REACT NATIVE EXPO

Expo SDK 38 - melhorias que fazem do Expo uma opção ainda melhor

por **Thiago Marinho** há 8 meses 8 MIN DE LEITURA

Vamos falar sobre o novo SDK 38 do Expo, mostrar uma ferramenta de criação de Apps com Expo + React Native + Typescript que roda o projeto na Web. No final concluiremos dando a nossa opinião se vale a pena ou não criar um App com Expo. Confira! 🚀

✔ Como era com Expo Kit

Expo Kit era uma iniciativa de ter o código nativo em um projeto React Native, porém totalmente diferente do projeto React Native CLI. Realizar um eject era temível, era complexo de manter essa estrutura e tinha limitações.

A partir da versão 34 do SDK veio o conceito de Bare Workflow, um projeto React Native com as unimodules, que integrava grande partes das funcionalidades (câmera, geolocalização, etc) do Expo sem a necessidade da sua estrutura. Quando é feito o eject do expo tínhamos as pastas ios e android disponíveis.

Agora no SDK 38, temos apenas o Bare Workflow, onde temos as unimodules com a estrutura do React Native CLI, que nos fornece acesso as pastas android e ios, facilitando ainda mais a manutenção e integração com códigos nativos.

✔ Atualização SDK 38

No dia 25 de junho de 2020 o Expo atualizou o seu SDK para a versão 38. Até a versão 34 o SDK vinha com o Expo Kit. Na versão 38 o Expo Kit foi depreciado.

O novo SDK vem com a versão 0.62 do React Native que facilita a utilização do Dark mode com suporte ao módulo *Appearance* com o hook *useColorScheme*. Entretanto, para usarmos na **web** precisamos instalar o pacote react-native-appearance.

Teve várias melhorias na acessibilidade e no React DevTools. Temos também a nova LogBox que ajuda o(a) dev interpretar os *warnings* (alertas) na aplicação. Podemos usar o Flipper em projetos Bare Workflow. Foi removido o módulo de Realidade Aumentada (AR).

Os **highlights** ϟ da atualização são:

⊷ Nova API de Notificação

Expo Notifications foi refeita, pode ser usada em projetos com *Bare* e *Managed workflows*, ou seja com a estrutura do expo (*managed*) ou com React Native CLI com unimodules (Bare).

Essa solução é uma grande concorrente do **OneSignal** e **Firebase Could Message**.

⊷ Build time Updates In Bare workflow

Code Push é uma ferramenta desenvolvida pela Microsoft onde atualiza aplicativos sem passar pelas lojas, o update é feito direto no app já em produção. Como o bundle do JavaScript é um arquivo, e não impacta no código nativo, ele pode ser enviado (*code push*), portanto, ele é mais simples para colocar no app em produção.

Esse é o conceito de updates *Over-the-Air* (OTA), que vemos acontecendo principalmente em jogos — que alteram o conteúdo sem a necessidade de ir na loja fazer update, só de abrir o aplicativo já vemos as mudanças.

O expo-updates é o módulo responsável por esse funcionamento dentro do Expo e também é suportado em projetos Bare Workflow.

🔌 **Faster tooling, better performance**

Diminuíram o tamanho do pacote em 13% da versão anterior SDK 37, que é bem significativo, a `node_modules` fica um pouco menor. 😁

Foi aprimorado o *create-react-native-app*, agora suporta o Bare Workflow (dá acesso as pastas android e ios e vem com as funcionalidades do expo) e suporte para web com react-native-web.

Vamos falar sobre isso mais para frente quando formos criar o nosso projeto.

🆕 **Novas features:**

Saíram novas funcionalidades: autenticação, google fontes, captura de tela, barra de status, linking, slider, picker e armazenamento.

- *Vamos comentar algumas*:

Autenticação: A parte de autenticação foi totalmente reescrita *expo-auth-session*. E temos mais de 15 provedores para realizar autenticação.

Google fontes: Podemos usar as diversas fontes do google de maneira simples, antes era um pouco trabalhoso utilizar fontes externas no React Native. Não precisamos mais baixar as fontes no projeto.

Screen Capture: Módulo que fornece segurança para as telas do aplicativo, com ele você pode bloquear o usuário de tirar *print screen*

de alguma tela.

Expo **recomenda** que todos os projetos que estão na versão SDK 37 **migrem** para a versão 38, está simples realizar a migração e **estável**. Nesse post tem o passo a passo.

Todos os detalhes da **atualização** você pode ler **aqui**.

✔ Alternativas para usar as funcionalidades do Expo nos projetos

Temos **três alternativas** para um projeto Bare Workflow com acesso ao código nativo e funcionalidades da Expo:

1) Você já tem um Projeto React Native com CLI (sem expo)

Se você criou um projeto sem expo, você pode integrar os unimodules do expo no projeto.

Unimodules é uma maneira de ter as funcionalidades do expo sem ter criado projeto com expo.

Você pode seguir o guia de instalação do react-native-unimodules.

Tem como adicionar só os módulos do expo que você precisa. Exemplo, se quiser só a câmera do Expo você pode instalar só o módulo de câmera.

2) Você já tem um projeto criado com Expo e quer migrar para o Bare workflow

⚠ **Atenção:** Faça um backup do projeto.

