soluções mobile

prof. Thyerri Mezzari

Desenvolvimento Mobile Híbrido
(ou multiplataforma)

Desenvolvimento Mobile Híbrido

A criação de um aplicativo focado em plataformas móveis demanda do desenvolvedor e sua equipe diversas escolhas, pontos cruciais para o sucesso e futuro do projeto.

Destaca-se o fato de cada plataforma possuir suas linguagens foco de desenvolvimento, o que exige da equipe de desenvolvimento muitas vezes uma grande multidisciplinaridade em programação, sem contar o maior tempo de produção e um investimento maior por parte do sponsor do projeto.

Por exemplo, como visto em sala de aula até o momento, para o desenvolvimento nativo na plataforma Android, somos requisitados a escrever uma base de código que envolve em sua grande parte Kotlin e/ou Java, sendo que não há qualquer possibilidade (até esse momento) de usar esta mesma stack para criação de apps na plataforma da Apple (iPhone OS, iPad OS ou Watch OS).

Desenvolvimento Mobile Híbrido

Com a intenção de proporcionar um modelo de desenvolvimento alternativo ao nativo visando resolver alguns problemas dos modelos tradicionais, como base de código duplicada, diferentes linguagens de programação envolvidas e prazo "dobrado", surge o desenvolvimento multiplataforma (ou híbrido).

O desenvolvimento multiplataforma (ou do inglês cross-platform) é uma prática na criação de softwares, produtos e serviços para no mínimo duas plataformas-alvo e sistemas operacionais (móveis ou não) orientados por uma base de código única visando a criação de um produto global.

Não original do mundo mobile, essas técnicas de desenvolvimento se tornaram muito popular nos últimos anos justamente com a velocidade e variedade de plataformas móveis, tendo grande players por trás como Facebook, Google, Apache Foundation, Adobe e etc.

Ferramentas

Desenvolvimento Mobile Híbrido

As ferramentas mais populares do mercado para desenvolvimento mobile multiplataforma na atualidade são:

- PhoneGap / Apache cordova
- Ionic
- Xamarin
- Flutter
- React Native
- Kotlin Multiplatform

PhoneGap / Apache Cordova

Apache Cordova é uma estrutura de desenvolvimento móvel de código aberto. Ele permite que você use tecnologias web padrão – HTML5, CSS3 e JavaScript para desenvolvimento multiplataforma. Os aplicativos são executados em wrappers direcionados a cada plataforma e contam com ligações de API compatíveis com os padrões para acessar os recursos de cada dispositivo, como sensores, dados, status da rede, etc

https://cordova.apache.org/

Ionic

O framework **Ionic** é considerado um plataforma completa (ou toolkit) de desenvolvimento mobile híbrido. Criado em 2012, sempre focado na mescla de HTML5 + JavaScript + CSS com o ambiente nativo de cada OS, hoje têm como base o framework Cordova.

O **Ionic** têm uma larga vantagem sobre os outras plataformas sobre o fato de possuir componentes visuais prontos "de fábrica" e também aceita que seu desenvolvimento principal seja escrito de diversas maneiras como Angular, React, Vue.js e até JavaScript puro.

https://ionicframework.com/

Xamarin

A plataforma **Xamarin**, calcada na linguagem C# com o conceito de reaproveitamento máximo de código para projetos cross-platform através de ligações entre o código C# e os componentes nativos de cada plataforma, visando obter o máximo desempenho. Você escreve em C# o código e o mesmo é compilado para cada plataforma através de bindings avançados.

Esta plataforma é muito popular em mercados enterprise especialmente depois que a Microsoft comprou a mesma e tornou-a open-source.

https://dotnet.microsoft.com/apps/xamarin

Flutter

O **Flutter** é um kit de desenvolvimento de interface de usuário (UI toolkit), de código aberto, criado pelo Google. Possibilita a criação de aplicativos compilados nativamente. Possui ótimo desempenho em seu produto final e usa como base uma linguagem de programação "bem particular", o *Dart*.

