



# React Native

Professor: Crispim Luiz Martins

# O que é?

- React Native não é uma linguagem exclusiva para Android. É um framework que utiliza JavaScript e permite criar aplicativos móveis multiplataforma, ou seja, tanto para Android quanto para iOS, utilizando o mesmo código base. Com React Native, você pode desenvolver aplicativos que oferecem uma experiência nativa em ambas as plataformas, sem precisar criar dois códigos separados.

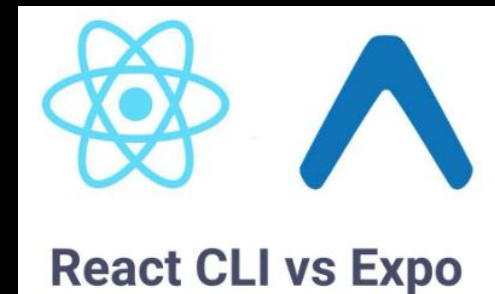


# Ferramentas



- Expo: É uma opção fantástica para iniciantes e projetos mais simples. Ele abstrai grande parte da configuração, tornando o processo mais rápido e amigável. O Expo funciona bem tanto no Android quanto no iOS.
- CLI do React Native (React Native CLI): Se você precisa de mais personalização ou está trabalhando em um projeto mais complexo, usar o CLI oficial do React Native é uma boa ideia. Com isso, você pode integrar diretamente com o Android Studio ou Xcode, mas também pode usar ferramentas alternativas.

# Vantagens do Expo em ambientes profissionais:



- Rapidez no desenvolvimento: O Expo facilita o processo inicial, economizando tempo.
- Compatibilidade multiplataforma: Oferece suporte integrado para Android e iOS.
- Biblioteca extensa: Inclui várias APIs, como câmera, localização e notificações, que agilizam o desenvolvimento.
- Facilidade de atualização: Projetos com Expo podem ser atualizados sem precisar recompilar o aplicativo.

## Limitações:

- Integrações nativas avançadas: Caso você precise de funcionalidades muito específicas ou bibliotecas nativas não suportadas pelo Expo, pode ser necessário ejetar o projeto do Expo.
- Flexibilidade: Em alguns casos, o React Native CLI (sem Expo) pode oferecer mais liberdade para personalizar e ajustar o comportamento do aplicativo.

# React Native CLI

- Com configuração personalizada é uma escolha melhor. Com ele, você tem acesso total ao código nativo e pode integrar bibliotecas e funcionalidades avançadas que não estão disponíveis no ambiente restrito do Expo.

Aqui estão os benefícios do React Native CLI para projetos sem essas limitações:

- **Acesso completo ao código nativo:** Você pode customizar partes específicas do aplicativo, tanto para Android quanto para iOS.
- **Liberdade para adicionar bibliotecas avançadas:** Se precisar de algo altamente específico (como integrar SDKs de terceiros), o CLI permite uma configuração livre.
- **Controle sobre o processo de build:** Você pode usar ferramentas como Android Studio e Xcode diretamente para compilar, ajustar e depurar seu aplicativo.

# Alertas, sempre use as bibliotecas da versão e do que está usando.

- Biblioteca Typescript = [TypeScript Tutorial](#)
- Biblioteca Expo = <https://docs.expo.dev/>
- Exemplo na instalação vou decidir o que quero trabalhar tirando do = [Install Expo Router - Expo Documentation](#)

**Quick start**

1 We recommend creating a new Expo app using `create-expo-app` to create a project with Expo Router library already installed and configured:

```
Terminal Copy  
- npx create-expo-app@latest
```

