UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

Isadora Leticia Tavares da Silva

Análise de Desempenho do Expo em Aplicações Móveis desenvolvidas com React Native

Recife

2025

Isadora Leticia Tavares da Silva

Análise de Desempenho do Expo em Aplicações Móveis desenvolvidas com React Native

Monografia apresentada ao Curso de Bacharelado em Sistemas de Informação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Universidade Federal Rural de Pernambuco – UFRPE

Departamento de Estatística e Informática

Curso de Bacharelado em Sistemas de Informação

Orientador: Cleviton Monteiro

Recife

2025

À Isadora criança que achava que isso não era possível.

Agradecimentos

Concluir este trabalho e esta etapa tão representativo e importante da minha vida não seria possível sem o apoio, a presença e o incentivo de pessoas especiais que estiveram ao meu lado nessa jornada. Por isso, gostaria de expressar minha profunda gratidão a todos que, de alguma forma, contribuíram para o desenvolvimento deste trabalho e para a realização de concluir o curso.

Em primeiro lugar, agradeço a Deus, pela força, coragem e sabedoria concedidas em todos os momentos. Pela fé que me guiou nos dias difíceis, pelas oportunidades que surgiram quando mais precisei e pela constante presença que me sustentou até aqui. Sem Ele, nada disso teria sido possível.

Meu agradecimento mais especial vai ao meu companheiro de vida, meu melhor amigo, Erick Barbosa. Sua presença, seu apoio e seu amor foram fundamentais em cada etapa desse processo. Obrigada por estar ao meu lado em todos os momentos, por me escutar, me motivar e acreditar em mim mesmo quando eu duvidava. Você é uma pessoa única e essencial na minha vida, e sem a sua companhia, esse caminho teria sido muito mais difícil. Saiba que você foi uma parte essencial dessa conquista, e sou eternamente grata por tudo que vivemos juntos até aqui. Eu amo você.

Aos meus grandes amigos Júlia de Melo, Carlos Henrique e Everton Matheus, minha gratidão. Ter vocês por perto tornou toda essa trajetória muito mais leve e especial. Obrigada pelas conversas, pelas risadas, pelo companheirismo e pela ajuda em tantos momentos. Nossa amizade é um presente que a universidade me deu, e levo cada um de vocês comigo no coração. Amo vocês e me sinto verdadeiramente abençoada por ter cruzado o caminho de pessoas tão incríveis.

Também deixo meu carinho e minha gratidão aos meus amigos Eloi Lima e Layza Lima. Vocês sempre estiveram ao meu lado, me apoiando, me ouvindo e me incentivando a seguir em frente. Suas palavras de conforto e seus gestos de carinho foram importantes. Obrigada por fazerem parte da minha vida e por contribuírem tanto para que eu chegasse até aqui. Amo muito vocês, e serei eternamente grata por tudo.

À minha família, que sempre me ofereceu apoio, amor e a base sólida necessária para que eu pudesse trilhar esse caminho, deixo minha mais sincera gratidão. Obrigada por estarem comigo, por acreditarem em mim e por me apoiarem em cada etapa dessa caminhada. Amo vocês profundamente.

Ao meu orientador, professor Cleviton Monteiro, deixo um agradecimento especial. Sua orientação dedicada, sua paciência e sua parceria ao longo de todo o de-

senvolvimento deste trabalho foram fundamentais para que eu conseguisse chegar a um resultado que me traz orgulho. Obrigada por acreditar no meu potencial, por me guiar com tanto zelo e por compartilhar seu conhecimento comigo. Tive a sorte de contar com um orientador comprometido, generoso e atencioso. Sua contribuição foi indispensável, e jamais esquecerei tudo que aprendi ao seu lado.

A todos que fizeram parte dessa caminhada, meu muito obrigada. Essa conquista é também de vocês.

Resumo

Este estudo compara o desempenho de aplicações móveis desenvolvidas em React Native CLI e Expo, analisando métricas críticas como tempo de resposta, uso de CPU e consumo de memória. Foram implementados dois aplicativos idênticos—um com cada abordagem—submetidos a testes controlados envolvendo geolocalização (Google Maps API), acesso à câmera/galeria e renderização dinâmica de mapas. Utilizou-se o Android Debug Bridge (ADB) para coleta automatizada de dados, processados estatisticamente no Google Colab (média, desvio padrão e teste t de Welch). Os experimentos, realizados em um Motorola Moto G52 (Android 13), demonstraram diferenças significativas ($\alpha = 5\%$) no desempenho entre as plataformas. O código e dados estão disponíveis publicamente (GitHub), garantindo reprodutibilidade.

Abstract

This study compares the performance of mobile applications developed in React Native CLI and Expo, analyzing critical metrics such as response time, CPU usage, and memory consumption. Two identical applications were implemented—one with each approach—and subjected to controlled tests involving geolocation (Google Maps API), camera/gallery access, and dynamic map rendering. Android Debug Bridge (ADB) was used for automated data collection, which was statistically processed in Google Colab (mean, standard deviation, and Welch's t-test). The experiments, conducted on a Motorola Moto G52 (Android 13), demonstrated significant differences ($\alpha = 5\%$) in performance between the platforms. The code and data are publicly available (GitHub), ensuring reproducibility.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

JS	JavaScript
RN	React Native
CLI	Command Line Interface
ADB	Android Debug Bridge

Sumário

1 Introdução

Segundo dados atualizados de 2024, divulgados pela GSMA Intelligence, observa-se um aumento significativo no uso de dispositivos móveis em todo o mundo. Atualmente, cerca de 69,4% da população mundial utiliza esses dispositivos, o que evidencia a dimensão e relevância dessa tecnologia no cotidiano das pessoas. Desde o início de 2023, houve um acréscimo de aproximadamente 138 milhões de usuários, indicando uma tendência contínua de crescimento anual (GSMA Intelligence, 2024).

