Universidade Federal de Pernambuco – UFPE

Centro de Informática

Graduação em Ciência da Computação

Comprehensive Repository Analysis of Mobile Projects Built with React Native

Pedro Sereno Galvão

Recife

2018

Pedro Sereno Galvão

Comprehensive Repository Analysis of Mobile Projects Built with React Native

Trabalho apresentado ao Programa de Graduação em Ciência da Computação do Departamento de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Universidade Federal de Pernambuco – UFPE

Centro de Informática

Graduação em Ciência da Computação

Orientador: Prof. Leopoldo Motta Teixeira

Recife

2018

A todos que desenvolvem aplicativos com React Native pelo mundo.

AGRADECIMENTOS

Agradeço primeiro a minha família, que sempre estiveram perto e me apoiando de todas as maneiras possíveis me dando as condições necessárias para estar realizando este trabalho e formaram a pessoa que eu sou. Sou imensamente grato por isso.

Agradeço também a meu orientador, o professor Leopoldo Teixeira, por ter me guiado durante a trajetória deste trabalho. Extremamente compreensível com a realização do trabalho a distância e sempre fornecendo norteamentos para ajudar na direção da pesquisa, contribuindo com meu crescimento.

Por fim, agradeço a todos que de alguma forma contribuíram com o desenvolvimento deste projeto. Em especial a meus amigos e minha namorada, que nos momentos de tensão me ajudaram a descontrair.

RESUMO

No desenvolvimento de aplicativos para dispositivos móveis existem várias escolhas de tecnologia para desenvolver uma aplicação para múltiplas plataformas minimizando a quantidade de código específico de plataforma e maximizando o reuso de código entre elas. Uma das tecnologias mais recentes com essa finalidade se chama React-Native, liberada como um projeto de código aberto em 2015 no Github e desenvolvida pelo Facebook. Ela cresceu em popularidade rapidamente pela sua funcionalidade de ter um alcance híbrido com a promessa de uma performance nativa, além de ter surgido com base no já popular React, uma solução para desenvolvimento de aplicações para a web também desenvolvida pelo Facebook. Este trabalho apresenta o resultado de uma mineração de dados em cima de projetos públicos no Github envolvendo React-Native em busca de entender como os projetos se comportaram desde sua concepção, para isso foram analisados 7138 repositórios e os resultados mostraram que a maioria dos projetos analisados usa uma versão defasada do React-Native.

Palavras-chave: React-Native. Mineração de Dados. Engenharia de Software.

ABSTRACT

In mobile application development there are a number of technology choices to develop a cross-platform application by minimizing the amount of platform-specific code and maximizing reuse of code between them. One of the most recent technologies for this purpose is called React-Native, released as an open source project in 2015 on Github and developed by Facebook. It has grown in popularity rapidly because of its functionality to have a hybrid range with the promise of a native performance, and has emerged based on the already popular React, a solution for web application development also developed by Facebook. This work presents the result of a data mining over public projects in Github involving React-Native in order to understand how the projects behaved from their conception, for that were analyzed 7138 repositories and the results showed that most of the analyzed projects uses a outdated version of React-Native.

Keywords: React-Native. Data Mining. Software Engineering.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

SDK	Software Development Kit
UWP	Universal Windows Platform
API	Application Programming Interface

SUMÁRIO

1 INTRODUÇÃO

Desde o lançamento dos smartphones, o número de usuários continua crescendo (POUSHTER; BISHOP; CHWE, 2018), o que os tornam uma plataforma crucial para negócios interagirem com seu público. Entretanto, hoje existem dois sistemas operacionais completamente distintos para desenvolver aplicações móveis, o que aumenta o custo para fazer o produto final, pois é necessário manter duas bases de código distintas.

Para simplificar o custo de desenvolvimento das aplicações de uma forma que alcance as duas plataformas, Android e iOS, surgiram tecnologias como React-Native (React Native Website, 2018). Com a promessa de entregar uma experiência nativa com um desenvolvimento em cima da linguagem de programação Javascript, mas ainda podendo ter acesso à interface de programação de aplicações (do inglês, *Application Programming Interface* - API) da plataforma em questão. Tornando quase que indistiguível para o usuário final que a aplicação não foi desenvolvida nativamente.

1.1 MOTIVAÇÃO

React-Native surgiu dois anos após React, uma biblioteca para desenvolvimentos de interfaces de usuário para web, seguindo o mesmo modelo e sendo utilizada também dentro do Facebook, podendo ser considerada uma tecnologia pronta para produção, estando presente em num dos aplicativos mais utilizados nos dispositivos.

Além disso, com React-Native é possível desenvolver aplicativos inteiros só utilizando Javascript, considerada a linguagem mais utilizada no Github, que possui um acervo enorme de projetos de código aberto para auxiliarem no desenvolvimento das aplicações.

Apesar de React-Native oferecer uma boa solução, ela não vem sem seus custos. Desde seu lançamento como projeto de código aberto em março de 2015, houve 57 versões lançadas, o que acarreta em mais de uma versão por mês em alguns casos. As mudanças várias vezes trouxeram incompatibilidades com versões anteriores, de modo inibir projetos a sequer atualizarem.

A motivação dessa pesquisa é entender como os projetos que utilizam React-Native se comportaram nesses anos em meio a esta situação de crescimento.

1.2 OBJETIVO

O presente trabalho tem como objetivo realizar uma análise de repositórios de React Native através de mineração de dados e encontrar pontos comuns de problemas nos

projetos que possam ter uma solução comum. Para alcançar este objetivo, as seguintes etapas são estabelecidas:

- Identificar projetos utilizando React-Native;
- Minerar os dados dos repositórios;
- Extrair métricas dos repositórios.
- Analisar as métricas gerais e tirar conclusões;

Algumas das métricas a serem extraídas incluem:

- Quantos repositórios são aplicativos ou bibliotecas;
- Quais versões da ferramenta estão em uso;
- Se o aplicativo foi desenvolvido utilizando Expo;
- Se o código é mantido em Javascript ou se utiliza Flow ou Typescript;
- Dada a documentação escassa da ferramenta, quais interfaces foram utilizadas;
- Quantidade de commits;

1.3 ORGANIZAÇÃO

A estrutura do trabalho divide-se em capítulos. O Capítulo 2 apresenta a fundamentação teórica para dar a base a respeito das tecnologias relacionadas no estudo. O Capítulo 3 explica como foi realizado o estudo e seu processo. O Capítulo 4 apresenta os resultados do estudo com sua análise. Por fim, o Capítulo 5 apresenta a conclusão do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

O propósito deste capítulo é fornecer uma fundamentação para a compreensão do React Native e tecnologias relacionadas a ele que foram o foco da pesquisa em questão.

2.1 REACT NATIVE

React Native é uma tecnologia que permite o desenvolvimento de aplicações hibrídas somente com Javascript tendo a opção de utilizar código nativo quando ncessário. Ela surgiu em 2015 de forma aberta com um anúncio do Facebook, que já estava utilizando em produção para seus aplicativos iOS e pouco tempo depois Android também expandindo as plataformas de alcance.