*Terá mais coisas, espere o próximo slide*

2 Now, you can start your project by running:

```
Terminal Copy  
- npx expo start
```

# Instalação Expo

```
C:\Users\crisp>node -v  
v22.14.0
```

```
C:\Users\crisp>npm -v  
10.9.2
```

- Verifique se tem o node.js com o comando node -v
- Para baixar [Node.js — Executar a JavaScript em Toda Parte](#)
- Instale o EXPO de o comando no terminal: npm install -g expo-cli
- Vá para a pasta onde coloca seus programas e de o comando =
- npx create-expo-app@latest --template blank-typescript
- Vai perguntar nome – coloque imepac-seu nome, exemplo: **imepac-crispim**

```
PS D:\Documents\Projetos-Programas> npx create-expo-app@latest --template blank-typescript  
Creating an Expo project using the blank-typescript template.
```

```
✓ What is your app named? ... imepac-crispim
```



```
✓ Downloaded and extracted project files.
```

```
> npm install
```

```
||
```

# Para começar a aplicação

- No terminal de o comando para inicializar = **npx expo start**



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays the project structure for 'MOBILE\_IMEPAC', including folders like 'app', 'assets', 'components', 'constants', 'hooks', 'node\_modules', 'scripts', and files like '.gitignore', 'app.json', 'package-lock.json', 'package.json', 'README.md', and 'tsconfig.json'. The main editor area shows the terminal output of the command 'npx expo start'. The output indicates that the server has stopped and the project is starting at 'D:\Documents\Projetos-Programas\mobile\_imepac'. A QR code is displayed in the terminal, which is used to scan and download the application. To the right of the QR code, text explains that after starting the project, the user will have the option to use it on a mobile device and should download the application in the next slide.

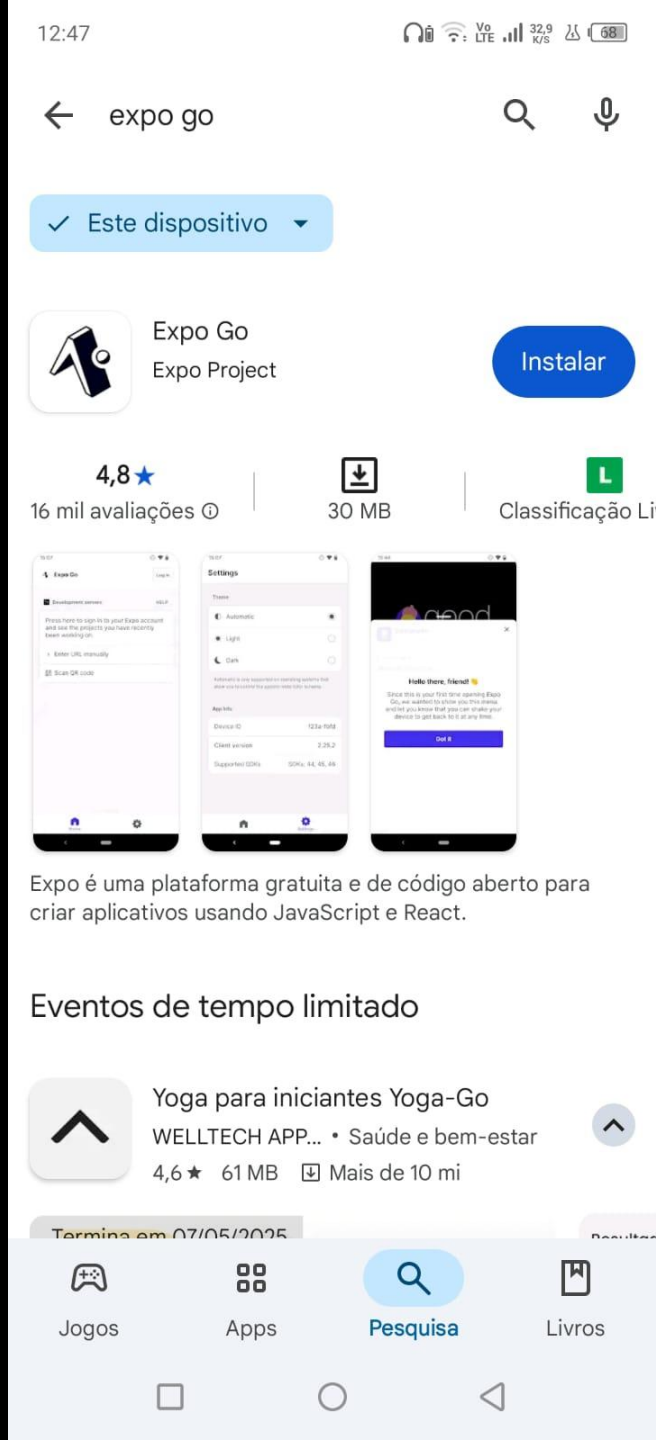
```
> Stopped server
PS D:\Documents\Projetos-Programas\mobile_imepac> npx expo start
Starting project at D:\Documents\Projetos-Programas\mobile_imepac
Starting Metro Bundler
```