Uma das grandes vantagens do **Flutter** são seus componentes visuais prontos que trazem uma boa unidade entre as plataformas.

https://flutter.dev/

Kotlin Multiplatform / Jetpack Compose

Kotlin Multiplatform permite manter uma única base de código da lógica do aplicativo para diferentes plataformas . Você também obtém vantagens da programação nativa, incluindo excelente desempenho e acesso total aos SDKs da plataforma.

Graças ao **Compose Multiplatform** , uma estrutura de UI declarativa baseada em Kotlin desenvolvida pela JetBrains, você também pode compartilhar UIs entre Android e iOS para criar aplicativos totalmente multiplataforma:

https://kotlinlang.org/docs/multiplatform.html
https://www.jetbrains.com/lp/compose-multiplatform/

React Native

Por fim, destaca-se o framework **React Native** desenvolvido pelo Facebook com conceitos semelhantes ao Xamarin e o Flutter, priorizando o desempenho nativo como resultado de seus processos.

Sem qualquer traço de HTML em seu desenvolvimento, o JavaScript torna-se a linguagem padrão de desenvolvimento desta plataforma facilitando muito para os desenvolvedores vindos do ambiente front-end WEB em sua adesão.

A principal base do **React Native** é a lib *React* também do Facebook, a ideia é compor componentes usando React + JavaScript e estes componentes através de uma "ponte" (bridge na doc do RN) serem refletidos nativamente em cada plataforma.

https://reactnative.dev/

Desvantagens e vantagens

...do desenvolvimento multiplataforma

Desvantagens do desenvolvimento multiplataforma

Começando pelas desvantagens deste modelo de desenvolvimento, na maioria dos casos estão limitadas as ferramentas utilizadas e não ao conceito de criação multiplataforma.

No caso de ferramentas híbridas (entre o nativo e o não nativo) baseadas em HTML5, temos uma queda de desenvolvimento significativa pelo fato dos componentes visuais não estarem diretamente ligados a SDK nativa de cada plataforma.

O caso mais famoso da desvantagem deste modelo de criação de aplicativos (multiplataforma híbrido) foi exposto pelo próprio Facebook no ano de 2012. No caso desempenho entregue até então pelo seu aplicativo principal era bem inferior aos recursos 100% nativos.

Desvantagens do desenvolvimento multiplataforma

Ciente de uma deficiência no desempenho de seu aplicativo (principalmente no SO Android), devido a um desenvolvimento híbrido baseado em HTML5 de algumas partes de seu aplicativo, o ***Facebook*** iniciou um processo de recriação de seu aplicativo no ano de 2012.

A versão do app do Facebook para Android, até então, era um híbrido entre código nativo e HTML5, o que economizava muito tempo no desenvolvimento do aplicativo mas deixava o desempenho bem ruim para o usuário final.

Depois disto o mesmo foi refeito totalmente em código nativo do Android. Na prática, isso significou que o carregamento das telas ficasse mais rápido e diversas informações passaram a ser armazenadas na memória do aparelho em vez de serem baixadas da internet, melhorando a fluidez.

Desvantagens do desenvolvimento multiplataforma

Em geral aplicativos híbridos também tem grande dificuldade em lidar com questões envolvendo o hardware da plataforma.

Muitos Apps conseguem atender seus requisitos usando apenas os recursos mais básicos como *inputs*, *texts*. *Buttons*, *imagens*, *tables*. Contudo, quando é necessário explorar recursos mais elaborados do sistema operacional, ou do hardware, como *câmera*, *bluetooth*, *wifi*, *acelerômetro*, *etc*. o desenvolvimento fica um pouco mais complicado.

Lidar com múltiplas *tasks*, *services* e ainda gerenciar corretamente as preferências do usuário, consumo de bateria e ciclo de vida da aplicação pode ser um grande desafio quando comparado ao desenvolvimento nativo.

Vantagens do desenvolvimento multiplataforma

No quadro de vantagens, em um interessante artigo sobre este modelo de desenvolvimento a equipe responsável pelo popular aplicativo/serviço *Dropbox*, destacou importantes vantagens deste modelo de criação, como redução do investimento na criação e manutenção (muitas vezes dobrado), um único sistema de registro de erros multiplataformas, debug e procedimentos de correção e principalmente focar o time de produção em uma única solução de código independente da plataforma.