Ela surgiu com base no React (React Website, 2018), outra ferramenta desenvolvida pelo Facebook surgida em 2013 para o desenvolvimento de aplicações web. Após um tempo do seu lançamento, React foi dividido em duas bibliotecas sendo elas o próprio `react` e o `react-dom`, isso permitiu que toda a lógica do `react` de controlar uma árvore de componentes da aplicação e o ciclo de vida dos mesmos ficasse independente da web, sendo `react-dom` responsável por renderizar o que a web espera. E com essa mesma ideia foi possível o React Native, sendo ele o responsável pela renderização nativa para Android e iOS.

Desde o lançamento do React Native, houve outras pontes para outras plataformas com base nele, embora somente iOS e Android sejam suportados pelo Facebook, como por exemplo `react-native-windows`, mantido pela própria Microsoft, para o desenvolvimento para a Plataforma Universal Windows (do inglês, *Universal Windows Platform* - UWP).

React Native funciona numa arquitetura com 3 threads, a thread principal no dispositivo da aplicação, uma thread executando uma máquina virtual com Javascript e uma thread para servir de ponte entre as outras duas, dado que elas nunca se comunicam diretamente. A ponte é responsável por toda a comunicação e funciona de maneira assíncrona, coordenando a troca de mensagens serializadas em lotes. A thread executando o Javascript toma conta do `react` e da árvore de componentes e ciclo de vida da aplicação, já a thread principal fica responsável por renderizar tudo que é passado para ela e captar eventos que influenciem o estado da mesma para enviar pela ponte (KOTLIARSKYI, 2015). Na Figura 2.1 é possível ver o fluxo de um evento na arquitetura do React Native e na Figura 2.2 é mostrado um exemplo de código e como fica sua visualização na aplicação.

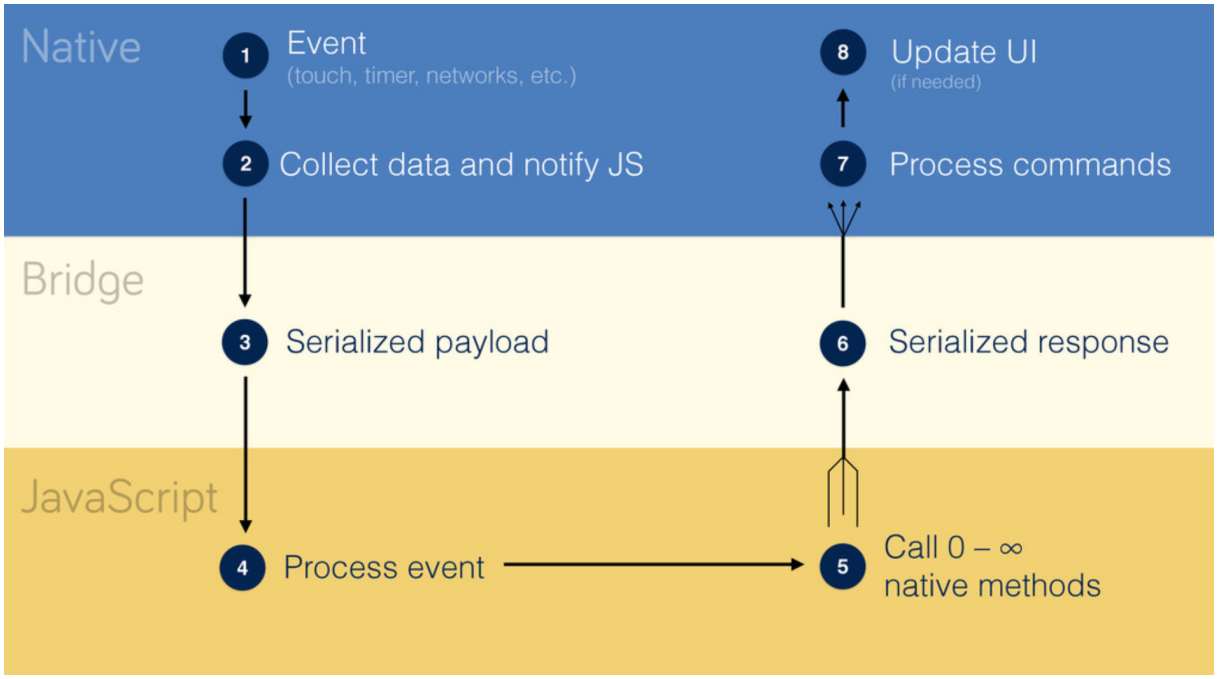

Figura 2.1 – Exemplo de comunicação entre as threads na arquitetura do React Native (KOTLIARSKYI, 2015).

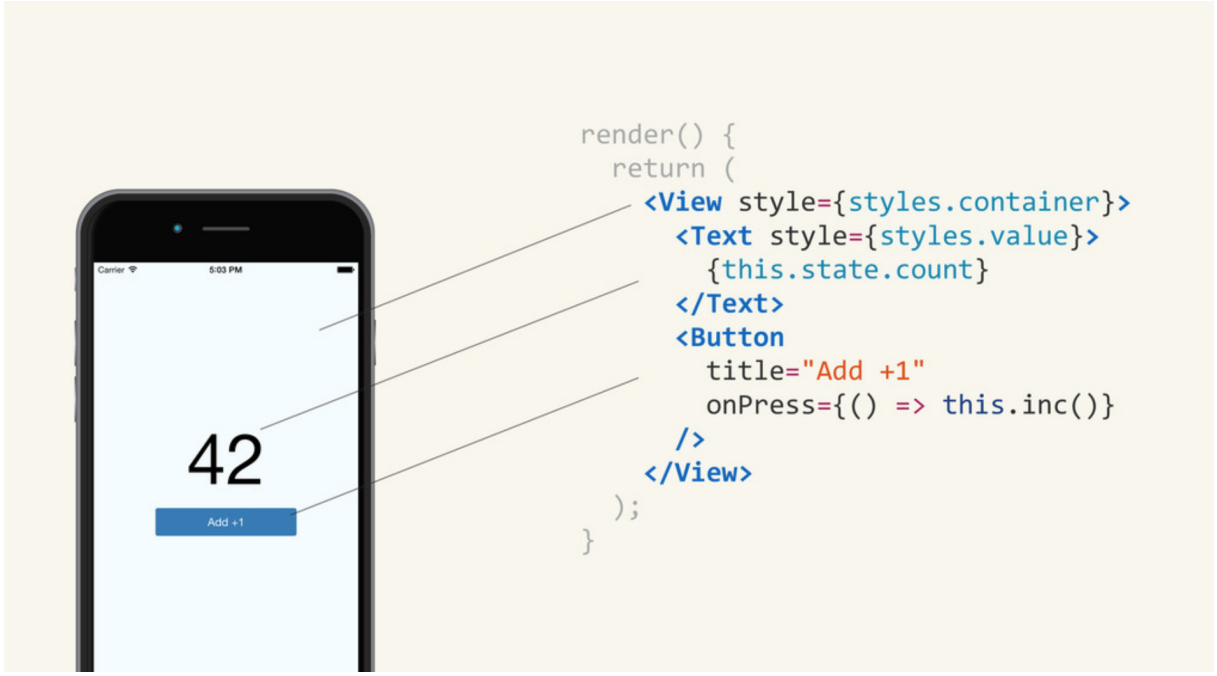

Figura 2.2 – Exemplo de código React Native e sua visualização (KOTLIARSKYI, 2015).