Ao dar start o projeto terá a opção de usar no Celular.  
Baixe o aplicativo – Próximo Slide

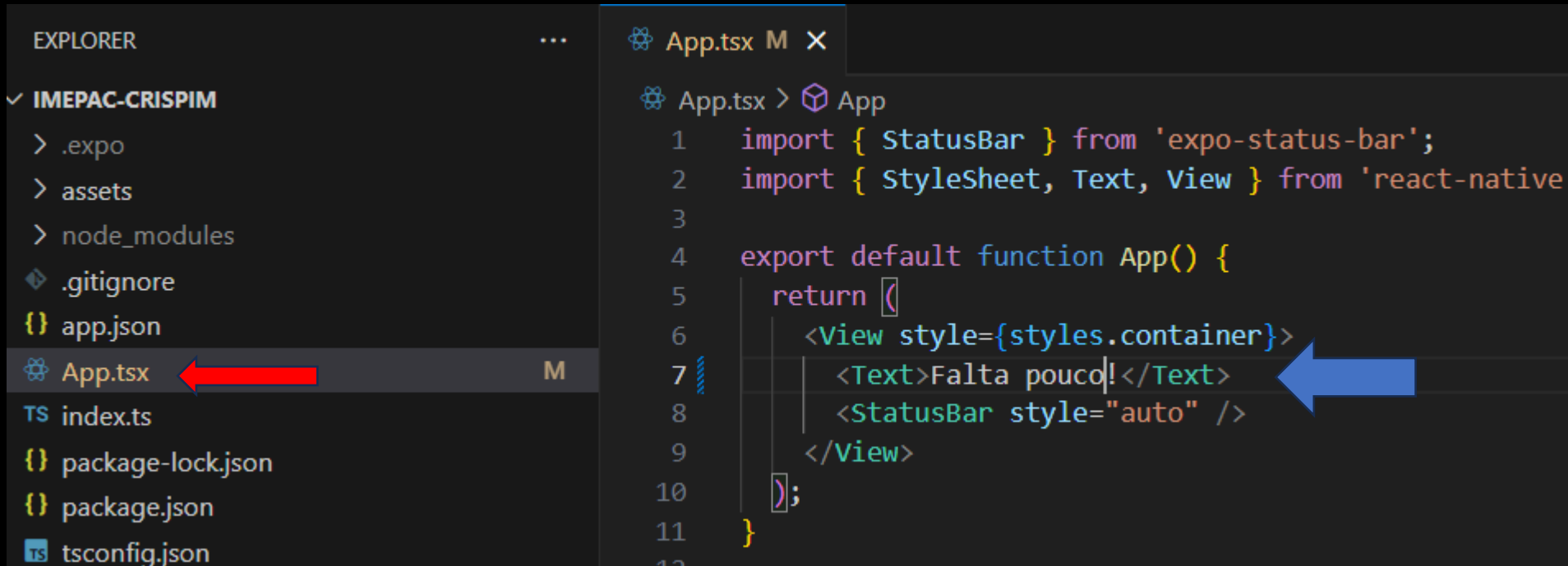


Abrir a aplicação

Baixe o aplicativo no  
seu celular e se  
estiverem na mesma  
rede verá as  
atualizações em  
tempo real