Warning ⚠️

disclaimers do professor

Aviso aos navegantes

Não obstante das técnicas e recomendações da ferramenta escolhida para o desenvolvimento multiplataforma, o desenvolvedor não poderá descartar ou ignorar as boas recomendações dadas pela Google ou Apple, uma vez que cada plataforma poderá ter seu produto final publicado através das lojas de aplicativos oficiais, que em muitos casos demandam um processo de aprovação e pré-requisitos.

Em um mundo ideal teríamos um design de produto único para cada plataforma, no desenvolvimento híbrido é possível fazer isso, separar alguns arquivos que irão rodar apenas em iOS e outros arquivos que rodarão apenas em Android. Mas no dia a dia dependendo do porte de seu app, seria mais fácil a utilização de frameworks visuais que garantem consistência de UX entre as plataformas, por exemplo algum framework baseado em Material Design.

Aviso aos navegantes

Em termos de performance eu os produtos gerados com ***Ionic, Flutter e React Native*** *se destacam*. As três soluções se dão muito bem em apps de pequeno porte. Para médio e grande porte eu descartaria o Ionic devido a sua base HTML5.

Por tanto em nossas próximas aulas a solução adotada para os novos conteúdos será o **React Native**, uma vez que poderá ser mais fácil para quem já passou pela matéria de WEB trabalhar com JavaScript e React em outro contexto (mobile no caso).

React Native: Quem é, o que come, como vive e *roda no meu PC*?

React Native

O **React Native** *(ou RN)* é um conjunto de tecnologias baseados na *lib React* que possibilita a construção de aplicativos nativos em diversas plataformas, utilizando uma base consistente de desenvolvimento baseada em JavaScript. O foco do framework é entregar de forma eficiente em todas as plataformas que você necessitar. O Facebook usa o React Native em múltiplos produtos e aplicativos e continuará a investir ativamente em seu desenvolvimento.

Disponibilizado publicamente em março de 2015 com lema "aprenda uma vez, crie para qualquer lugar" este framework do Facebook teve por objetivo inicial resolver um problema interno na manutenção dos mais diversos aplicativos para smartphone que envolvem a empresa, sempre evitando o uso de HTML5 ou outras técnicas de baixo desempenho.

React Native

O Facebook em um determinado momento entre 2013 e 2015 possuia muitas webview's em seu App principal e apesar da facilidade de ter um desenvolvimento "híbrido" veloz (utilizando HTML embutido em seu app poupava tempo de desenvolvimento) o desempenho era muito baixo, especialmente em dispostivos Android.

Dentre as diversas iniciativas que surgiram internamente para solucionar este problema, além do desenvolvimento nativo real, surgiu a questão de portar a biblioteca React (que vinha ganhando enorme popularidade no dev web) para que também funcionasse nativamente em um cenário mobile.

React Native

Atualmente o framework atende oficialmente (de fábrica) as plataformas iOS e Android, porém outras iniciativas open-source baseadas nos mesmos paradigmas já estenderam seu uso a softwares desktop para Mac, Ubuntu Linux e Windows.

Em um cenário de pequena ou média complexidade poucas requisitos do mundo "nativo" serão exigidos, muita coisa tende a funcionar de forma "mágica" no RN, porém conhecimentos de **JavaScript** (ou TypeScript) são bem relevantes para a evolução nesta stack. Por tanto devs com background de front-end web tendem a ter um início mais "suave" com o framework.

Setup React Native

o que precisamos para nosso primeiro projeto

Requisitos para "rodar" o React Native

Se vocês já estiverem curiosos para instalar o RN em suas máquinas, aí vai a lista base de um bom setup inicial:

- **Node.js** (versão 14.x ou superior)
- Java SE Development Kit (**JDK**)
- **XCode** e *Command Line Tools* (para iOS)
- **Android Studio** (para Android) + **Android SDK**, Android SDK Platform, Performance (Intel ® HAXM) e Android Virtual Device

https://reactnative.dev/docs/environment-setup

Instalando o React Native

Primeiro devemos preparar nosso ambiente seguindo os passos deste link:

https://reactnative.dev/docs/environment-setup

Depois partiremos para o ***pacote NPM*** do React Native em si.