2.2 EXPO

Expo (Expo Website, 2018) é um framework em cima do React Native tendo foco em aplicações escritas inteiramente em Javascript. Sua ideia principal é fornecer uma gama

diversa de alto nível de funcionalidades para que os desenvolvedores possam se manter somente no Javascript e alcançar uma velocidade maior no desenvolvimento com menos conhecimento específico de plataforma.

Como o Expo oferece um ambiente de desenvolvimento baseado somente no Javascript, é possível começar a produzir a aplicação sem precisar do Android Studio ou Xcode, com o contra ponto que não é permitido adicionar módulos nativos que o Expo forneça, caso isso seja necessário, a aplicação precisará ser ejetada para o estado normal de um projeto React Native, ainda podendo utilizar o que o Expo oferecia da mesma maneira.

Ele possui um fork do React Native para controlar qual a compatibilidade entre seu kit de desenvolvimento de software (do inglês, *Software Development Kit* - SDK) e o produto do Facebook, podendo inclusive corrigir alguns bugs que assolam a tecnologia original, dando mais estabilidade para seus usuários. Nesta pesquisa iremos ver como ele tem aparecido nos projetos, como suas versões tem sido utilizadas e ver o que as pessoas mais utilizam dentro do que ele oferece.

3 ESTUDO

Este capítulo apresenta o estudo realizado neste trabalho, organizado de acordo com a Figura 3.1. A seguir estarão tópicos detalhando os passos, exceto a última parte do processo recebeu seu próprio capítulo a seguir para a discussão dos resultados.

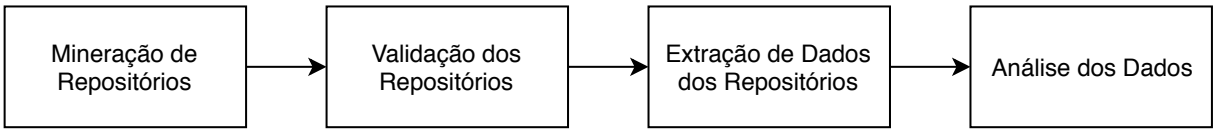

Figura 3.1 – Fluxograma do processo executado no estudo.

3.1 MINERAÇÃO DE REPOSITÓRIOS

O processo de mineração de repositórios foi a utilização de um script escrito em Typescript que realiza consultas na API v3 do Github em busca de repositórios com uma string de consulta base e intervalos de tempo para gerar as consultas efetivamente executadas.

Os intervalos de tempo foram necessários para contornar as limitações de uso da API do Github, que permite no máximo 1.000 resultados para uma string de consulta. Então fazendo uso de intervalos de tempo foi possível gerar diversas consultas mais específicas com relação à data de criação dos repositórios retornando um subconjunto do resultado final esperado e depois juntando tudo.

Para os dados desta pesquisa, a consulta base utilizada foi topic:react-native e os intervalos de tempo foram todas as semanas entre os dias 01/01/2015 e 20/11/2018. O resultado final das buscas foi uma lista de 11.547 repositórios com o tópico React-Native. Alguns deles poderiam ser duplicados, pois os intervalos de tempo não eram exclusivos.

3.2 VALIDAÇÃO DOS REPOSITÓRIOS

Possuindo a lista de 11.547 repositórios como resultado da primeira etapa, foi preciso validar quais deles realmente se encaixavam no conjunto esperado desta pesquisa utilizando efetivamente React-Native. Para realizar a validação, foi escrito outro script em Typescript que enviava requisições para o site https://raw.githubusercontent.com, site do Github que hospeda o conteúdo dos arquivos de cada repositório no sistema, em busca do arquivo package.json na raíz de cada projeto na branch padrão dos repositórios, dado obtido pela API do Github no passo anterior.

Os repositórios que não possuíam um arquivo package.json já eram descartados, já para os que possuíam foi feita uma busca pela chave react-native dentro das dependências, dependências de desenvolvimento e dependências de pares, também removendo os repositórios em que não havia a chave listada. Essa etapa reduziu a listagem inicial para 7139 repositórios para a extração de dados. Nesta etapa também foram removidos repositórios duplicados que vieram da mineração.

3.3 EXTRAÇÃO DE DADOS DOS REPOSITÓRIOS

Para iniciar a extração de dados dos repositórios todos eles foram baixados através do uso de git clone para o disco local, com a exceção de um repositório que não estava mais disponível na hora do download. Os repositórios foram mantidos no disco durante a pesquisa para facilitar sua reutilização caso houvesse a necessidade de reexecutar alguma análise.

Com os repositórios disponíveis no disco, outro script foi utilizado para consumi-los em busca das métricas. Para obter dados de versionamento, como a presença de ferramentas auxiliares foi novamente feita uma análise em cima do package.json em busca de chaves de bibliotecas listadas. Como para algumas ferramentas só isso não era suficiente, também foi verificada a presença dos seguintes arquivos no diretório: `yarn.lock` (arquivo gerado pelo gerenciador de pacotes YARN), `package-lock.json` (arquivo gerado pelo gerenciador de pacotes NPM), `exp.json` (arquivo de configuração para versões antigas do expo), `tsconfig.json` (arquivo de configuração de typescript), `.flowconfig` (arquivo de configuração de flow). Para determinar se o aplicativo nativo ou biblioteca eram direcionado para as plataformas, foi verificada a presença das pastas android e ios.

Já para métricas de uso do React-Native ou Expo, foi feita uma análise estática através de Babel, um compilador Javascript, para gerar uma árvore sintática abstrata do código e extrair tokens indicando o uso de API's do React-Native ou Expo. Para serem incluídos na análise os arquivos deveriam ter uma das extensões: `".js"`, `".jsx"`, `".ts"`, `".tsx"`e não estar dentro das pastas: `"node_modules"`, `"sample"`, `"example"`, `"demo"`, `"demos"`, `"Example"`.

Além das métricas acima, foi utilizado o git para obter a quantidade de commits do repositório excluindo os provenientes de merge.

4 RESULTADOS

Neste capítulo, todos os resultados encontrados são apresentados. A Figura 4.1 apresenta um gráfico com a distribuição de commits dos repositórios. Para as análises a seguir, os gráficos foram gerados agrupando os repositórios com até 50 commits e os que tem mais de 51 commits, para examinar se isso afetava os resultados encontrados.

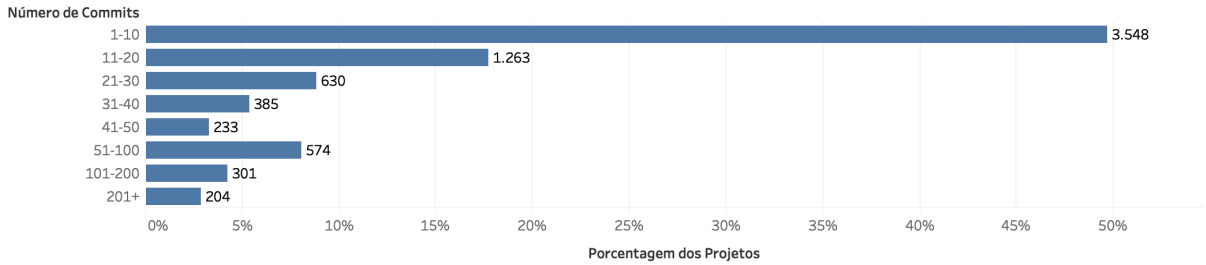

Figura 4.1 – Distruição dos projetos analisados pelo número de commits.