O tempo gasto pelo usuário padrão atingiu 5 horas diárias, demonstrando assim que a maioria dos usuários desses dispositivos passa boa parte de seus dias utilizando-os. Houve também uma quebra de recorde no número de downloads de aplicativos, com 255 bilhões de downloads totais, apresentando uma média de 485 mil aplicativos baixados por minuto por consumidores globais.

De acordo com a pesquisa anual do Stack Overflow em 2023, o React Native ocupa a décima segunda posição entre os frameworks mais utilizados do ano, desconsiderando os frameworks web. Ele aparece como uma das poucas tecnologias voltadas ao desenvolvimento mobile multiplataforma, juntamente com o Flutter (Stack Overflow, 2023).

Por sua vez, o *Expo* (PagePro, 2025) reúne um conjunto de ferramentas e serviços integrados ao *React Native*, que está em crescimento contínuo em paralelo ao desenvolvimento mobile. Recentemente, o *Expo* foi incorporado à documentação oficial do *React Native* como uma alternativa de desenvolvimento recomendada. Registrou uma média de mais de 500 mil downloads semanais em 2024 (NPM, 2025). Esse aumento é significativo, representando um acréscimo de cerca de 100 mil downloads em comparação com fevereiro de 2023.

Esses dados refletem o aumento da adoção e do reconhecimento do Expo entre os desenvolvedores, consolidando sua posição como uma ferramenta relevante no desenvolvimento de aplicativos móveis com React Native. No entanto, à medida que mais ferramentas e bibliotecas são incorporadas ao processo de desenvolvimento, torna-se necessário analisar os impactos no desempenho do aplicativo.

Segundo a Scientia Mobile (2023), grande parte dos dispositivos móveis utilizados atualmente apresenta configurações modestas de hardware, com limitações de processamento, memória e armazenamento. Diante desse cenário, é importante que os aplicativos sejam desenvolvidos de forma otimizada, visando reduzir o consumo de recursos e garantir uma boa experiência de uso, especialmente em aparelhos de entrada (Scientia Mobile, 2023).

Nesse sentido, é necessário avaliar se o uso do Expo, ainda que proporcione diversas facilidades ao desenvolvimento, pode ocasionar uma sobrecarga de recursos que comprometa o desempenho da aplicação. A análise desse equilíbrio entre produtividade e eficiência é fundamental para garantir que o produto final esteja alinhado às necessidades dos usuários.

1.1 Objetivo Geral

A pesquisa tem como objetivo principal desenvolver e comparar o desempenho de duas aplicações móveis criadas com React Native: uma desenvolvida utilizando o ambiente de desenvolvimento padrão do React Native CLI e a outra construída com a plataforma Expo. Ambas as aplicações serão desenvolvidas com o mesmo código-fonte em React Native, diferenciando-se apenas nas configurações e no ambiente de desenvolvimento utilizado.

O estudo busca analisar as diferenças de desempenho e eficiência entre essas duas abordagens, destacando como as escolhas da plataforma de desenvolvimento podem influenciar a qualidade e a sustentabilidade das aplicações móveis resultantes.

1.2 Objetivos Específicos

1. Implementar dois aplicativos idênticos com React Native, diferenciando-os pelo ambiente de desenvolvimento: um será desenvolvido com o Expo e o outro com o React Native CLI.
2. Implementar e comparar funcionalidades em ambas as aplicações, abrangendo tanto o uso de recursos nativos do dispositivo quanto funcionalidades comuns no desenvolvimento com React Native, como navegação entre telas, requisições à API, gerenciamento de estados, entre outras.

2 Referencial Teórico

2.1 Desempenho de software

O desempenho de um aplicativo possui impacto direto na experiência do usuário. É crucial garantir que o sistema responda de forma rápida, permitindo que o usuário execute suas ações de maneira fluida e sem interrupções. Dessa forma, a preocupação em desenvolver um sistema eficiente, livre de gargalos e atrasos, torna-

O desempenho é uma característica amplamente presente em sistemas de software, sendo influenciado por diversos fatores, desde o próprio software até todas as camadas tecnológicas subjacentes, como o sistema operacional, o middleware, o hardware e as redes de comunicação (WOODSIDE; FRANKS; PETRIU, 2007).

De acordo com a norma ISO/IEC 25010, existem diversos pilares que asseguram a qualidade de um produto de software. Entre essas características, destaca-se o *Desempenho de Eficiência* (*Efficiency Performance*), que foca especificamente no desempenho do sistema, abrangendo aspectos como o tempo de resposta, utilização de recursos e capacidade do software em manter um bom funcionamento sob determinadas condições (International Organization for Standardization, 2011).

Figura 1 – Modelo de qualidade da norma ISO/IEC 25010:2011.

A Figura2 apresenta o modelo de qualidade da norma ISO/IEC 25010:2011, que descreve um conjunto de características para avaliar a qualidade de produtos de software. Entre essas características está o desempenho, o qual está diretamente relacionado à eficiência do sistema em ambientes reais de uso, especialmente quando executado em dispositivos com recursos limitados.