Hoje em dia o caminho indicado usar o executável ***npx*** e rodar o cli "*on-the-fly*":

```
npx react-native init MeuPrimeiroProjeto
```

Também temos o Expo

Um ótimo caminho para adentrar o mundo do ***React Native*** é a plataforma **Expo**:

https://expo.dev/

Dentre suas ferramentas podemos destacar o Snack, que assim como o *Codepen* e o *codesandbox.io* do mundo WEB, proporciona a criação de apps simples via navegador com testes e emuladores nativos.

https://snack.expo.io/

Também temos o Expo

Em projetos de pequeno, médio e "semi-grande" porte podemos *(e devemos)* utilizar o **Expo** tranquilamente durante todo o processo, em muitos casos nunca será necessário "ejetar" o setup do projeto e poderemos ir do começo ao fim apenas usando o Expo.

Para instalarmos o **Expo** basta usar o seguinte comando:

```
npm install -g expo-cli
```

E depois para criarmos nosso primeiro projeto Expo:

```
npx create-expo-app my-app
```

Também temos o Expo

Mais informações sobre a instalação do EXPO (que eu recomendo fortemente):

https://docs.expo.dev/get-started/installation/

Outra jogada interessantíssima é instalar em seu celular (e outros dispositivos de testes) o aplicativo **EXPO GO**:

Para Android: https://play.google.com/store/apps/details?id=host.exp.exponent

Para iOS: https://apps.apple.com/app/expo-go/id982107779

React Native "puro" ou Expo?

Ao longo de nossas aulas, em exercícios e até para o projeto geral do curso, a resposta é: ***tanto faz***. Irei aceitar os dois "modelos".

Para mim o importante é a organização do projeto de vocês e principalmente o código JavaScript *(ou TypeScript)* dentro dos padrões exigidos pela lib React e as boas práticas do framework RN que **também se estendem ao Expo**.

Mas sem sombra de dúvidas criar um projeto dentro do ecossistema fornecido pelo **Expo** é muito mais rápido, estável e testável do que customizar uma aplicação em React Native do zero, principalmente para quem está iniciando.

Criando um projeto em RN

react-native init
expo init

Criando um projeto em RN

A primeira coisa que temos que ter em mente é: **voltamos ao mundo do NPM**. É provável que na disciplina de desenvolvimento Web, no módulo de front-end, diversas vezes era preciso recorrer ao terminal/shell/CMD e digitar os famosos comandinhos "npm install", "npm blablabla"?

Pois é, no mundo do React Native a **DX** (Developer eXperience) foi relacionada o mais próximo possível do mundo de front-end web para atrair justamente esse público de desenvolvedores.

Por tanto durante o desenvolvimento de um projeto RN (e com Expo também) na maior parte do tempo ficaremos com o Android Studio (ou o XCode no caso do Mac) fechados, tentaremos resolver o máximo de coisas possíveis apenas com as linhas de comando no terminal/shell/CMD.

Criando um projeto em RN

Vamos supor que em sua máquina você tenha uma pasta chamada Apps em sua pasta Documentos. Você poderia usar essa pasta para armazenar cada um dos seus projetos RN. Primeiro deveríamos acessar esta pasta antes de criar qualquer projeto:

`cd %userprofile%\Documents\Apps` ou

`cd C:\Users\nomedoseuusuario\Documents\Apps`

No caso de ***Linux / Mac*** o comando ficaria:

`cd ~/Documents/Apps`

Criando um projeto em RN

E na sequência podemos criar nosso primeiro projeto chamado ***ExemploApp***:

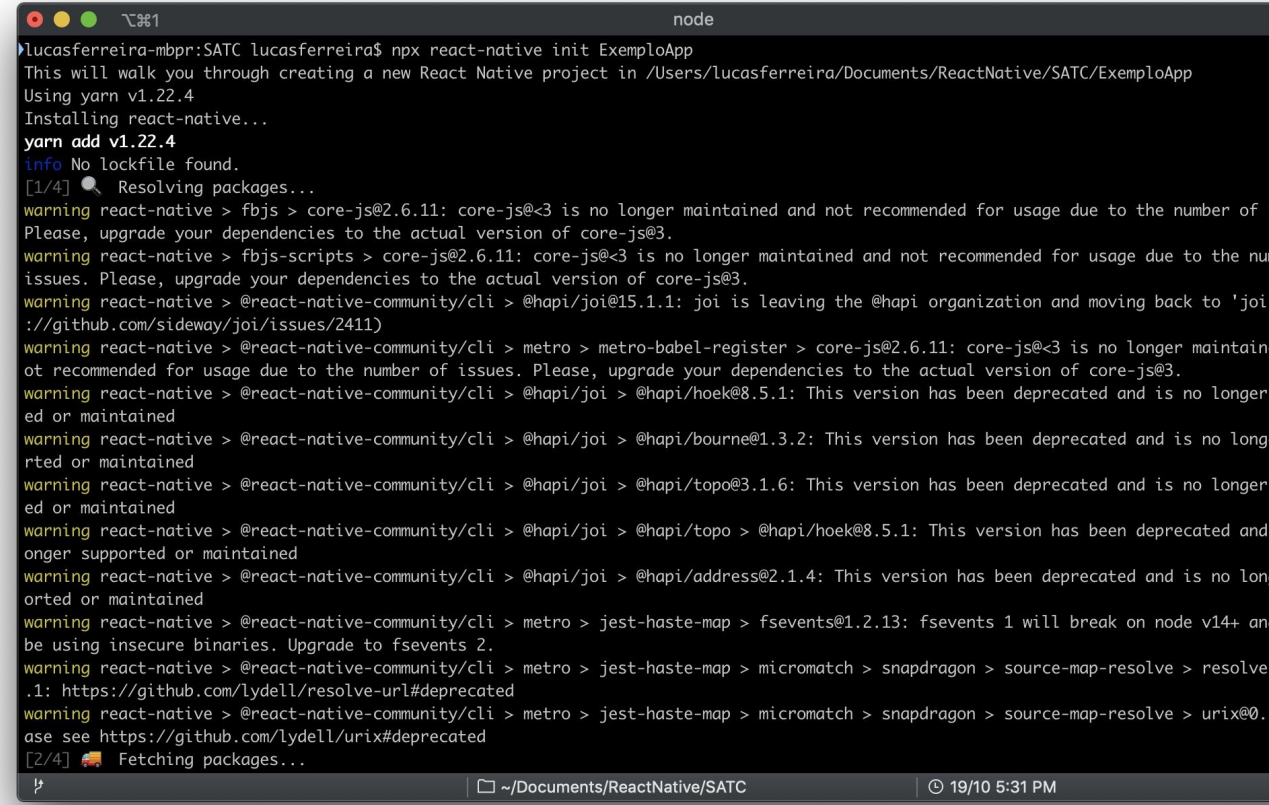
```
npx react-native init ExemploApp
```

No caso se quisermos usar o *Expo*:

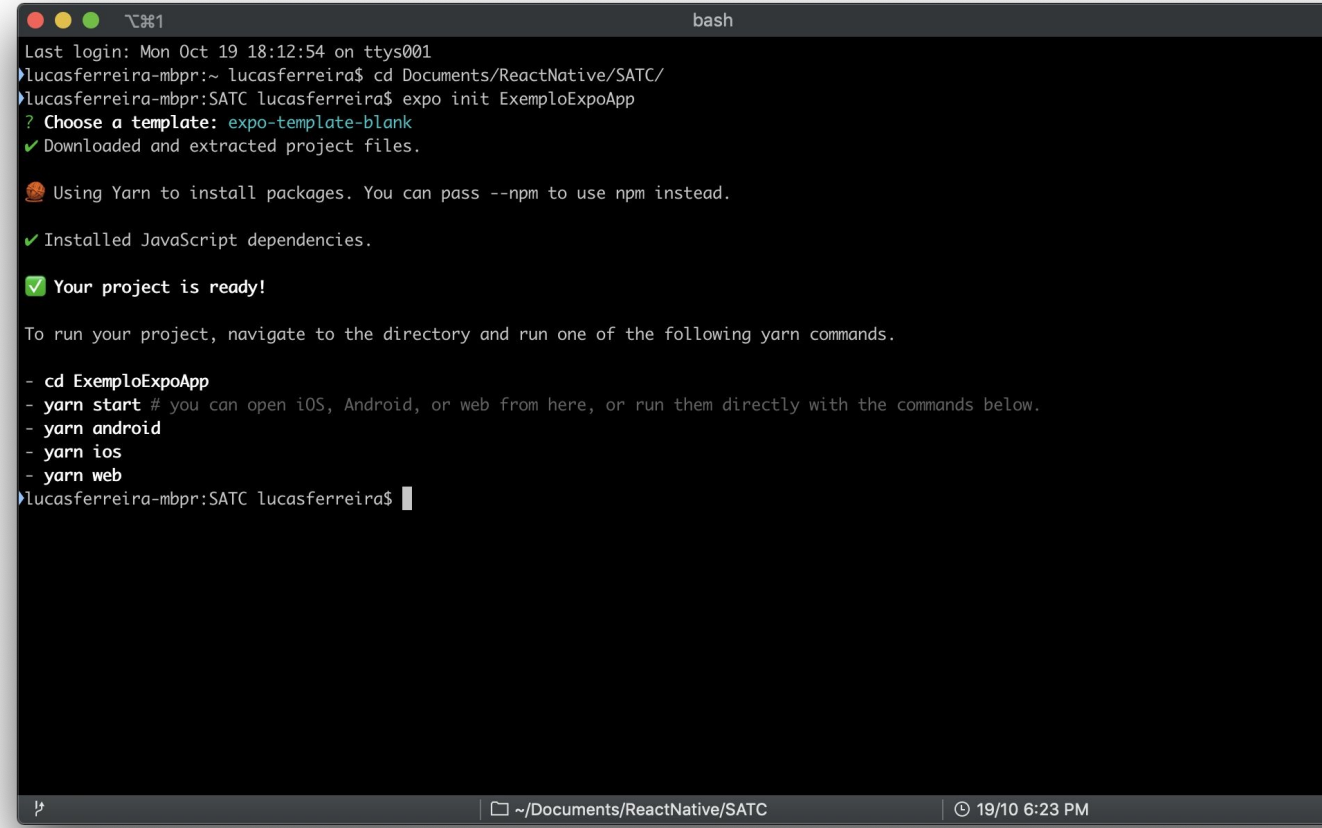
```
npx create-expo-app ExemploApp
```

Qualquer um dos dois comandos acima irá criar uma nova pasta de projeto chamada "**ExemploApp**" dentro de *~/Documents/Apps*.

react-native



expo


bash
Last login: Mon Oct 19 18:12:54 on ttys001
lucasferreira-mbpr:~ lucasferreira$ cd Documents/ReactNative/SATC/
lucasferreira-mbpr:SATC lucasferreira$ expo init ExemploExpoApp
? Choose a template: expo-template-blank
✔ Downloaded and extracted project files.
Using Yarn to install packages. You can pass --npm to use npm instead.
✔ Installed JavaScript dependencies.
✅ Your project is ready!
To run your project, navigate to the directory and run one of the following yarn commands.
- cd ExemploExpoApp
- yarn start # you can open iOS, Android, or web from here, or run them directly with the commands below.
- yarn android
- yarn ios
- yarn web
lucasferreira-mbpr:SATC lucasferreira$
~/Documents/ReactNative/SATC
19/10 6:23 PM

Acessando o projeto

Após criar seu projeto, deste momento em diante nossa pasta de trabalho será o **nome do projeto**:

```
cd ./ExemploApp
```

ou

```
cd %userprofile%\Documents\Apps\ExemploApp
```

Rodando o projeto

Conforme já mencionado anteriormente a DX do React Native é a mais próxima da web possível, por tanto teremos que iniciar *uma espécie de "servidor"* em uma das janelas de comando e deixar o mesmo rodando durante todo o tempo em que tivermos a necessidade de testar o app.

Basicamente estaremos trabalhando sempre com no mínimo 2 janelas do terminal/shell/cmd uma janela para manter o servidor de compilação ativo e outra janela para rodar os comandos de testes do aplicativo.