4.1 ANÁLISE DE CRESCIMENTO

Para analisar o crescimento do React Native, foi utilizada a data de criação dos projetos no Github e visto como o número cresceu ao longo dos anos.

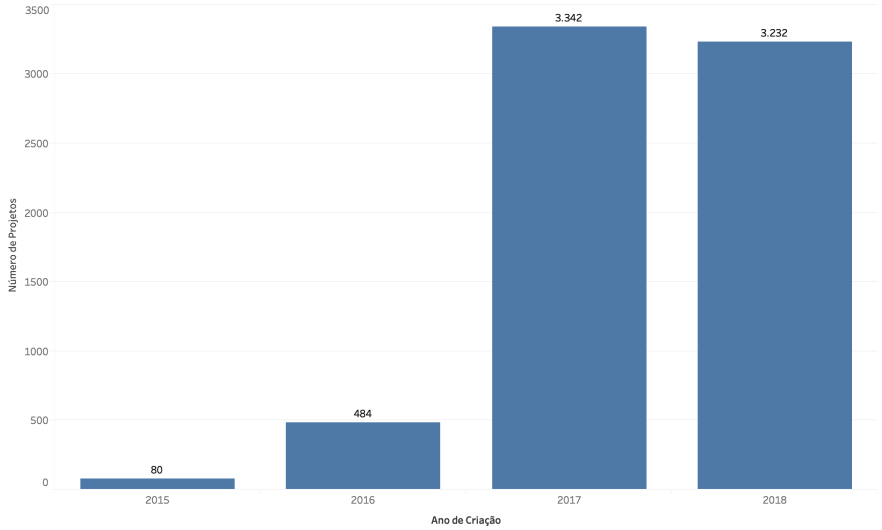

Figura 4.2 – Crescimento de projetos ao longo dos anos de 2015 até 2018.

Conforme a Figura 4.2 ilustra, foi visto um crescimento por volta de 6 a 7 vezes ao passar dos anos de 2015 até 2017 com uma estagnação de 2017 a 2018. Para a análise em questão, 2017 continua sendo o ano com mais projetos por uma pequena margem, podendo ser ultrapassado nos meses de novembro e dezembro de 2018.

4.2 ANÁLISE DE TIPOS DE PROJETOS

Para esta análise, os projetos foram separados entre aplicativos e bibliotecas. Para os aplicativos ainda foram considerados os desenvolvidos somente com React Native e os que utilizaram o Expo SDK. As Figuras 4.3 e 4.4 apresentam os resultados encontrados separados por ano.

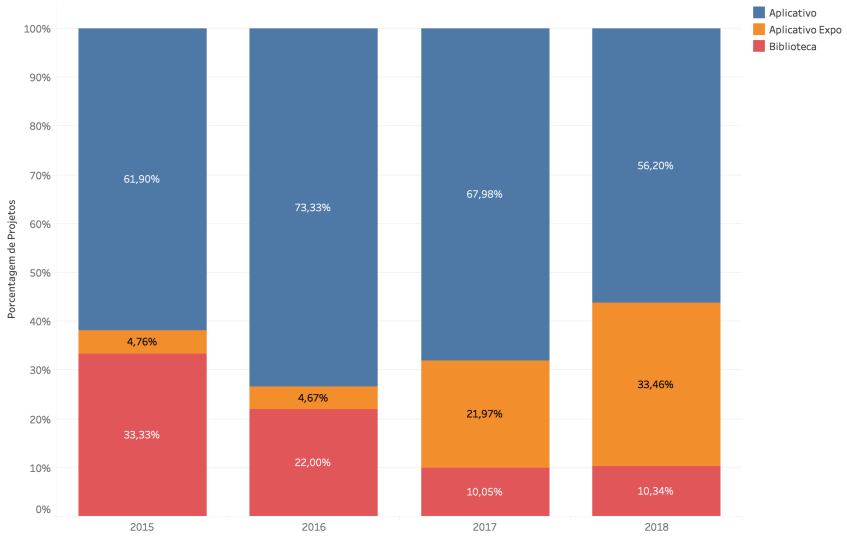

Figura 4.3 – Distribuição dos tipos de projetos ao longo dos anos de 2015 até 2018 para projetos com até 50 commits.

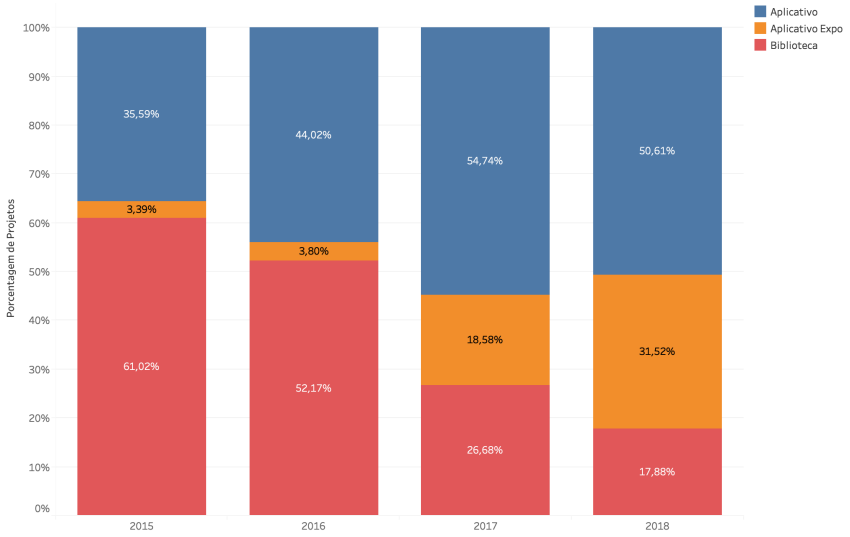

Figura 4.4 – Distribuição dos tipos de projetos ao longo dos anos de 2015 até 2018 para projetos com mais de 50 commits

Com base nos resultados, é possível ver que o número de aplicativos teve pouca alteração, enquanto o número de bibliotecas diminuiu consideravelmente. Também houve o crescimento de aplicativos desenvolvidos com o Expo SDK. Também é possível identificar que os projetos com mais commits possuem mais bibliotecas. Como a tecnologia tem seu

foco para o desenvolvimento de aplicativos e teve um crescimento no número de projetos ao longo dos anos, o número de bibliotecas não cresceu da mesma maneira e acabou tendo uma queda percentual em relação ao total de projetos.

4.3 ANÁLISE DE VERSIONAMENTO

Para a análise de versionamento, as versões extraídas durante o estudo foram normalizadas desconsiderando atualizações menores como por exemplo 0.57.2, as versões foram agrupadas pelo número do meio e em alguns casos não foi possível identificar a versão do projeto. Uma das preocupações desta análise é identificar se os projetos estão atualizados com a última versão disponível com base em sua data de última atividade, os projetos que não tiveram sua versão identificada ficaram marcados como indefinidos. As Figuras 4.5, 4.6, 4.7 e 4.8 mostram os resultados desta análise para todos os tipos de projeto como também separados.