Você vai rodar o comando abaixo na raiz do projeto, para atualizar a SDK:

```
expo upgrade
```

Depois vai ejetar:

```
yarn eject
```

Com isso seu projeto vai ser migrado para o Bare Workflow, se seu projeto estiver versionado com git você vai ver as diferenças (*diff*), a pasta android e ios vai ser criada para você.

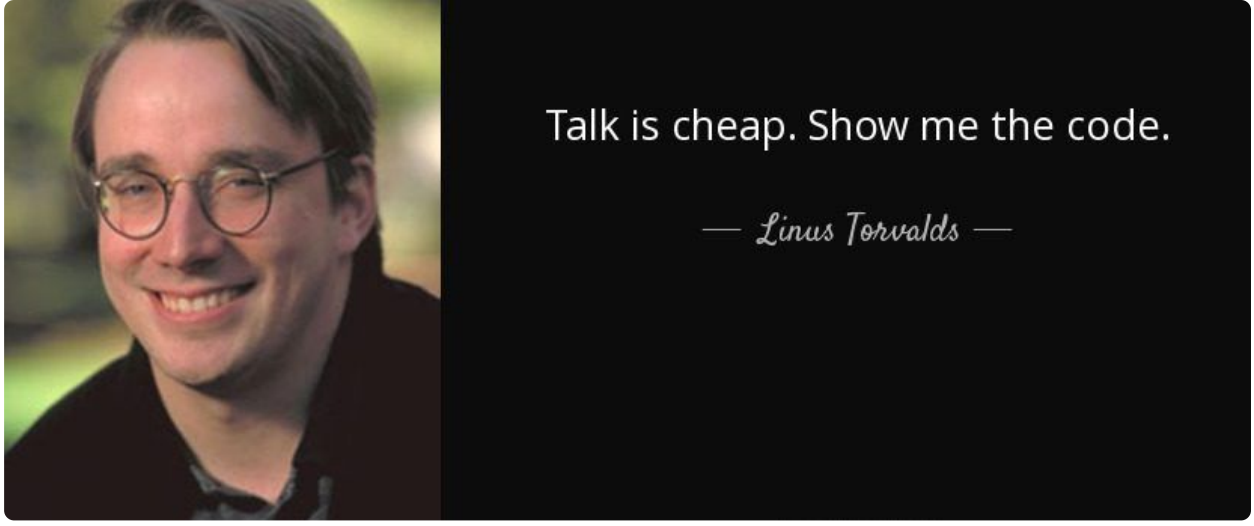
3) Criando um projeto novo

"Quero ter a produtividade do Expo, mas não quero ter suas limitações"

Pode utilizar o create-react-native-app que cria um projeto React Native com as unimodules e você já tem acesso as pastas android e ios. Atualmente é a maneira mais recomendada para utilizar em 95% dos casos. 5% fica por conta da otimização do bundle, uma dependência depende da outra, deixando o bundle grande.

É com essa ferramenta que vamos começar a criar nossos apps.

> ## Agora vamos para o código!