Para iniciar o serviço de compilação e publicação de apps em dev:

```
npx react-native start
```

ou se for Expo:

```
npx expo start
```

Rodando o projeto

Uma das vantagens do React Native (ou Expo) é a atualização em tempo real das mudanças que fazemos no código de nosso projeto. Esta é a principal serventia da janela de compilação que deverá ficar "aberta" o tempo todo.

Inclusive em modo de desenvolvimento você irá precisar do comando `start` inclusive para testar o app em seu emulador.

O próximo passo no caso de você precisar testar o seu app Android é abrir o emulador antes de solicitar o teste nele, em alguns setups de algumas máquinas nem sempre o RN encontra o local que seu emulador Android está instalado, por tanto eu recomendo abrir o Android Studio, ligar o emulador (vou demonstrar) e depois você pode fechar o Android Studio (mas deixar o emulador aberto).

Rodando o projeto

Por fim, após localizado e executado o emulador de Android, com a janela `start` já funcionando, você poderá rodar um dos comandos abaixo para testar seu app RN ou Expo:

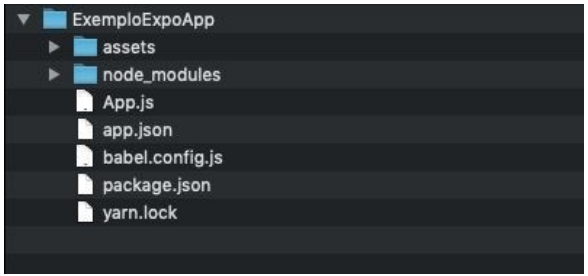
```
npx react-native run-android
```

ou

```
npx expo android
```

Estrutura do Projeto

Em relação a estrutura inicial de um projeto, vale destacar uma grande diferença entre um projeto RN "puro" e um projeto baseado em Expo:



Como podemos ver nas imagens acima, o projeto RN "puro" possibilita o acesso direto a base de seu projeto em cada plataforma nativa, enquanto no caso do Expo não temos acesso (inicialmente) a isto, **pois o Expo gerencia toda essa parte "avançada" para nós**.

Estrutura do Projeto

Caso você precise criar um código específico que rode apenas em uma determinada plataforma, existem dois caminhos óbvios:

1) Criar um arquivo específico, basta fazê-lo ***.android.js*** *ou* ***.ios.js***
2) Utilizar a API ***Platform*** (https://reactnative.dev/docs/platform-specific-code#platform-module) para criar um código específico

Demo time!

A ideia agora é rodar o projeto e mostrar para vocês um pouquinho de como se comportam os comandos e como fica a estrutura de arquivos dos projetos 😉

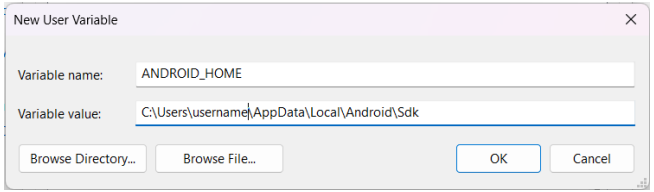
Demo time!

Para rodar os projetos nos computadores do laboratório UniSATC talvez seja preciso atualizar o NPM

```
npm install -g npm@10.5.2
```

Para executar o App no emulador Android também é necessário informar via variáveis de ambiente aonde está localizado o Android SDK. Faça isso pelas configurações do seu usuário no painel de controle do Windows.

Obs.: Está config é por máquina/usuário. Se mudar de máquina terá que fazer novamente.



https://docs.expo.dev/workflow/android-studio-emulator/#set-up-android-studios-tools

homework

Para quem tiver tempo:

- Acessar o guia de instalação do ***React Native*** -> https://reactnative.dev/docs/environment-setup -> e já ir instalando as necessidades.
- Sugiro cruzar as informações como: ***React Native CLI Quickstart -> Seu SO -> Android***.
- **Instale o Expo** e crie seu primeiro projeto.
- Quem quiser se adiantando... Teremos a versão *hibrida* da nossa Calculadora simples. 📱

obrigado 🚀