A definição de cada característica é a seguinte:

- **Comportamento temporal**: Grau em que os tempos de resposta e processamento e as taxas de rendimento de um produto ou sistema, ao executar suas funções, atendem aos requisitos.

- **Utilização de recursos**: Grau em que as quantidades e tipos de recursos utilizados por um produto ou sistema, no desempenho de suas funções, atendem aos requisitos.
- **Capacidade**: Grau em que os limites máximos de um parâmetro de produto ou sistema atendem aos requisitos.

O teste de desempenho, conforme descrito na norma ISO/IEC/IEEE 29119-1, avalia o desempenho de um item de teste em relação aos limites de tempo, uso de recursos (CPU, memória, disco, rede) e capacidade, mensurando sua capacidade de executar suas funções designadas (International Organization for Standardization, 2013).

Neste projeto, propomos a realização e aplicação de uma variedade de processos de testes de desempenho, empregando duas soluções distintas e explorando diversas ferramentas. O objetivo é conduzir uma análise abrangente do uso de recursos essenciais, como CPU, memória e tempo, visando a compreensão e otimização do desempenho dos sistemas em questão.

2.2 React Native e Expo

O React Native é uma biblioteca de código aberto voltada para o desenvolvimento de aplicativos móveis multiplataforma, compatível com os sistemas iOS e Android, que utiliza a sintaxe do React para a construção de interfaces. Lançado pelo Facebook em 2015, o projeto tem sido mantido pela empresa desde então. Em 2018, o framework alcançou o segundo maior número de contribuidores entre todos os repositórios do GitHub, evidenciando sua popularidade e o forte apoio da comunidade (Meta, 2015).

Tanto no desenvolvimento Android quanto no iOS, as visualizações são os elementos fundamentais que compõem a interface do usuário. Para Android, geralmente usa-se o Kotlin ou Java, enquanto para iOS, o Swift ou Objective-C, como ferramenta para criação de visualizações e apps.

O React Native une a estrutura e sintaxe declarativa do React com as funcionalidades e componentes nativos do Android e iOS. Durante tempo de execução gera automaticamente as visualizações correspondentes para cada sistema, adaptando-as conforme necessários mas apresentando um resultado idêntico. A biblioteca possui uma lista de componentes padrão que são chamados de componentes principais do React Native.

A arquitetura do React Native é chamada de Fabric e ela foi é relativamente nova, já que foi introduzida a partir da versão 0.69 lançada em 2022. Seu renderizador

passa por uma série de etapas para chegar na renderização em uma plataforma nativa. Para a primeira renderização o app passa por três fases: Render, Commit e Mount.

Já o Expo é uma plataforma que foi construída em torno do desenvolvimento de aplicativos móveis com React Native, que oferece aos desenvolvedores um leque de ferramentas para simplificar o processo de desenvolvimento e distribuição de aplicativos Android e iOS. Você só precisa lidar com JavaScript, sem contato com nativo. Não sendo necessário utilizar ferramentas como Android Studio ou Xcode; apenas com o aplicativo Expo Go, você pode simular e desenvolver seu aplicativo no dispositivo.

Com acesso a APIs nativas, componentes prontos e uma interface de linha de comando (CLI), o Expo agiliza o fluxo de trabalho, enquanto seus serviços de construção e hospedagem facilitam a distribuição de aplicativos em diferentes plataformas. Dentre os seus serviços oferece um ambiente de construção gerenciado e ferramentas para testes e depuração. Isso inclui o Expo SDK, a API de Módulos, o aplicativo Go, a CLI, um sistema para roteamento, documentação e várias outras ferramentas de suporte.

3 Trabalhos Relacionados

Esta seção apresenta e analisa artigos relacionados ao tema de desenvolvimento mobile e análise comparativa de performance , cujas metodologias utilizadas foram inspiradas para este trabalho.

Ezequias Rocha (2023) realizou um estudo com o objetivo de analisar a aplicação de técnicas voltadas à melhoria de desempenho em sistemas front-end. O trabalho teve como foco reunir práticas utilizadas no desenvolvimento web, especialmente aquelas que pudessem contribuir para a performance de aplicações desenvolvidas com React. A metodologia adotada incluiu revisão de literatura e coleta de informações em comunidades de desenvolvedores, como o Reddit, que foram posteriormente aplicadas em um sistema com grande volume de dados e múltiplas requisições. Diversas métricas de avaliação foram utilizadas, como o *Tempo Total de Bloqueio* — que indica o quanto o JavaScript impede o processamento principal da página — e o *Índice de Velocidade*, que mede o tempo para o carregamento visual completo. Os resultados demonstraram que algumas das técnicas aplicadas contribuíram significativamente para a melhora do desempenho da aplicação, reforçando a importância da análise contínua de performance (ROCHA, 2023).

Orestes (2023) desenvolveu um trabalho comparativo entre os frameworks Flutter e React Native, com foco em três aspectos principais: curva de aprendizado, experiência de desenvolvimento e desempenho das aplicações. O estudo foi motivado pelo interesse em entender o comportamento de cada tecnologia desde o início do projeto até sua execução final. Para isso, o autor desenvolveu duas versões de uma mesma aplicação — uma calculadora — utilizando cada um dos frameworks. A comparação se baseou tanto no processo de criação quanto no desempenho dos apps resultantes. Como resultado, o Flutter apresentou melhor desempenho nos testes realizados, enquanto o React Native se destacou por ter uma comunidade mais consolidada e maior suporte. O trabalho contribui com uma visão prática e técnica sobre as vantagens e limitações de cada ferramenta, sendo relevante para desenvolvedores que precisam escolher entre essas soluções para projetos cross-platform (FABRIS, 2023).