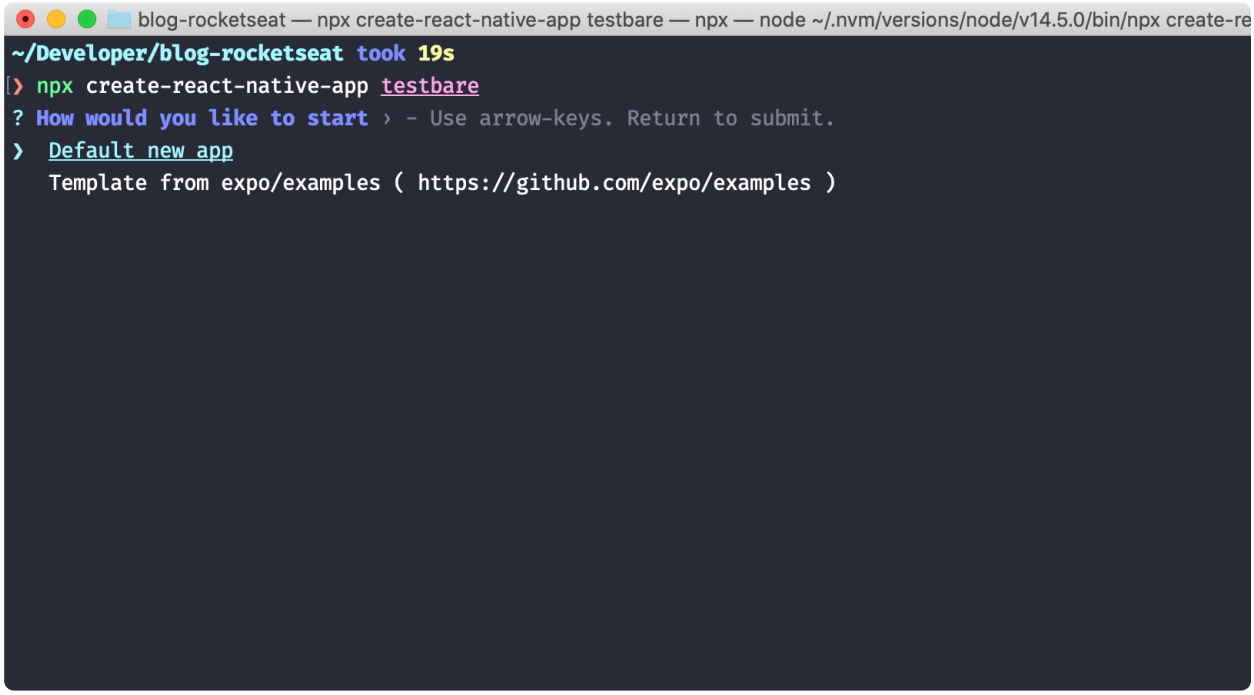


🤡💻 Criando um novo projeto com Expo usando create-react-native-app

Execute o comando abaixo que irá criar um app chamado testbare:

```
npx create-react-native-app testbare
```

Escolha: `Default new app`.

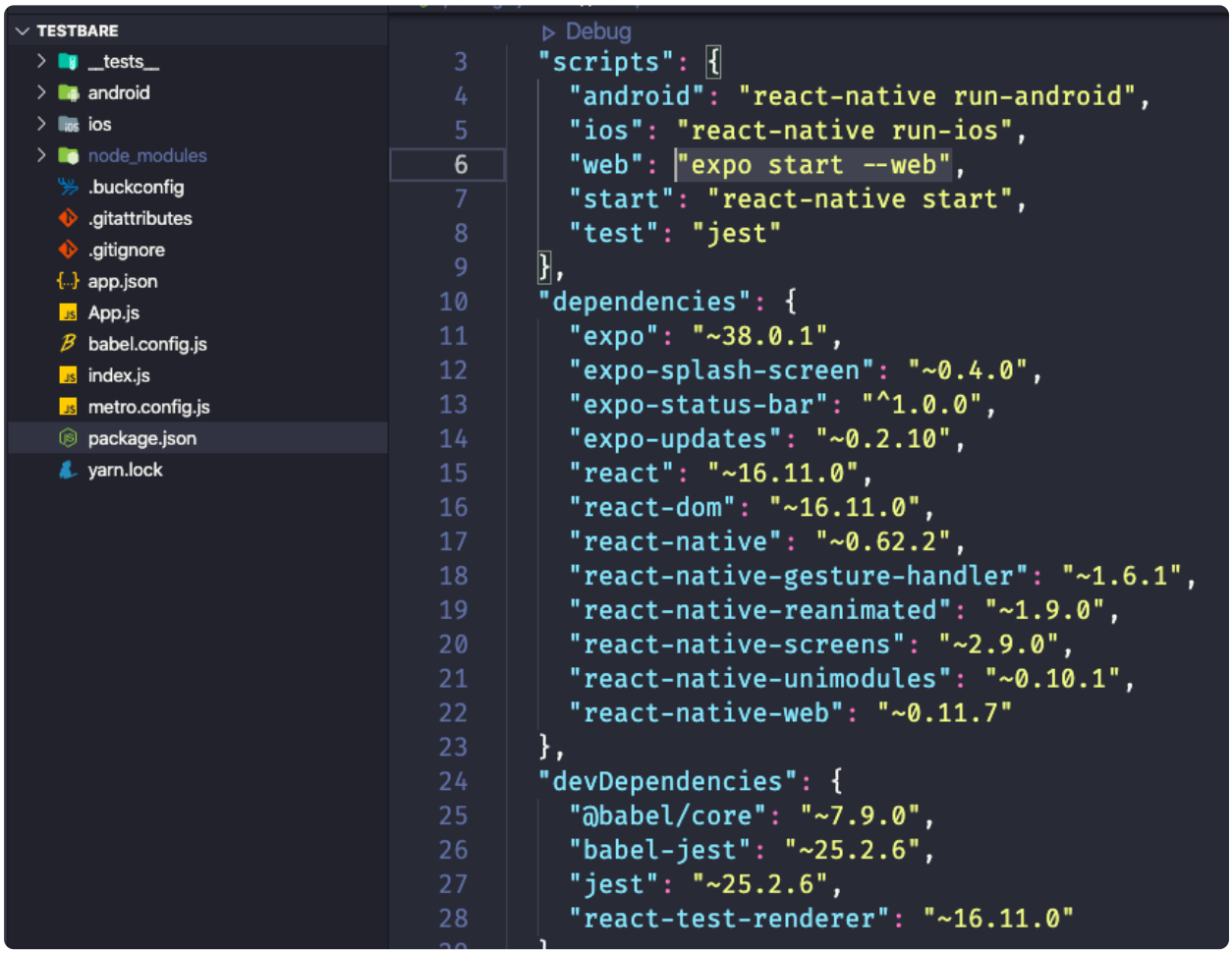


O projeto vai ser inicializado com **JavaScript**, mas para trocar para **TypeScript** é bem rápido e vamos fazer isso logo mais.

Depois do projeto criado, acesse sua pasta:

```
cd testbare && code .
```

Usando o **VsCode**, temos a seguinte **estrutura**:



Temos acesso ao **código nativo** com as pastas `android` e `ios`.