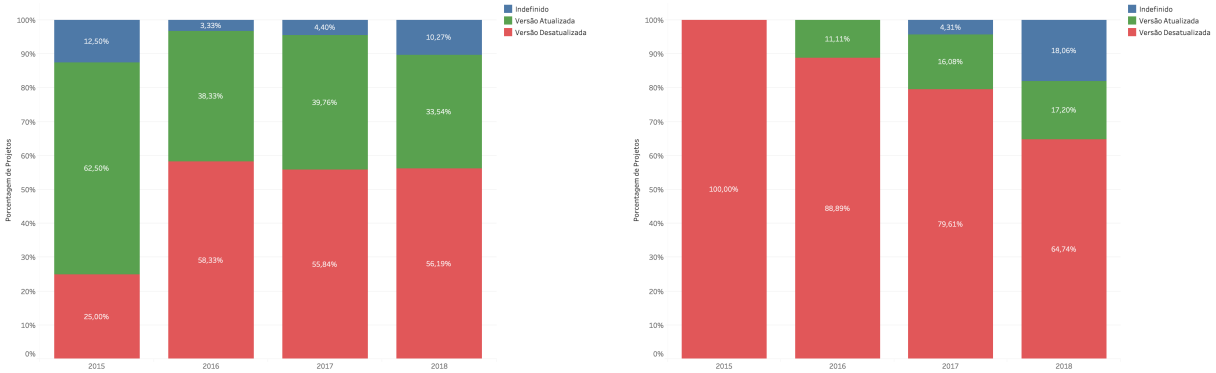

Figura 4.5 – Distribuição do status da versão dos projetos ao longo dos anos de 2015 até 2018. A distribuição da esquerda corresponde para projetos com até 50 commits, já a da direita para projetos com mais de 51 commits.

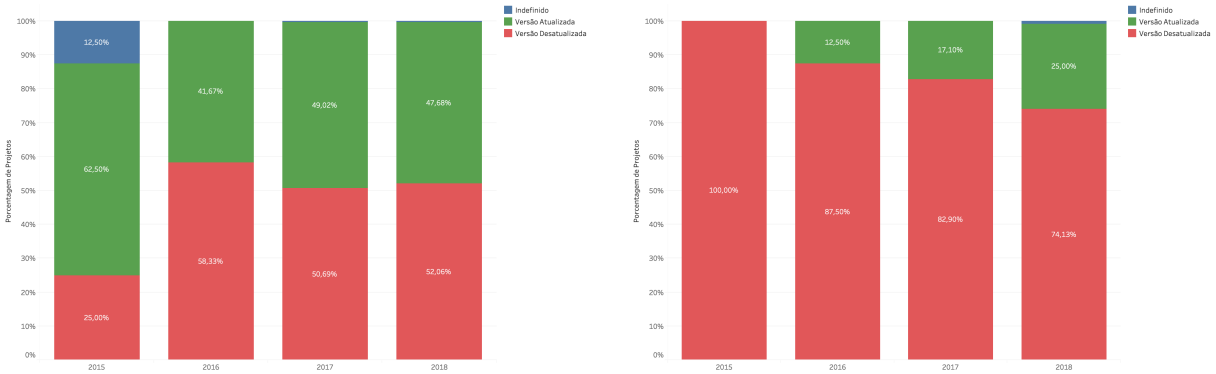

Figura 4.6 – Distribuição do status da versão dos projetos de aplicativo ao longo dos anos de 2015 até 2018. A distribuição da esquerda corresponde para projetos com até 50 commits, já a da direita para projetos com mais de 51 commits.

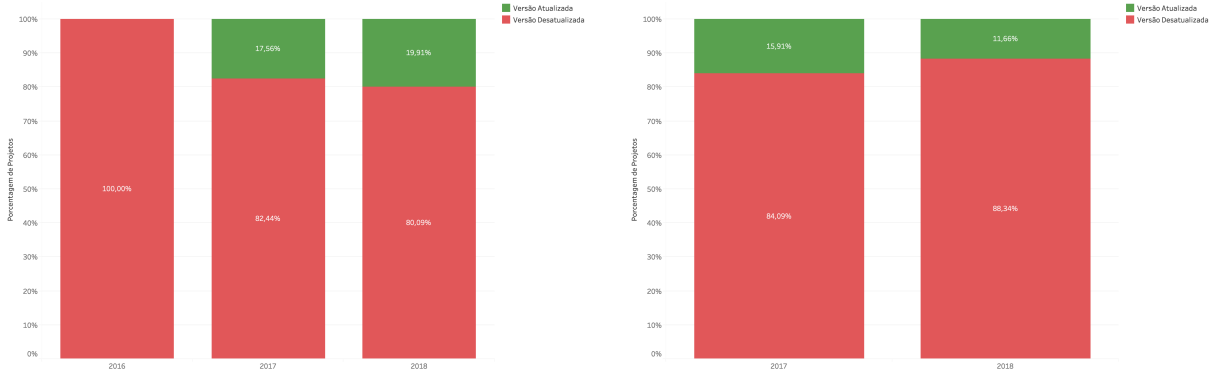

Figura 4.7 – Distribuição do status da versão dos projetos de aplicativos Expo ao longo dos anos de 2015 até 2018. A distribuição da esquerda corresponde para projetos com até 50 commits, já a da direita para projetos com mais de 51 commits.

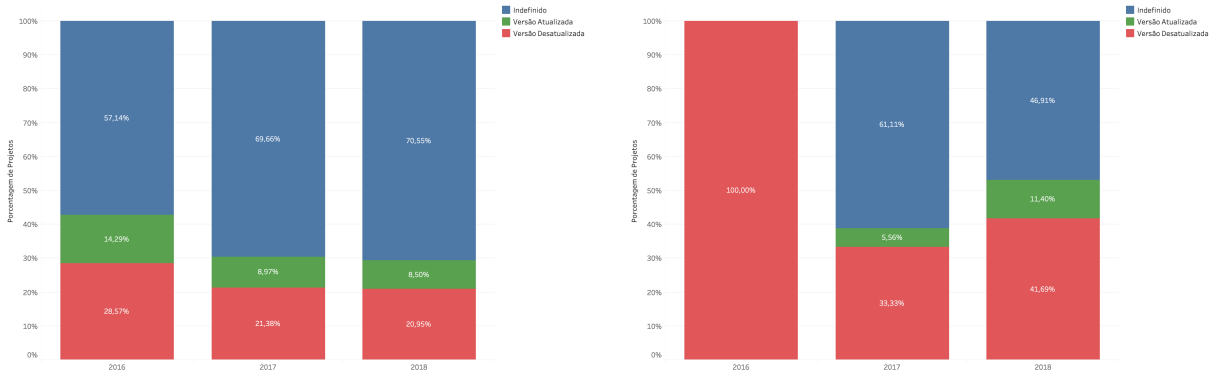

Figura 4.8 – Distribuição do status da versão dos projetos de biblioteca ao longo dos anos de 2015 até 2018. A distribuição da esquerda corresponde para projetos com até 50 commits, já a da direita para projetos com mais de 51 commits.