Breno de Freitas (2022), no trabalho ”Flutter e React Native: uma análise comparativa entre dois frameworks de desenvolvimento mobile multiplataforma”, desenvolveu cinco aplicações idênticas em Flutter e React Native para avaliar o desempenho em diferentes cenários: um contador que incrementa a cada 16ms; seis contadores simultâneos, sendo metade atualizada alternadamente; um contador manual ativado por botão; uma navegação entre duas telas; e uma lista com mil itens gerados. Para

garantir a precisão dos testes, todas as interações (toques e deslizamentos) foram simuladas via comandos ADB, com os apps executados 50 vezes em um Xiaomi Mi A1 (4 GB de RAM, Snapdragon 625), em modo avião e com notificações desativadas. A automação foi feita com um script Python que lia configurações em JSON e armazenava os resultados, e a coleta de dados de desempenho foi feita por ferramentas nativas: ADB para React Native e Dart DevTools para Flutter. A métrica principal foi o percentual de janky frames, e os resultados mostraram que o Flutter teve melhor desempenho em listas e navegação, enquanto o React Native se destacou em tarefas mais simples, como os contadores, reforçando a importância de considerar o tipo de aplicação ao escolher o framework. (FREITAS, 2022)

Outro estudo relevante é o artigo *Comparison of React Native and Expo*, que teve como objetivo avaliar a experiência de desenvolvimento e as limitações de ambos os frameworks no contexto de aplicações móveis multiplataforma no ambiente React Native. A abordagem metodológica consistiu na implementação de funcionalidades amplamente utilizadas em aplicativos reais, como *splash screen*, localização, permissões e ícones personalizados. Os critérios de avaliação incluíram o número de arquivos modificados, quantidade de linhas de código necessárias para implementar cada funcionalidade e o nível de dificuldade percebido. O estudo revelou que, embora o React Native exija um maior número de arquivos e linhas de código para determinadas tarefas, o Expo oferece maior praticidade e velocidade na implementação de funcionalidades padrão. A pesquisa destacou ainda que o Expo tende a ser mais acessível para desenvolvedores iniciantes e mais produtivo em projetos com requisitos comuns, enquanto o React Native oferece maior flexibilidade para customizações avançadas. Como limitação, os autores apontam que os testes foram realizados apenas em dispositivos Android e que o foco esteve na experiência de desenvolvimento, sem aprofundamento em métricas de desempenho. A lista de funcionalidades utilizada neste estudo inspirou a seleção de critérios adotados neste trabalho, que também compara o React Native e o Expo em termos de performance e facilidade de uso (HUTRI, 2023).

4 Metodologia

Esta seção descreve os procedimentos adotados para conduzir o estudo comparativo entre os frameworks analisados, com foco na avaliação de desempenho e experiência de desenvolvimento.

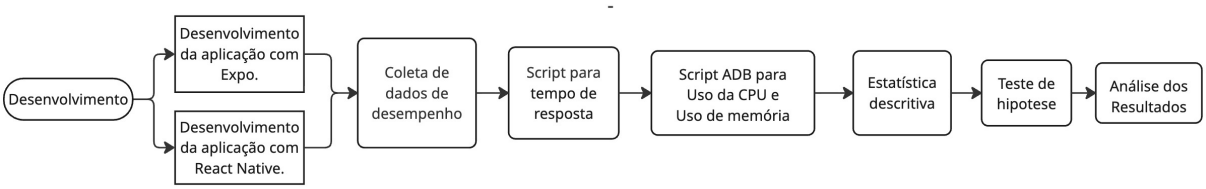

Figura 2 – Etapas do desenvolvimento e métodos aplicados na condução da pesquisa.

A Figura 2 mostra o fluxo adotado para a realização dos experimentos desta pesquisa. O processo inicia-se com o desenvolvimento de dois aplicativos com funcionalidades equivalentes: um utilizando o framework Expo e outro com o React Native puro. Em seguida, à coleta de dados de desempenho. Após a coleta, os dados são processados por meio de diferentes scripts: um script específico é utilizado para medir o tempo de resposta, enquanto outro, por meio do ADB (Android Debug Bridge), é responsável por monitorar o uso da CPU e da memória. Os dados obtidos são posteriormente analisados por meio de estatística descritiva, com cálculos estatísticos, como média e desvio padrão, seguidos de testes de hipótese, com o objetivo de verificar a significância dos resultados. Por fim, realiza-se a análise final dos dados, analisando as conclusões sobre o desempenho das abordagens comparadas.

4.1 Funcionalidades

A escolha das funcionalidades foram implementadas para servirem de base para a comparação entre as tecnologias. Além disso, foram utilizados recursos que são bastante utilizados na maioria do desenvolvimento de aplicativos. Nesta seção, serão apresentados os principais recursos abordados nesta pesquisa.

Uma das funcionalidades implementadas no aplicativo foi um mapa interativo, projetado que inicia exibindo a localização atual do usuário após a solicitação e concessão da permissão de geolocalização. Além disso, a funcionalidade permite a adição de pontos personalizados diretamente no mapa.