Verificando o `package.json` acima observamos que o expo foi instalado. Alguns módulos essenciais para todo projetos estão instalados também:

`expo-splash-screen` exibe uma telinha inicial do projeto, geralmente com a logo ou nome.

`expo-status-bar` barra de status onde mostra o relógio, bateria do celular, sinal do wifi, e outros status.

`expo-updates` módulo responsável por atualizar o código em produção sem precisar passar na loja de aplicativos.

Foi instalado também o `react-native-unimodules` que citamos bastante.

O mais surpreendente é o `react-native-web` e consequentemente `react-dom` que agora nos permitem usar o mesmo código para rodar na web. Reaproveitando não só conhecimento, mas também o código!

E temos o script: `expo start --web` para rodar o projeto no navegador. Podemos até criar um PWA.

```
"scripts": {
    "web": "expo start --web",
  },
```

Configurado o TypeScript

Renomeie o arquivo `App.js` para `App.tsx`

Crie o arquivo `tsconfig.json` na raiz do projeto, com o seguinte conteúdo:

```
{
  "compilerOptions": {
    "allowJs": true,
    "allowSyntheticDefaultImports": true,
    "esModuleInterop": true,
    "isolatedModules": true,
    "jsx": "react",
    "lib": ["es6"],
    "moduleResolution": "node",
    "noEmit": true,
```

```
    "strict": true,
    "target": "esnext"
  },
  "exclude": ["node_modules", "babel.config.js", "metro.config
}
```

Fonte: https://docs.expo.io/versions/v37.0.0/react-native/typescript/

Instale as dependências do TypeScript e as tipagens (*@types*) de algumas ferramentas:

```
yarn add --dev typescript @types/jest @types/react @types/reac
```

Pronto, só isso e já temos um projeto RN com TypeScript.

📲 Executando o projeto no Simulador iOS & Android

Pode instalar o App no seu emulador Android ou simulador iOS:

Android:

```
yarn android
```

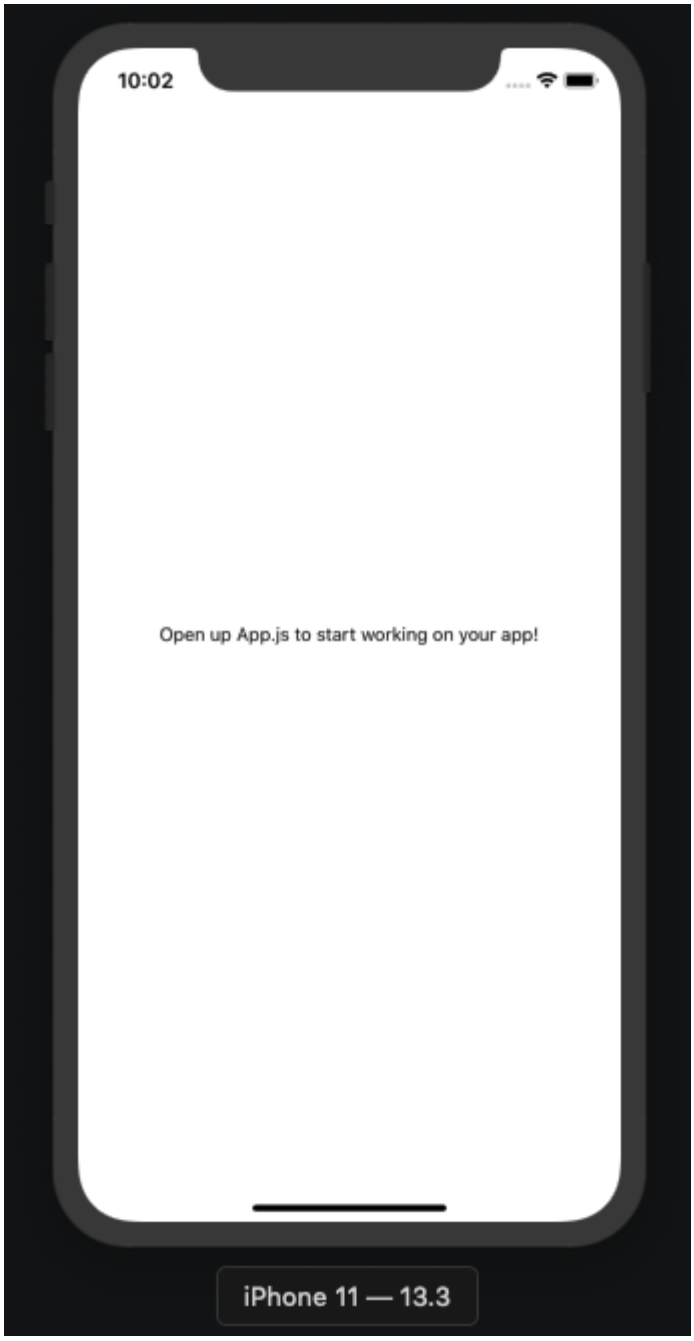
iOS:

```
yarn ios
```

Esse comando vai demorar uns 3 a 12 minutos, depende da configuração da sua máquina. Na primeira vez demora mesmo!