É notório que mais da metade dos projetos, independente da quantidade de commits, utiliza uma versão defasada na sua atividade mais recente. Também é possível reparar que bibliotecas possuem maior índice de versões indefinidas, seja por má formatação ou ausência dela no repositório, sendo somente especificada como dependência de par. Os aplicativos que usam Expo tem o maior índice de versão desatualizada, e isso provavelmente está ligado ao fato que cada aplicativo Expo utiliza uma versão do SDK que é compatível com uma versão específica do React Native e nem sempre é a última versão lançada em relação a atividade do repositório. Nos projetos menores o número de projetos desatualizados se manteve quase que constante nos últimos anos, uma possível explicação para isto é que boa parte dos projetos possuem poucos commits e provavelmente foram iniciados com a versão mais atualizada na sua criação e não ficaram ativos tempo o suficiente para uma nova versão ser lançada, uma boa fonte para esses projetos pode ter sido tutoriais na internet. As bibliotecas possuem uma grande porcentagem de versão indefinida por má

especificação no `package.json`, sendo complexo resolver a expressão para definir a versão utilizada.

Dado que o Expo tem um versionamento separado, também foi analisado como os projetos deste tipo estão atualizados em relação ao SDK na Figura 4.9. E o mesmo comportamento presente no React Native se repete no Expo com mais da metade dos projetos utilizando uma versão desatualizada. Esse comportamento é intensificado em projetos maiores, o que pode estar relacionado ao custo de atualização ser maior como o próprio projeto.

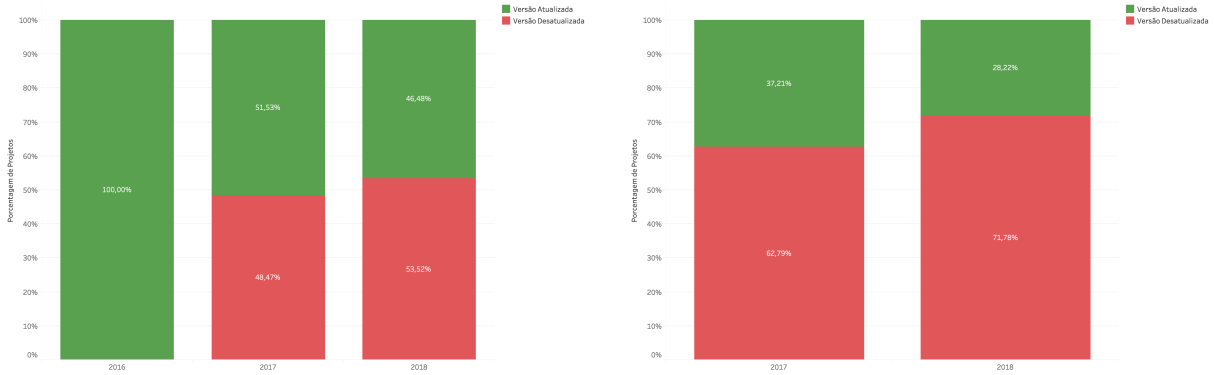

Figura 4.9 – Distribuição do status da versão do Expo SDK dos projetos ao longo dos anos de 2015 até 2018. A distribuição da esquerda corresponde para projetos com até 50 commits, já a da direita para projetos com mais de 51 commits.

4.4 ANÁLISE DE USO DO REACT NATIVE

Esta análise tem como foco identificar quais são as interfaces mais utilizadas do React Native nos projetos. Pode ser um ponto de referência para ver o uso e priorizar documentação ou definir se deve ser mantido no projeto principal ou se deve ser extraído para um projeto da comunidade.

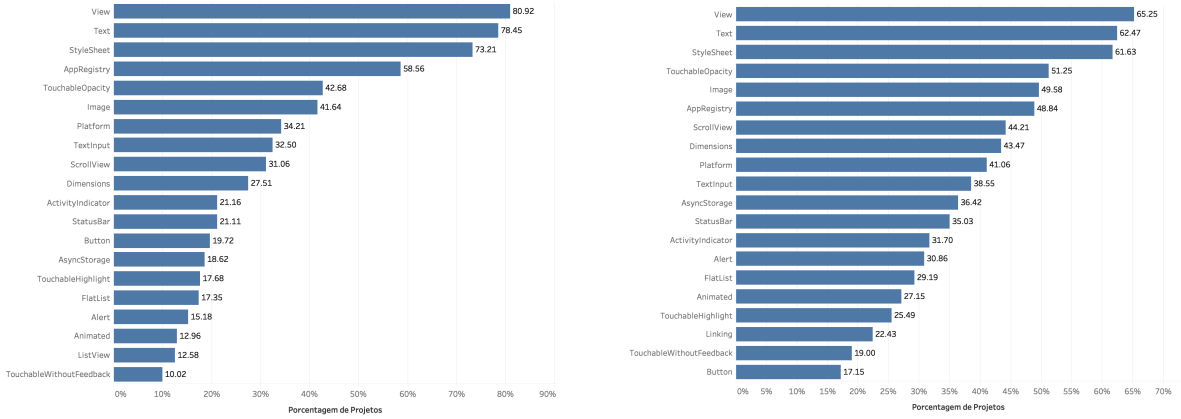

Figura 4.10 – Lista das 20 interfaces mais utilizadas de React Native nos projetos. A lista da esquerda corresponde para projetos com até 50 commits, já a da direita para projetos com mais de 51 commits.

Os resultados apresentados na Figura 4.10 apresentam que poucos projetos não fazem uso das interfaces mais básicas que podem compor uma aplicação como `View`, `Text` ou `Stylesheet`, mas poucos utilizam `Button`, provavelmente fazendo uso do `TouchableOpacity` como botão nas aplicações. `AppRegistry` tem um uso pequeno, apesar de obrigatório para aplicações nativas, não tem presença em aplicativos desenvolvidos somente com Expo ou bibliotecas. Menos da metade dos projetos utiliza a interface `Platform` que permite fazer alguma forma de distinção entre plataformas no código. `TouchableOpacity` aparece mais do que duas vezes do que as suas outras opções para interação, `TouchableWithoutFeedback` e `TouchableHighlight`. Poucos projetos fazem uso de `Animated` para incluir animações no aplicativo, ainda podendo fazê-las por meio de outras bibliotecas.

4.5 ANÁLISE DE USO DO EXPO

Esta análise tem como foco identificar quais são as interfaces mais utilizadas do Expo, similar a de cima, pode servir como referência para os mesmos propósitos.