Para a implementação dos experimentos, foi utilizada a API do Google Maps, tanto na renderização da interface quanto na busca e exibição de localizações. A escolha dessa API se justifica por sua ampla adoção no desenvolvimento mobile e pelo

alto nível de complexidade que impõe às aplicações, especialmente em termos de renderização gráfica e consumo de recursos. Segundo a Acropolium (Acropolium, 2023), o Google Maps detém uma participação de mercado de 61,3% e é integrado a mais de um milhão de aplicativos em diferentes setores, o que reforça sua representatividade como ferramenta de geolocalização. O uso de mapas, portanto, representa um cenário realista e desafiador, exigindo manipulação gráfica intensiva, acesso constante à rede e atualizações dinâmicas da interface.

Estudos como o de Aubergine (2023) demonstram que, ao realizar ações comuns em mapas, como o gesto de ”fling”, há diferenças significativas no desempenho entre frameworks — com o Flutter, por exemplo, apresentando menor tempo de renderização e construção de frames em comparação ao Android nativo. Esses dados reforçam a relevância do uso do Google Maps como métrica concreta para avaliação da eficiência das aplicações móveis em diferentes tecnologias.(AUBERGINE, 2023)

O uso da câmera e da galeria de imagens exige acesso a recursos nativos do dispositivo, além de permissões específicas para seu correto funcionamento. Como cada framework adota abordagens distintas para lidar com essas funcionalidades, optou-se por incluir uma análise dedicada, com o objetivo de identificar possíveis impactos no desempenho das aplicações.

No experimento, a funcionalidade foi implementada por meio de um componente denominado *Pin*, associado a cada ponto adicionado ao mapa. Esse componente permite ao usuário vincular uma imagem ao local selecionado, oferecendo duas opções: capturar uma foto utilizando a câmera do dispositivo ou selecionar uma imagem diretamente da galeria.

Portanto, foram implementadas funcionalidades — como o mapa interativo com pontos personalizados e a integração com câmera e galeria — que não apenas refletem casos de uso recorrentes em aplicações reais, mas também impõem desafios técnicos relevantes. A utilização desses recursos, que demandam acesso a APIs nativas, manipulação gráfica e controle de permissões, permite observar como cada tecnologia se comporta em cenários comuns do desenvolvimento mobile.

4.2 Métricas

4.2.1 Tempo de resposta

O tempo de resposta é uma métrica fundamental para avaliar a experiência do usuário ao utilizar o aplicativo. Além disso, mensurar esse tempo permite identificar possíveis gargalos no processamento das solicitações de permissão, seja na camada do sistema operacional, seja na forma como as ferramentas gerenciam essas requisi-

ções.

Com base em estudos relacionados que utilizam o Android Debug Bridge (ADB) para análises de desempenho, foi desenvolvido um script que automatiza o processo de fechamento e reabertura do aplicativo. Em cada aplicativo, foi implementada uma função que aguarda a resposta do sistema para as permissões de localização, câmera e galeria, registrando o tempo decorrido em um arquivo de texto.

O Android Debug Bridge (ADB) revelou-se uma ferramenta essencial nesse processo, pois possibilita a execução controlada e repetível dos testes, minimizando possíveis interferências externas e garantindo que os resultados obtidos não sejam enviesados. O script foi executado 30 vezes, totalizando 30 amostras do tempo necessário para a concessão dessas permissões.

4.2.2 Utilização da CPU e memória

Para a ánalise das métricas de desempenho do uso da CPU e o consumo de memória, foi necessário obter uma análise mais precisa e realista do consumo de recursos pelos aplicativos, essa questão motivou a reformulação do script de testes. Os experimentos iniciais não impunham carga suficiente ao sistema, o que limitava a observação de impactos significativos no desempenho. Para contornar isso, foi desenvolvido um novo script em React, projetado para simular interações intensas — como múltiplas requisições à API do Google Maps e re-renderizações sucessivas do mapa — com o objetivo de estressar o sistema e capturar métricas mais representativas de uso de CPU e memória em cenários próximos ao uso real.

O script utiliza o `useEffect`, acionado automaticamente na abertura do aplicativo, para adicionar dez pontos no mapa, realizando dez chamadas à API e forçando a atualização da interface. No início e no fim da execução, os timestamps são registrados em um arquivo de texto para delimitar com precisão o intervalo de análise. O Android Debug Bridge (ADB) foi empregado para automatizar a abertura e o encerramento do aplicativo, além de coletar dados de uso da CPU e memória durante esse período, também com marcações temporais.

Os dados coletados foram processados e analisados no Google Colab, um ambiente em nuvem amplamente utilizado em pesquisas por oferecer suporte a bibliotecas científicas e de machine learning, facilitando a visualização e interpretação dos resultados (BISONG, 2019).

Por fim, no Google Colab, os dados foram filtrados com base nos timestamps registrados durante a execução do script, garantindo que apenas as informações coletadas nesse intervalo fossem consideradas. Em seguida, foi calculada a média das amostras para possibilitar uma análise mais precisa e consistente.

4.2.3 Média e desvio padrão

Para a análise estatística dos dados coletados, foram aplicados cálculos de média e desvio padrão em cada um dos experimentos. No script que mede o tempo de resposta para obtenção das permissões, foram realizadas 30 execuções, e, para cada permissão, foi calculada a média e o desvio padrão dos tempos registrados.

No script que analisa o uso da CPU e memória, foram realizadas 30 coletas de dados. Em cada coleta, foram registrados valores de uso da CPU e memória a cada 1 milissegundo, durante a execução do script que adiciona dez pontos no mapa. Para cada coleta, foi calculada a curva de uso dos recursos (CPU e memória), que representa o consumo total de recursos durante o período analisado.