Drink your coffee now! ☕

Pronto 😴



"Ahh mas eu não tenho um macbook para rodar no simulador iOS e meu PC é fraco não consigo rodar no emulador do Android, o Android Studio já comeu todo meu HD". 😉

📱 Executando o projeto na Web

Executando o comando abaixo podemos rodar o projeto no navegador:

```
yarn web
```

😞 **Observação**: Se você tiver esse problema quando rodar o comando acima:

```
"__fbBatchedBridgeConfig is not set, cannot invoke native modu
```

Você precisa ter a última versão do expo-cli instalada.

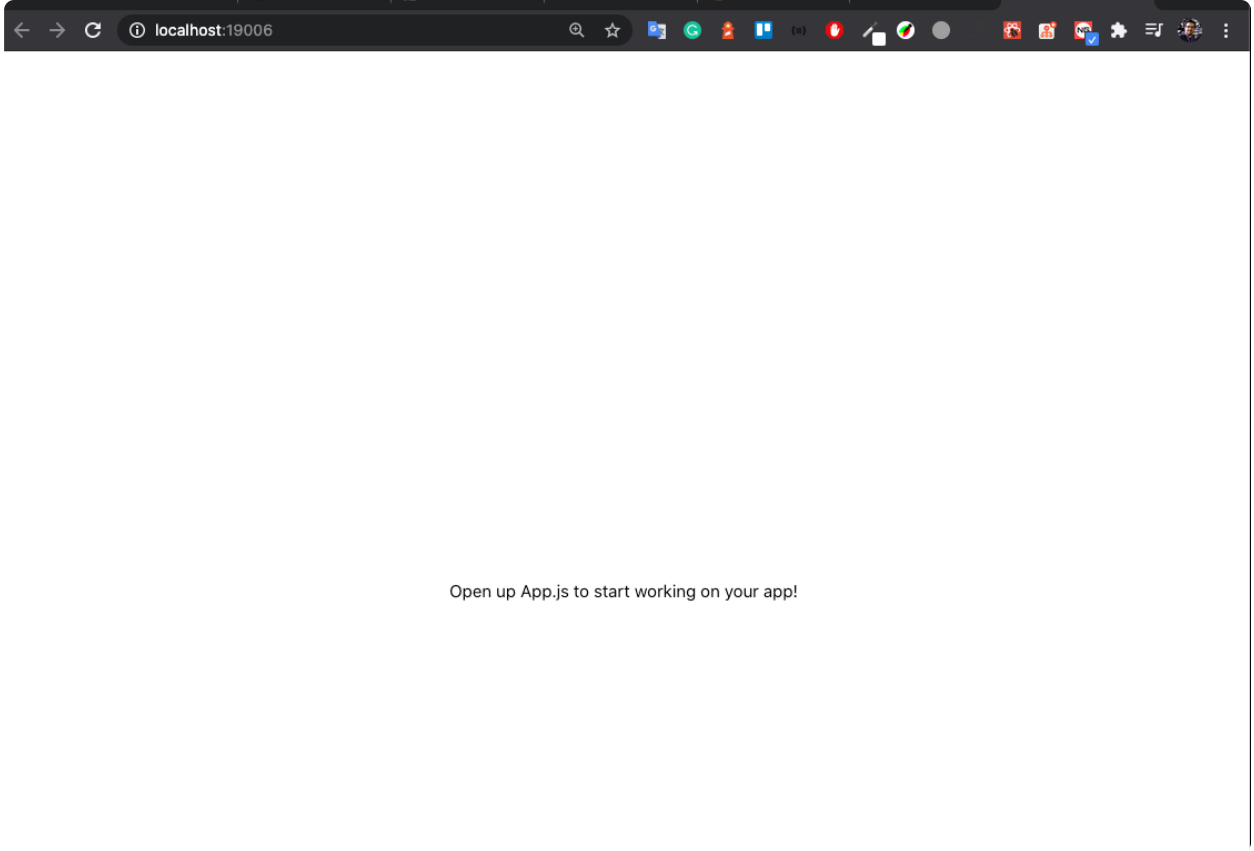
Execute o comando:

```
npm uninstall -g expo-cli && npm install -g expo-cli
```