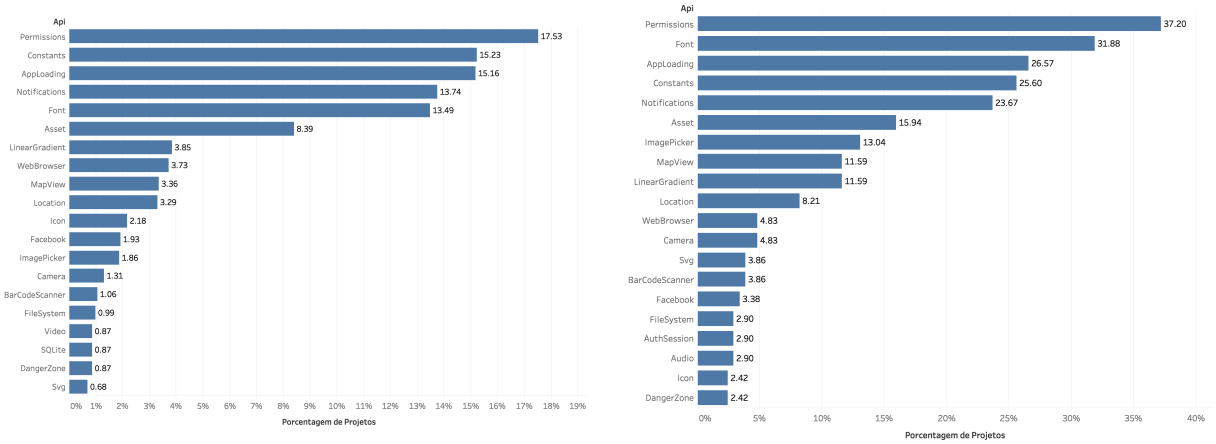

Figura 4.11 – Lista das 20 interfaces mais utilizadas de Expo nos projetos de aplicativo Expo. A lista da esquerda corresponde para projetos com até 50 commits, já a da direita para projetos com mais de 51 commits.

Com os resultados da Figura 4.11 é possível ver que as interfaces do Expo tem um uso mais distribuído do que as do React Native, provavelmente porque ele oferece interfaces de mais alto nível, então elas tem caso de uso distinto e específico, não podendo ter uma base comum como `View` é no React Native. E em ambos os tamanhos de projetos a interface mais utilizada é a `Permissions`, importante para controlar o que a aplicação vai requerer do usuário, simplificando bastante a forma como é feita somente com React Native tendo que controlar manifestos no Android e arquivos de propriedades do iOS além de incluir no código.

4.6 ANÁLISE DE DEPENDÊNCIAS

Esta análise tem como foco simplesmente identificar quais são as dependências mais comuns nos projetos de React Native, os resultados são apresentados na Figura 4.12. E é possível analisar que não há uma diferença muito grande nas dependências utilizadas entre projetos grandes e pequenos. Sendo mais notável uma variação na posição da lista das 20 dependências do que uma presente em uma lista e não em outra. Vale ressaltar que em praticamente todos os projetos React está presente, por ser um requisito obrigatório do React Native.

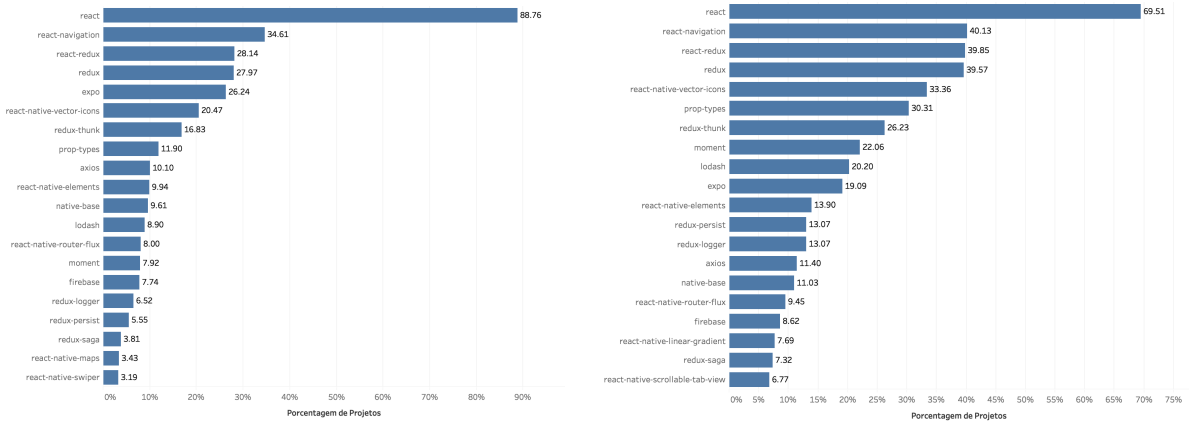

Figura 4.12 – Lista das 20 dependências mais utilizadas nos projetos. A lista da esquerda corresponde para projetos com até 50 commits, já a da direita para projetos com mais de 51 commits.

Um destaque dos resultados é que ambos os grupos de projetos incluem `redux` e `react-navigation` entre as quatro primeiras colocações. Sendo `redux` um gerenciador de estado da aplicação muito popular no desenvolvimento web, principalmente com React, que aparentemente fez uma boa transição pro ambiente mobile estando presente em mais de um quarto dos projetos e `react-navigation`, estando presente em pelo menos 4 vezes mais projetos do que a sua concorrente na lista `react-native-router-flux`. Vale ressaltar que em ambas as listas as bibliotecas mais utilizadas normalmente não tem ligação com React Native, mas somente com Javascript, mostrando como é compatível a utilização do vasto ecossistema para o desenvolvimento móvel.

4.7 ANÁLISE DE LINGUAGEM

No ecossistema de Javascript existem duas alternativas populares para adicionar tipos estáticos na linguagem durante o desenvolvimento com o propósito de evitar bugs e facilitar a manutenção, sendo elas Typescript e Flow. A distribuição de uso dessas ferramentas se encontra na Figura 4.13.

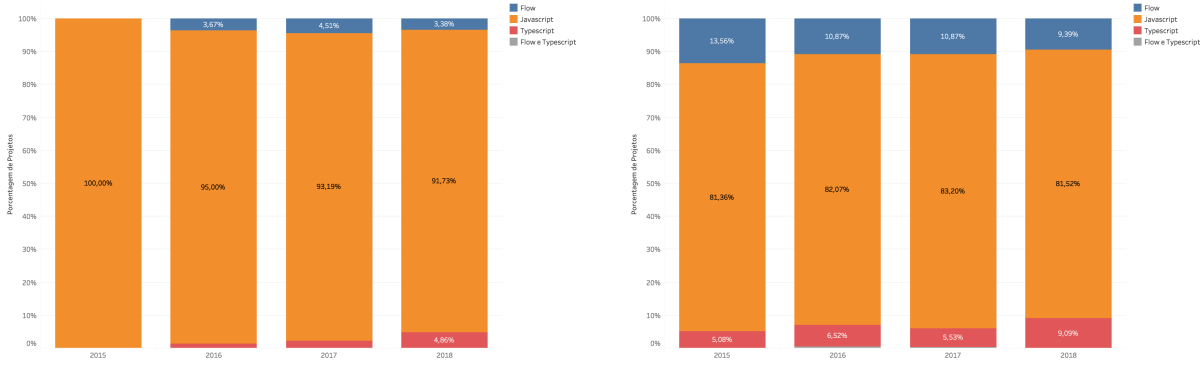

Figura 4.13 – Distribuição da linguagem utilizada ao longo dos anos de 2015 até 2018. A distribuição da esquerda corresponde para projetos com até 50 commits, já a da direita para projetos com mais de 51 commits.

De acordo com os resultados, projetos maiores utilizam pouco mais de 2 vezes uma forma de tipagem estática na linguagem para o desenvolvimento em relação a projetos menores. Não foi identificado uma discrepância entre o uso de Flow e Typescript, entretanto ambos estão com pouca presença nos projetos, mesmo nos grandes.