Ao final das 30 execuções, foi calculada a média das médias e o desvio padrão para cada um dos aplicativos (com e sem o Expo). Essa metodologia permitiu uma comparação quantitativa e robusta do consumo de recursos entre as duas versões, destacando possíveis diferenças de eficiência e estabilidade durante a execução de tarefas intensivas.

4.2.4 Teste de hipótese

Para validar estatisticamente se houve diferença significativa no desempenho entre os aplicativos desenvolvidos com e sem o Expo, foi aplicado um teste de hipótese bilateral. O objetivo foi verificar se a média dos valores coletados (como uso da CPU ou tempo de resposta) difere entre as duas tecnologias.

As hipóteses consideradas foram:

- **Hipótese nula (H0):** as médias de desempenho dos dois aplicativos são iguais, ou seja, $\mu_{\text{Expo}} = \mu_{\text{React Native}}$.
- **Hipótese alternativa (H1):** as médias de desempenho dos dois aplicativos são diferentes, ou seja, $\mu_{\text{Expo}} \neq \mu_{\text{React Native}}$.

Trata-se de um teste bilateral, pois não se assume previamente que uma tecnologia seja superior à outra, apenas se busca identificar qualquer diferença significativa entre elas.

Para cada permissão (localização, câmera e galeria), foram coletados 30 valores amostrais de tempo de resposta, agrupados em dois conjuntos: um referente ao aplicativo com Expo (`expo_x`) e outro ao aplicativo com React Native puro (`rn_x`), onde *x* representa a permissão avaliada.

A análise estatística dos dados coletados foi realizada utilizando o teste *t* de Student com correção de Welch, por meio da biblioteca SciPy da linguagem Python. Esse

teste é apropriado para comparar as médias de dois grupos com possíveis variâncias diferentes (STUDENT, 1908; WELCH, 1947; VIRTANEN et al., 2020).

O procedimento forneceu, para cada comparação, um valor estatístico e um valor-*p*, que foi confrontado com um nível de significância de 5%. Quando o valor-*p* foi inferior a esse limite, considerou-se que existia uma diferença estatisticamente significativa entre os desempenhos das duas versões da aplicação.

4.3 Ferramentas

Neste projeto, foram utilizadas duas ferramentas principais para coleta e análise de dados de desempenho: o Android Debug Bridge (ADB) e a biblioteca react-native-performance.

O Android Debug Bridge (ADB) é uma ferramenta de linha de comando versátil que permite a comunicação entre um computador e um dispositivo Android. Parte integrante do Android SDK (Software Development Kit), é amplamente utilizada por desenvolvedores para depuração, testes e análise de aplicativos Android. No contexto deste trabalho, o ADB foi fundamental para automatizar a coleta de dados. Por meio dele, foi possível controlar o ciclo de vida do aplicativo — abrindo e fechando o app de forma programática — e coletar métricas detalhadas, como uso de CPU e memória em tempo real.

Complementando essa abordagem, a biblioteca react-native-performance foi utilizada para monitorar métricas internas da aplicação React Native. Essa ferramenta fornece informações sobre o tempo de renderização de componentes, o carregamento de telas e o uso de recursos pelo aplicativo. Sua aplicação foi essencial para compreender o comportamento da interface e dos fluxos de navegação em tempo real, diretamente no dispositivo.

Todo o desenvolvimento dos aplicativos, scripts de coleta e dados brutos foram versionados em um repositório público no GitHub (disponível em: <https://github.com/isadoratavare/TCC>). Para análise estatística e processamento dos resultados, utilizou-se o ambiente Google Colab (disponível em: <https://colab.research.google.com/drive/1ZMbX6IRoPEMLJjYkLszBKIMX9XSYYe8k>), onde foram implementados os algoritmos para cálculo de métricas de desempenho e testes estatísticos.

4.4 Ambiente de testes

Os experimentos foram realizados em um Motorola Moto G52, com 4 GB de RAM e sistema operacional Android 13. Para garantir condições controladas, o dispositivo era reiniciado antes de cada coleta, e todos os aplicativos em segundo plano

eram encerrados manualmente. Um script ADB foi empregado para assegurar o encerramento completo do aplicativo testado (sem processos em background ou cache), além de verificar a ausência de interferências de outros serviços durante os testes.

5 Resultados

Nesta seção, serão apresentados os resultados obtidos a partir das análises de desempenho realizadas. As métricas avaliadas incluem tempo de resposta para busca de permissões, consumo de CPU e memória, tempo de execução de tarefas e tamanho do arquivo executável.

5.1 Tempo de Resposta na obtenção das permissões

Para avaliar o desempenho na obtenção das permissões no aplicativo, foram realizados testes para cada tipo de permissão: galeria, localização e câmera. O objetivo foi comparar o tempo médio necessário para a obtenção dessas permissões utilizando duas abordagens distintas de implementação.

Os resultados demonstram diferenças entre as duas abordagens. A seguir apresenta os valores médios, os desvios padrão e os testes de hipótese obtidos nos testes.

Tabela 1 – Estatísticas descritivas do tempo de resposta para acesso à Galeria

Métrica	**Expo**	**React Native**
Média (ms)	326,35	13,62
Desvio padrão (ms)	111,78	6,40
Estatística do teste *t*	*t* = 14.78	
Valor-*p*	*p* $8{,}38 \times 10^{-15}$	

O valor-p está muito abaixo do nível de significância estabelecido, levando à rejeição da hipótese nula. Isso indica que também existe diferença significativa entre os tempos médios de solicitação da permissão da galeria entre Expo e React Native.