Por fim:

```
yarn web
```

E agora está rodando: *http://localhost:19006* 🙌



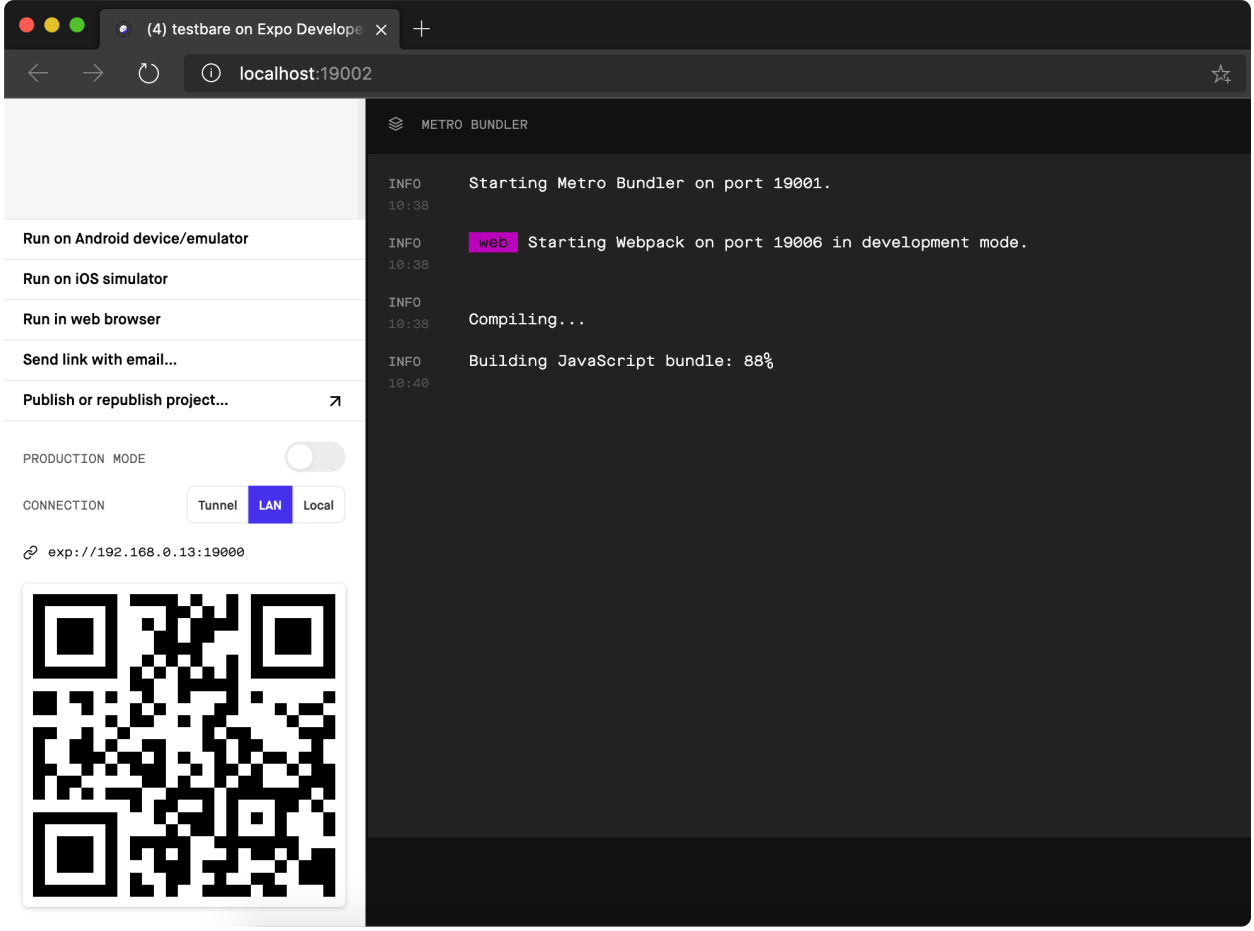
Expo Web: http://localhost:19006

📱 Executando o projeto no smartphone via conexão LAN

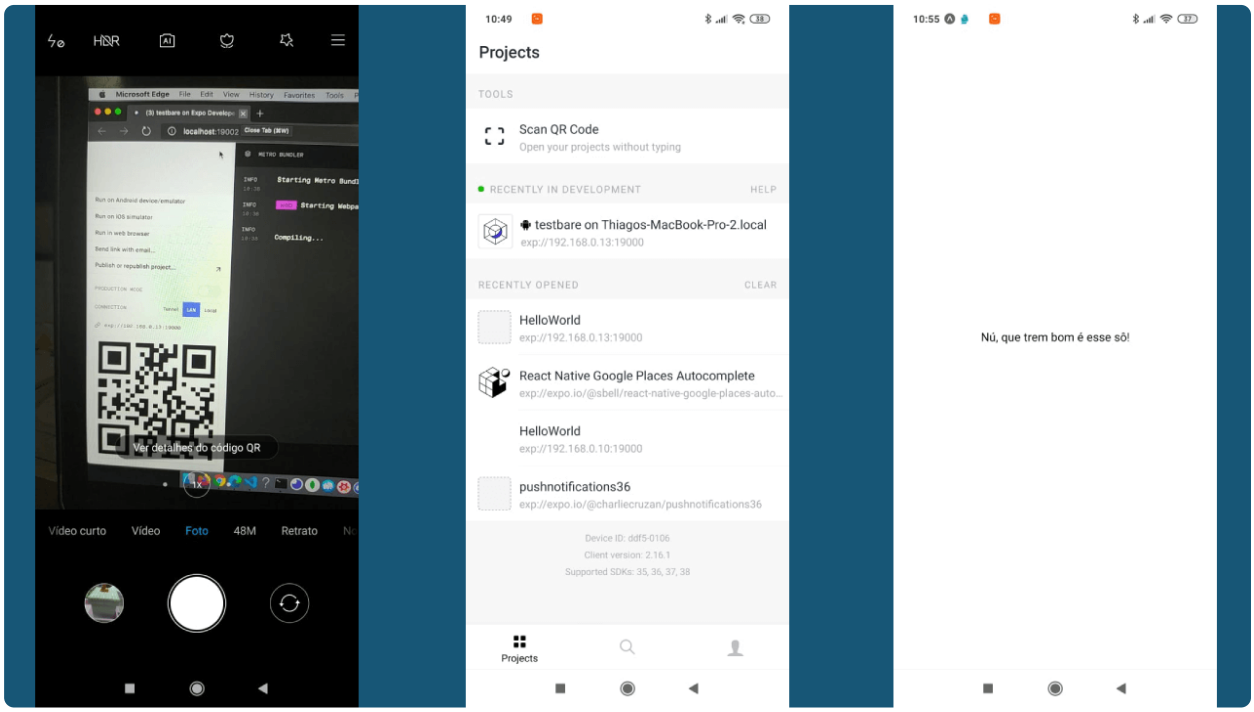
Como temos o Expo no projeto, podemos aproveitar desse benefício de executar o app no dispositivo real, ou seja, no seu smartphone (Android ou iPhone) e ainda usando apenas a conexão wifi para isso.