4.8 ANÁLISE DE GERENCIADOR DE PACOTES

Para fazer o uso de bibliotecas os projetos normalmente utilizam alguma forma de gerenciador de pacotes sendo eles NPM e Yarn. Para identificá-los foi verificada a existência de seu arquivo de controle na raíz do repositório. Este arquivo serve para garantir que o download das dependências seja determinístico na resolução de versões.

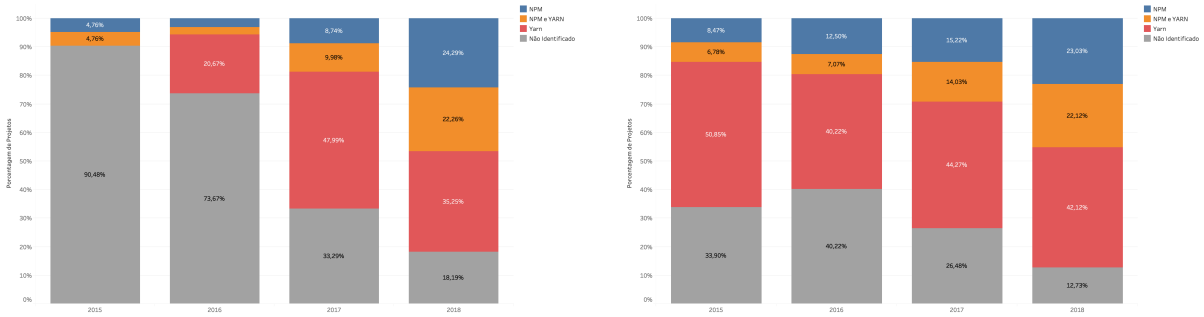

Figura 4.14 – Distribuição da gerenciadores de pacote utilizada ao longo dos anos de 2015 até 2018. A distribuição da esquerda corresponde para projetos com até 50 commits, já a da direita para projetos com mais de 51 commits.

A Figura 4.14 mostra que não há uma grande diferença entre os projetos grandes e pequenos neste quesito nos projetos mais recentes. Alguns projetos utilizam ambos NPM e Yarn, mas Yarn está presente em mais da metade dos projetos. É possível que o Yarn tenha sua presença mais forte devido a ser uma ferramenta também desenvolvida pelo Facebook.

4.9 ANÁLISE DE PLATAFORMA

A análise de plataforma serve para identificar as plataformas que os projetos de aplicativos somente com React Native que estão sendo usadas como alvo. A Figura 4.15 mostra que o uso da ferramenta corresponde ao esperado de gerar aplicativos tanto para Android quanto iOS, em poucos casos só para uma delas, e outros poucos casos para nenhuma, provavelmente um erro na extração dos dados.

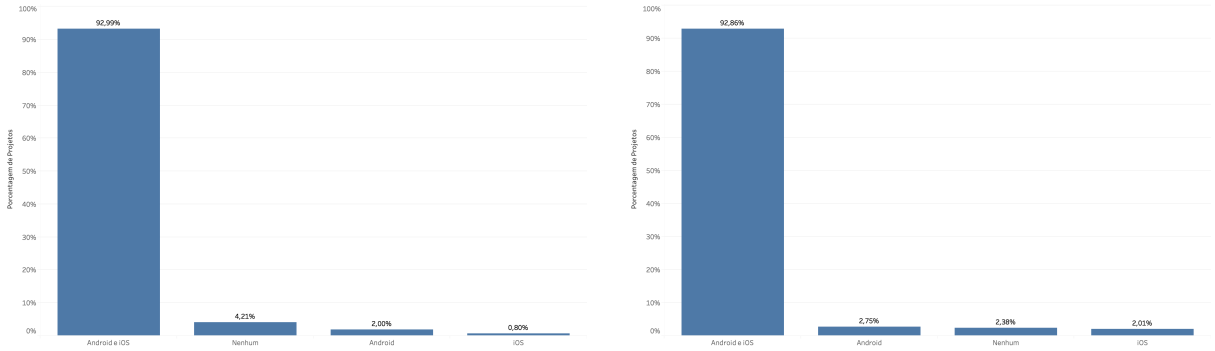

Figura 4.15 – Distribuição das plataformas alvo dos projetos de aplicativo. A distribuição da esquerda corresponde para projetos com até 50 commits, já a da direita para projetos com mais de 51 commits.

5 CONCLUSÃO

O React Native é uma tecnologia que cresceu nos últimos anos. Tem ganhado tração por estar inserida num ecossistema enorme do Javascript e por trás de uma empresa grande que a desenvolve e utiliza num dos aplicativos mais utilizados. Ela deu origem a uma nova forma de desenvolver aplicações multi plataformas, começando com iOS, depois Android e com uma arquitetura que permite adicionar novas plataformas, independente de ser um alvo de smartphones.

Seu crescimento foi tão rápido em popularidade quanto em versões de modo que a comunidade não acompanhou muito bem. Este trabalho mostrou que mais da metade dos projetos que utilizam React Native não a utilizam na sua versão mais recente e que uma grande porcentagem dos projetos são feitos inteiramente com Javascript. Além disso, desde o seu lançamento surgiu uma ferramenta feita em cima dela, Expo, e que também ganhou tração, estando presente agora em um terço dos projetos e fornecendo uma maneira simplificada para desenvolver as aplicações. E mesmo o Expo sofre do mal do React Native de ter muitas versões e poucos projetos atualizados.

O estudo foi capaz de mostrar através da análise de dependências que práticas comuns de desenvolvimento com React foram passadas para o React Native, como a presença do redux, uma biblioteca de gerenciamento de estado da aplicação, em mais de um quarto dos projetos. E que react-navigation é a opção mais popular para controlar a navegação das telas no aplicativo.

O trabalho havia como propósito fazer um panorama geral dos projetos de React Native atualmente e com isso identificar possíveis problemas para contribuir. Foi concluído que um dos maiores problemas é a forma como os projetos lidam com versão e um possível trabalho futuro poderia envolver uma ferramenta para realizar a atualização automática nos projetos realizando as correções necessárias, além de identificar mais fontes de dados para a análise, dado que a utilização de tópicos no Github necessita que o repositório tenha se classificado.

REFERÊNCIAS

Expo Website. *Expo*. 2018. <https://expo.io/>. Citado na página 13.

KOTLIARSKYI, A. *React Native: Under the Hood*. 2015. Disponível em: <https://speakerdeck.com/frantic/react-native-under-the-hood>. Citado 3 vezes nas páginas 6, 12 e 13.

POUSHTER, J.; BISHOP, C.; CHWE, H. *Social Media Use Continues to Rise in Developing Countries*. Pew Research Center's Global Attitudes Project, 2018. Disponível em: <http://www.pewglobal.org/2018/06/19/social-media-use-continues-to-rise-in-developing-countries-but-plateaus-across-developed-ones/>. Citado na página 10.

React Native Website. *React Native · A framework for building native apps using React*. 2018. <https://facebook.github.io/react-native/>. Citado na página 10.

React Website. *React – A JavaScript library for building user interfaces*. 2018. <https://reactjs.org/>. Citado na página 12.