Tabela 2 – Estatísticas descritivas do tempo de resposta para acesso à Localização

Métrica	**Expo**	**React Native**
Média (ms)	399,47	23,36
Desvio padrão (ms)	63,64	44,37
Estatística do teste *t*	*t* = 25.95	
Valor-*p*	*p* $2{,}10 \times 10^{-31}$	

Novamente, o valor-p é inferior a 0,05, o que leva à rejeição da hipótese nula. Assim, conclui-se que há diferença significativa entre os tempos médios para a permissão de localização entre as duas tecnologias.

Tabela 3 – Estatísticas descritivas do tempo de resposta para acesso à Câmera

Métrica	**Expo**	**React Native**
Média (ms)	332,03	20,63
Desvio padrão (ms)	113,08	42,85
Estatística do teste *t*	*t* = 13,87	
Valor-*p*	valor-*p* $2{,}89 \times 10^{-16}$	

Como o valor-p é inferior ao nível de significância adotado, rejeita-se a hipótese nula. Portanto, há evidência estatística de que existe uma diferença significativa entre os tempos médios na solicitação da permissão da câmera entre Expo e React Native.

Observa-se também que a versão com apenas React Native apresentou desempenho superior em todas as permissões testadas, com tempos de resposta consideravelmente menores. Além disso, essa abordagem também apresentou menor variabilidade nos resultados, mostrada pelos desvios padrão mais baixos, especialmente na permissão de galeria. Isso sugere que o uso de bibliotecas nativas diretas proporciona uma experiência mais rápida e previsível para o usuário no momento da solicitação de permissões.

5.2 Uso da CPU

Tabela 4 – Estatísticas descritivas do uso da CPU para Expo e React Native

Métrica	**Expo**	**React Native**
Média (%)	129,08	147,73
Mediana (%)	134,54	170,85
Desvio padrão (%)	24,36	48,99
Estatística do teste *t*	-1,84	
Valor-*p*	0,073	

A média de uso da CPU ao utilizar o Expo foi de aproximadamente 129,08%, com uma mediana de 134,53% e desvio padrão de 24,36%. Já no caso do React Native puro, a média de uso da CPU foi mais elevada, atingindo 147,73%, com uma mediana de 170,85% e desvio padrão de 48,99%, indicando uma maior variabilidade nos dados.

Para verificar se essa diferença era estatisticamente significativa, foi realizado um teste *t* de Welch, que resultou em uma estatística de teste de aproximadamente -1,84 e um valor-*p* de 0,073. Como o valor-*p* é maior que o nível de significância usual de 0,05, não se pode rejeitar a hipótese nula. Portanto, conclui-se que não há evidências estatísticas suficientes para afirmar que existe uma diferença significativa no uso da CPU entre o Expo e o React Native.

5.3 Uso da memória

Tabela 5 – Estatísticas descritivas do uso de memória para Expo e React Native

Métrica	**Expo**	**React Native**
Média (MB)	6,72	6,46
Mediana (MB)	6,59	6,50
Desvio padrão (MB)	0,33	0,29
Estatística do teste *t*	-3,21	
Valor-*p*	0,0022	

Quanto ao uso de memória, os resultados mostraram um comportamento oposto. A média de uso da memória com Expo foi de 6,72 MB, com mediana de 6,59 MB e desvio padrão de 0,33. Por outro lado, o React Native apresentou média ligeiramente inferior, de 6,46 MB, com mediana de 6,50 MB e desvio padrão de 0,29.

O teste *t* de Welch realizado nesse caso apresentou uma estatística de teste de -3,21 e valor-*p* de aproximadamente 0,0022. Esse valor-*p* está abaixo do limite de significância de 0,05, permitindo rejeitar a hipótese nula. Assim, pode-se afirmar que há uma diferença estatisticamente significativa no uso de memória entre as duas tecnologias, sendo o Expo mais exigente nesse aspecto.

6 Conclusão

Este trabalho teve como principal objetivo investigar se a utilização do Expo no desenvolvimento de aplicações com React Native impacta significativamente o desempenho do aplicativo final em comparação com uma abordagem utilizando apenas o React Native puro. Para isso, foi desenvolvido um mesmo aplicativo com ambas as tecnologias, mantendo funcionalidades e fluxos idênticos, a fim de possibilitar uma análise comparativa justa e precisa. As métricas analisadas incluíram o tempo de resposta na solicitação de permissões e o consumo de CPU e memória durante a execução de um fluxo automatizado no aplicativo.

Os resultados obtidos apontam que a versão do aplicativo construída sem o Expo apresentou tempos de resposta mais baixos e mais consistentes ao lidar com permissões, evidenciando uma vantagem em termos de agilidade. Por outro lado, apenas no aspecto de uso de CPU, a versão com Expo demonstrou um desempenho mais estável, com menor variabilidade e tempo médio de execução inferior, o que pode indicar uma melhor gestão de recursos em determinados cenários, já a memória o expo continou sendo mais instável.

Apesar dos resultados significativos, o estudo enfrentou algumas limitações. A principal foi a impossibilidade de realizar testes em dispositivos iOS, restringindo a análise ao ambiente Android. Além disso, houve dificuldades em encontrar ferramentas específicas para mensuração de desempenho voltadas ao React Native, uma vez que a maioria das soluções disponíveis é orientada ao desenvolvimento nativo.