Cada alteração no código já reflete no seu próprio celular. Isso graças ao App do Expo que podemos baixar nas lojas de aplicativos.

Abra o app Expo do celular ou a camêra pra ler o **QRCode** abaixo e pronto, o app vai ser baixado no seu smartphone.



📱 Dispositivo físico:



A sensação de criar um App e poder testar nos emuladores, no meu próprio celular de maneira rápida é muito boa, ainda mais com a

possibilidade de trabalhar com libs nativas de maneira mais fácil e rodar o projeto na web é bem interessante.

✔ Conclusão

Expo mudou muito em dois anos para cá, evoluindo positivamente, mudando inclusive a nossa nossa opinião, conforme pode ser visto nesse post e também no vídeo.

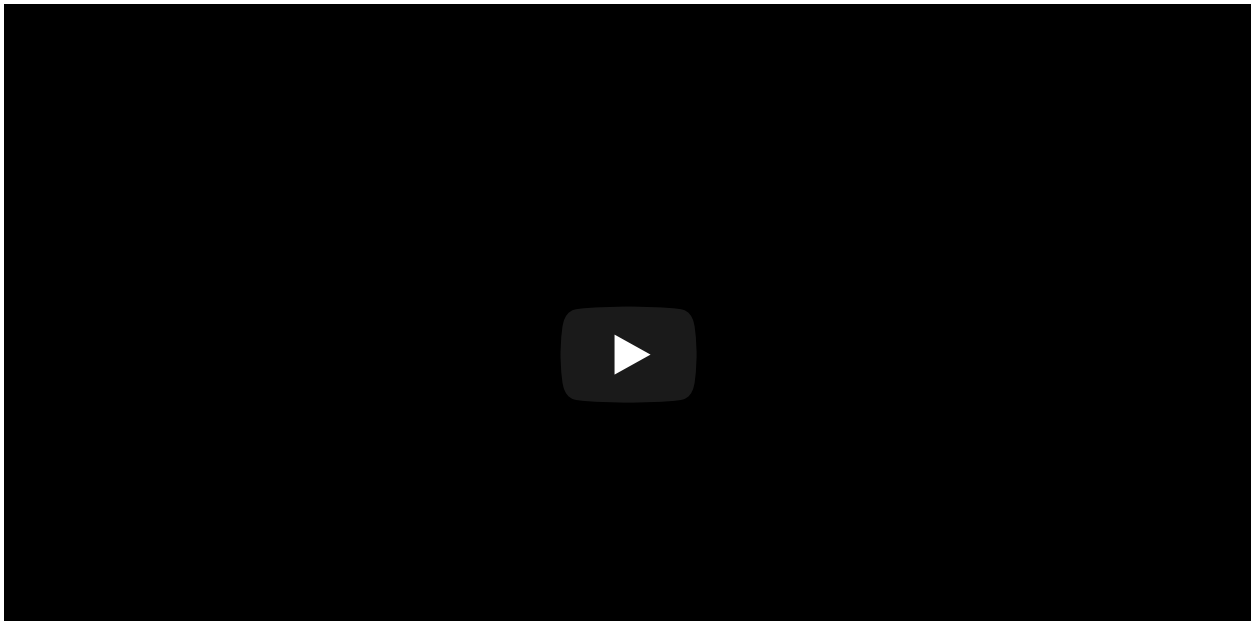
"**Início o projeto com Expo ou não?"**

A nova atualização do Expo SDK 38.0 trouxe muitos benefícios para a ferramenta. Agora com os recursos delas podemos dizer que na maioria dos casos iremos utilizar o expo nos projetos mobile com React Native. Criar um novo projeto com Expo, mas sem as limitações utilizando o **create-react-native-app**. 👍

A única coisa negativa, no entanto, é o fato do bundle ainda estar pesado. 👎

Veja mais um benefício do expo no projeto.

O **Diego Fernandes** fez um **Code/Drops** falando sobre o lançamento do SDK 38 e a evolução do Expo, confira:



E aí, o que achou desse post? Curtiu a SDK 38 do expo? Teve alguns *insights* legais? Compartilhe com a gente!