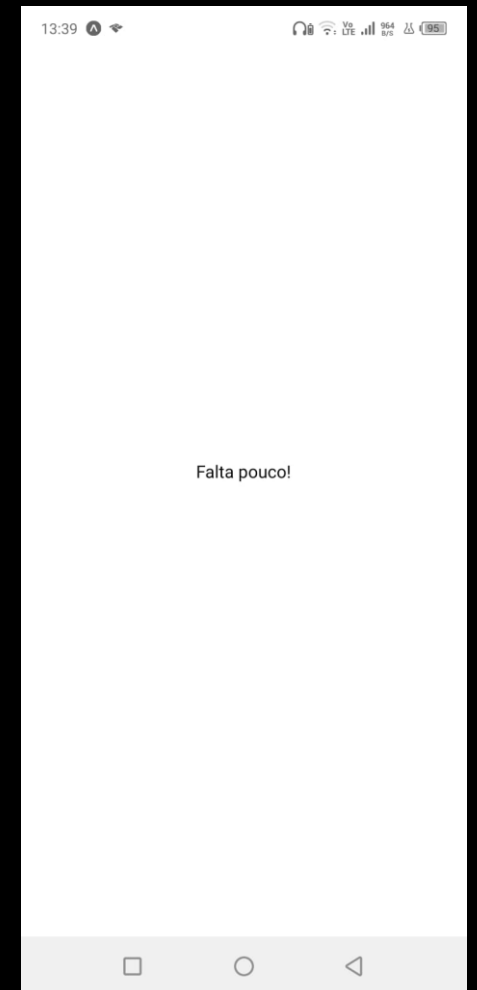
# Bora testar!!!



The image shows a VS Code editor interface. On the left, the Explorer sidebar shows a project named 'IMEPAC-CRISPIM' with files like .expo, assets, node\_modules, .gitignore, app.json, App.tsx (highlighted with a red arrow), index.ts, package-lock.json, package.json, and tsconfig.json. The main editor area shows the App.tsx file with the following code:

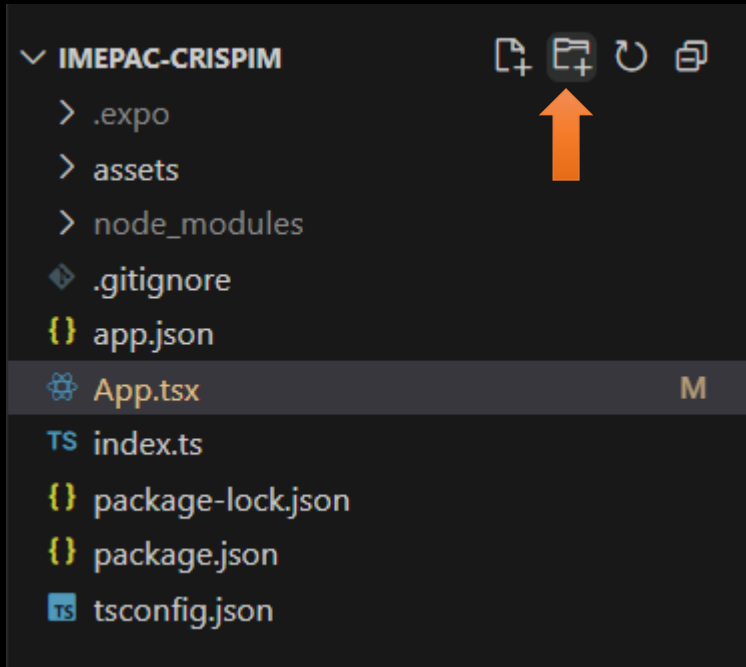
```
1 import { StatusBar } from 'expo-status-bar';
2 import { StyleSheet, Text, View } from 'react-native'
3
4 export default function App() {
5   return (
6     <View style={styles.container}>
7       <Text>Falta pouco!</Text>
8       <StatusBar style="auto" />
9     </View>
10   );
11 }
```

A blue arrow points to the text 'Falta pouco!' in the code.

