experimentou um notável cresci-

mento, sugerindo que continuará sendo uma das preferências da comunidade no futuro próximo. Por outro lado, o React Native também demonstra ter uma grande e empenhada comunidade, focada no crescimento da tecnologia, buscando sempre atrair mais desenvolvedores para a plataforma. Sua vantagem reside em ser uma tecnologia mais antiga e já bem estruturada no mercado.

A pesquisa desenvolvida por Mota (2020) analisou a comunidade de ambas as tecnologias com base em dados do GitHub. A conclusão foi que o Flutter apresentou um crescimento significativo na comunidade, com aumento de 19% em estrelas, 27% em forks e 12% em watchs. Em contraste, o React Native, sendo uma tecnologia mais madura com um grande número de usuários, registrou um crescimento menor, com 6% em estrelas, 5% em forks e nenhum crescimento em watchs.

4 CONCLUSÃO

No decorrer do estudo, foram desenvolvidas duas aplicações: uma utilizando Flutter e a outra utilizando React Native, com o objetivo de analisar o processo de desenvolvimento de ambas as aplicações e comparar seu desempenho com base em várias métricas.

Com base nos resultados apresentados, é possível observar que, mesmo sendo uma tecnologia recente, o Flutter demonstrou ser bem organizado, apresentando uma documentação robusta. Sua comunidade mostrou-se unida, com uma curva de aprendizado considerável, reconhecendo que a linguagem Dart pode representar um desafio no início do desenvolvimento da aplicação. O desempenho do Flutter foi excelente, destacando-se em termos de CPU, Linhas de Código, FPS e consumo de RAM, o que resultou em uma aplicação leve e eficiente.

Por outro lado, o React Native mostrou ser uma escolha madura, com uma comunidade consolidada e fóruns para esclarecimento de dúvidas e resolução de problemas. No entanto, sua documentação deixou um pouco a desejar, sendo desorganizada e confusa para iniciantes. Apresentando também resultados de desempenho inferiores em comparação com o Flutter. Contando com uma grande comunidade ativa e uma variedade de conteúdos disponíveis que contribuem significativamente para o crescimento da tecnologia.

Com base na experiência deste estudo, é aconselhável a utiliza-

ção do Flutter para o desenvolvimento de aplicações, principalmente pelo fato de ter obtido resultados melhores de desempenho e apresentar uma documentação sólida, além de uma comunidade ativa em constante crescimento e uma boa curva de aprendizado. No entanto, não se deve descartar o uso do React Native, que, mesmo apresentando resultados inferiores em termos de desempenho e documentação desorganizada, é uma tecnologia mais consolidada no mercado, com uma comunidade ativa e diversidade de conteúdos disponíveis.

Portanto, a escolha entre essas duas ferramentas é relativa e depende do propósito e objetivo de cada projeto. Ambos os frameworks são capazes de produzir resultados de alta qualidade, e a decisão principal deve levar em consideração os requisitos específicos e o contexto de aplicação.

Como possíveis trabalhos futuros, pode-se conduzir testes com aplicações de maior complexidade, que exijam um desempenho mais rigoroso de ambas as tecnologias, e compará-las com aplicações nativas. Além disso, uma linha de estudo a ser considerada seria a realização de testes voltados à plataforma iOS.

REFERÊNCIAS

FLUTTER. *Flutter*. [S.l.], 2023. Disponível em: <https://github.com/flutter/flutter>.

FLUTTER. *Flutter documentation*. 2023. Disponível em: <https://docs.flutter.dev/>.

GITHUB. *Facebook React Native*. [S.l.], 2023. Disponível em: <https://github.com/facebook/react-native>.

LIMA, F. *Avaliação de Frameworks para o Desenvolvimento de Aplicações Híbridas*. [S.l.], 2019.

MOTA, D. *Estudo comparativo de frameworks multiplataforma de desenvolvimento de aplicações móveis*. Dissertação (Mestrado em Engenharia Informática – Computação Móvel) — Escola Superior de Tecnologia e Gestão, 2020.

PAIM, G.; JúNIOR, I. *Comparativo De Performance Entre Aplicativos Móveis Nativos E Multiplataforma*. Dissertação — Universidade Católica do Salvador, 2022.

PEREZ, S. *Mobile users are now spending 4-5 hours per day in apps*. [S.l.], 2022. Disponível em: <https://techcrunch.com/2022/08/03/mobile-users-now-spend-4-5-hours-per-day-in-apps-report-says/>.

PREZOTTO, E. *Estudo de Frameworks Multiplataforma Para Desenvolvimento de Aplicações Mobile Híbridas*. [S.l.], 2016.

REACTNATIVE. *ReactNative*. 2023. Disponível em: <https://reactnative.dev>.

SILVA, A. *ANÁLISE COMPARATIVA ENTRE OS FRAMEWORKS DE DESENVOLVIMENTO DE APLICATIVOS MÓVEIS MULTIPLATAFORMA*. Dissertação (BACHARELADO EM SISTEMAS DE INFORMAÇÃO) — UNIVERSIDADE FEDERAL DO CEARÁ CAMPUS QUIXADÁ, 2021.

SMUTNý, P. *Mobile development tools and cross-platform solutions*: In proceedings of the 13th international carpathian control conference (iccc). [S.l.], 2012.

STATCOUNTER. *Mobile Operating System Market Share Worldwide*. [S.l.], 2023. Disponível em: <https://gs.statcounter.com/os-market-share/mobile/worldwide/>.

STATISTA. *Cross-platform mobile frameworks used by software developers worldwide from 2019 to 2021*. [S.l.], 2021. Disponível em: <https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/>.

TURNER, A. *How many smartphones are in the world?* [S.l.], 2023. Disponível em: <https://gs.statcounter.com/os-market-share/mobile/worldwide/>.