Como sugestão para trabalhos futuros, propõe-se a realização de análises semelhantes em dispositivos iOS, bem como a ampliação das métricas avaliadas — incluindo consumo de bateria, tempo de carregamento da aplicação e fluidez de animações. Investigar o impacto de bibliotecas de terceiros também pode trazer novos insights sobre o desempenho entre as abordagens com e sem Expo. Além disso, adicionar uma ánalise estátistica mais robusta através de intervalos de confiança nos testes de hipótese para casos com diferenças pequenas entre médias, cálculo do tamanho do efeito para avaliar a relevância prática das diferenças e testes de normalidade para validar os pressupostos dos testes paramétricos.

Dessa forma, conclui-se que a escolha entre utilizar ou não o Expo deve considerar o equilíbrio entre facilidade de desenvolvimento e requisitos de desempenho específicos da aplicação. O uso do Expo pode trazer benefícios em termos de produtividade e simplicidade, mas pode implicar em perdas de desempenho em cenários mais sensíveis, especialmente na manipulação de permissões nativas.

Referências

Acropolium. *Mapbox vs. Google Maps: Choosing a Map API*. 2023. Disponível em: <https://acropolium.com/blog/choosing-a-map-api-mapbox-vs-google-maps/>. Acesso em: 31 jul. 2025. Citado na página 19.

AUBERGINE. *Comparing the performance of an app built with Native Android and Flutter*. 2023. Disponível em: <https://www.aubergine.co/insights/comparing-the-performance-of-an-app-built-with-native-android-and-flutter>. Acesso em: 31 jul. 2025. Citado na página 19.

BISONG, E. Google colaboratory. In: *Building Machine Learning and Deep Learning Models on Google Cloud Platform*. Berkeley, CA: Apress, 2019. p. 59–64. Citado na página 20.

FABRIS, L. O. *Análise do desempenho e desenvolvimento de aplicações mobiles, utilizando o framework Flutter e a Biblioteca React Native*. 2023. <http://repositorio.unesc.net/handle/1/10969>. Trabalho de Conclusão de Curso – Universidade do Extremo Sul Catarinense (UNESC). Citado na página 16.

FREITAS, B. C. de. *Flutter e React Native: uma análise comparativa entre dois frameworks de desenvolvimento mobile multiplataforma*. 2022. <https://pantheon.ufrj.br/bitstream/11422/19671/1/BCFreitas.pdf>. Trabalho de Conclusão de Curso – Universidade Federal do Rio de Janeiro. Citado na página 17.

GSMA Intelligence. *Digital 2024: Global Overview Report*. 2024. <https://datareportal.com/reports/digital-2024-global-overview-report>. WE ARE SOCIAL; MELTWATER. Acesso em: jul. 2025. Citado na página 11.

HUTRI, H. *Comparison of React Native and Expo*. Dissertação (Master's Thesis) — Lappeenranta–Lahti University of Technology LUT, 2023. Master's Programme in Software Engineering and Digital Transformation. Disponível em: <https://lutpub.lut.fi/bitstream/handle/10024/165256/diplomityo_hutri_hugo.pdf?sequence=1&isAllowed=y>. Citado na página 17.

International Organization for Standardization. *ISO/IEC 25010:2011 – Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*. 2011. Citado na página 13.

International Organization for Standardization. *ISO/IEC/IEEE 29119-1:2013 – Software and systems engineering – Software testing – Part 1: Concepts and definitions*. Geneva: ISO, 2013. Citado na página 14.

Meta. *React Native*. 2015. <https://reactnative.dev/>. Lançado em 2015 como projeto de código aberto e mantido desde então. Em 2018, registrou o segundo maior número de contribuidores de qualquer repositório no GitHub. Acesso em: jul. 2025. Citado na página 14.

NPM. *Expo Weekly Downloads Statistics - 2024*. 2025. <https://www.npmjs.com/package/expo>. Citado na página 11.

PagePro. *Why Expo is Becoming a Go-To Tool for React Native Developers*. 2025. <https://pagepro.co/blog/why-expo-is-becoming-a-go-to-tool/>. Citado na página 11.

ROCHA, E. d. O. *Estudo e análise de técnicas para melhorar desempenho de sistemas front-end com React*. 2023. <https://dspace.sti.ufcg.edu.br/jspui/handle/riufcg/29299>. Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Software) – Universidade Federal de Campina Grande. Citado na página 16.

Scientia Mobile. *Mobile Overview Report (MOVR) - Q2 2023*. 2023. <https://www.scientiamobile.com/movr>. Reston, VA. Acesso em: jul. 2025. Citado na página 11.

Stack Overflow. *Stack Overflow Developer Survey 2023*. 2023. <https://survey.stackoverflow.co/2023/?utm_source=chatgpt.com>. Acesso em: jul. 2025. Citado na página 11.

STUDENT. The probable error of a mean. *Biometrika*, v. 6, n. 1, p. 1–25, 1908. Citado na página 22.

VIRTANEN, P. et al. Scipy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, v. 17, p. 261–272, 2020. Citado na página 22.

WELCH, B. L. The generalization of 'student's' problem when several different population variances are involved. *Biometrika*, v. 34, n. 1-2, p. 28–35, 1947. Citado na página 22.

WOODSIDE, C. M.; FRANKS, G.; PETRIU, D. C. The future of software performance engineering. In: *Proceedings of the Future of Software Engineering (FOSE'07)*. Minneapolis: [s.n.], 2007. p. 171–187. Acesso em: jul. 2025. Disponível em: <https://www.researchgate.net/profile/Dorina-Petriu/publication/4250870_The_Future_of_Software_Performance_Engineering/links/02bfe5107ed14f3a8f000000/The-Future-of-Software-Performance-Engineering.pdf>. Citado na página 13.