✔ Links citados:

- https://docs.expo.io/bare/updating-your-app/
- https://docs.expo.io/versions/v37.0.0/react-native/typescript/
- https://blog.expo.io/expo-sdk-38-is-now-available-ab6cd30ca2ee
- https://blog.expo.io/time-to-start-using-expos-bare-workflow-expokit-now-deprecated-d6052890c18b
- https://docs.expo.io/bare/installing-unimodules/
- https://github.com/expo/create-react-native-app

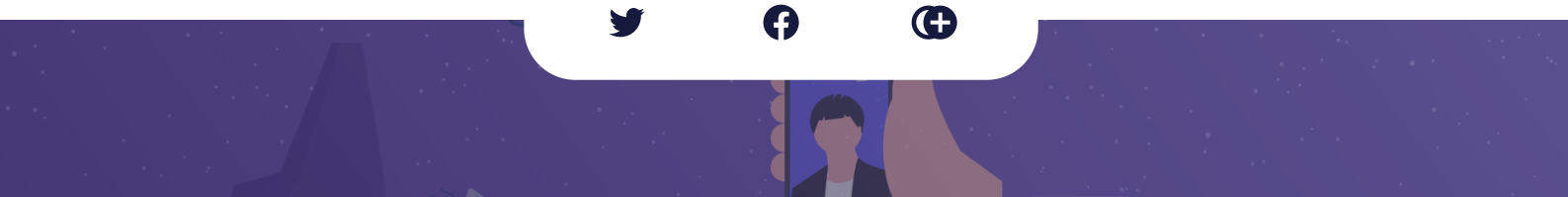
Espero que tenha curtido! 🤍

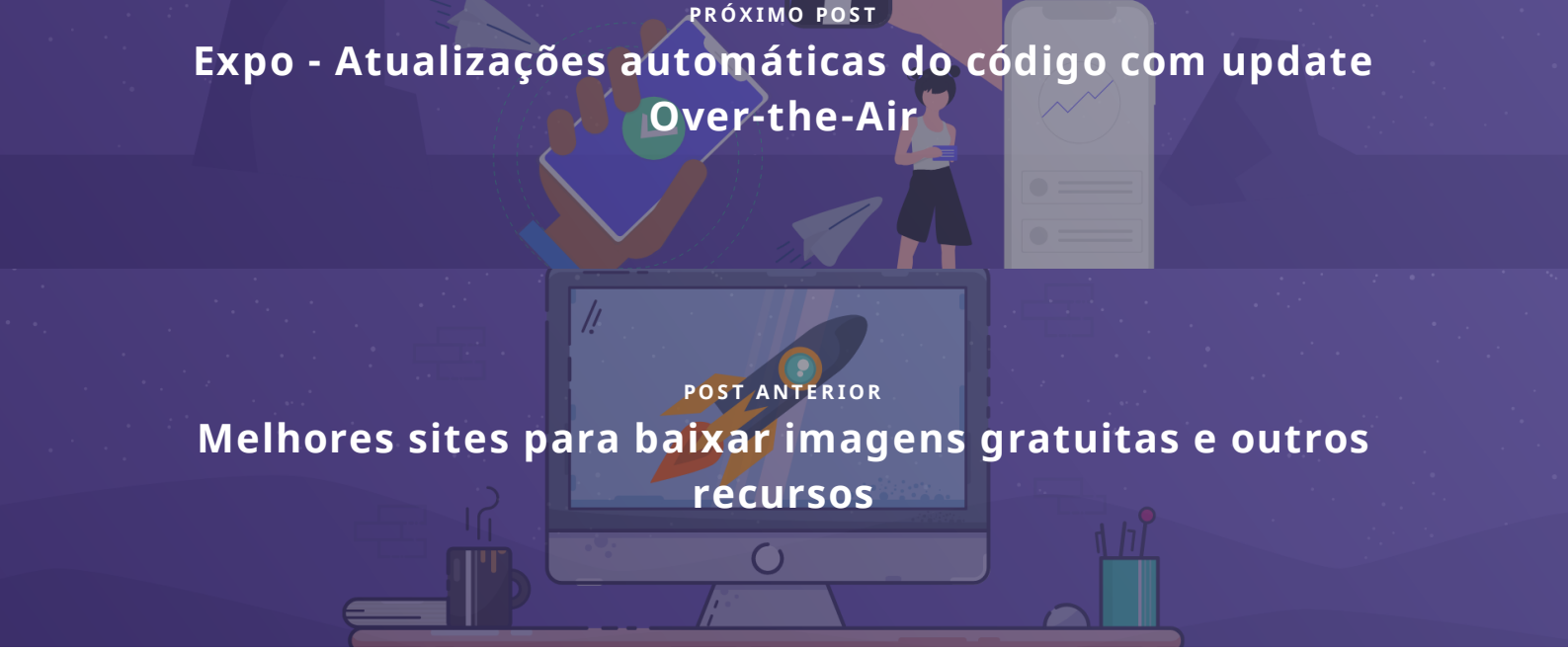
O aprendizado é contínuo e sempre haverá um próximo nível! 🚀

READ MORE POSTS BY THIS AUTHOR

Thiago Marinho

Dev Apaixonado por Tecnologia & Educação! Evolua rápido como a tecnologia, aprendizado é contínuo e sempre haverá um próximo nível. Boost Yourself e conte conosco! 🚀 🧑‍🚀 🧑‍🚀





PRÓXIMO POST

Expo - Atualizações automáticas do código com update Over-the-Air

POST ANTERIOR

Melhores sites para baixar imagens gratuitas e outros recursos