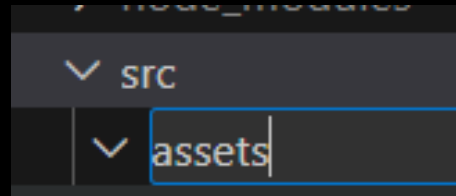


Na tela do celular irá  
mudar

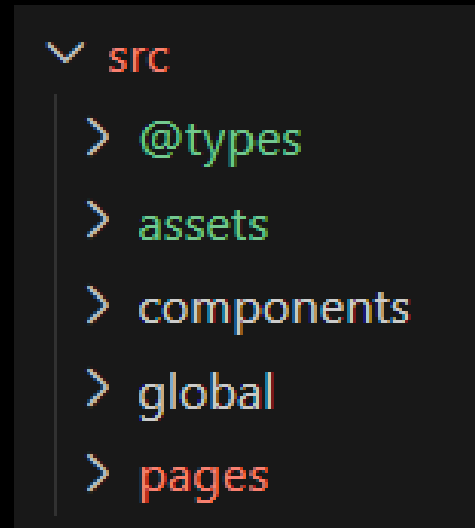
# Configurar o projeto



New folder de o nome de src



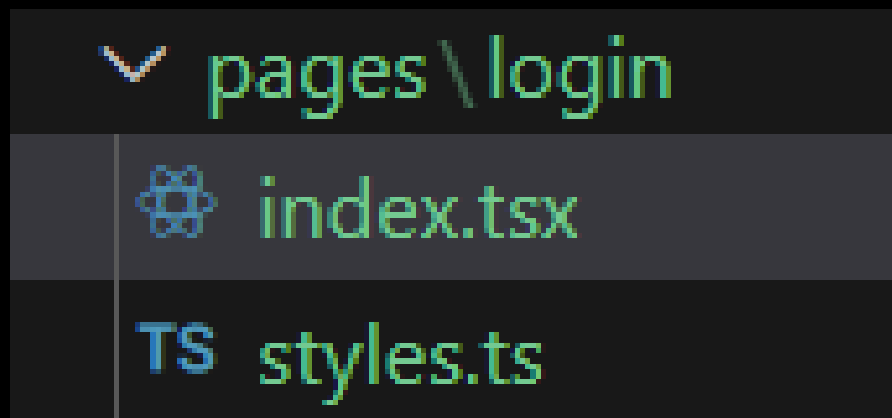
Dentro de src crie  
outra pasta assets



Dentre de src deve ter @types,  
assets, componentes, global,  
pages

# Primeira página

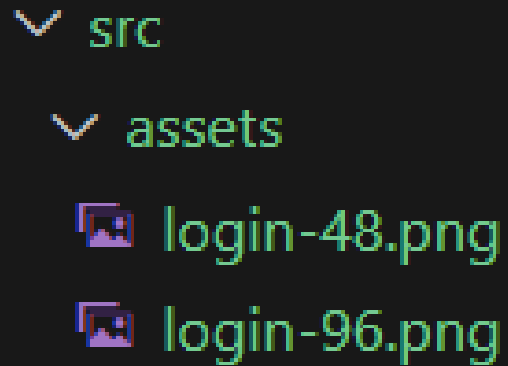
- → dentro de pages crie uma pasta com o nome de login
- Dentro dessa pasta login será onde fica os arquivos da página que pode ser o index.tsx e o styles.ts



Se precisar de imagens  
ícones pegue no link.

[Login Icons, Logos, Symbols – Free Download PNG, SVG](#)

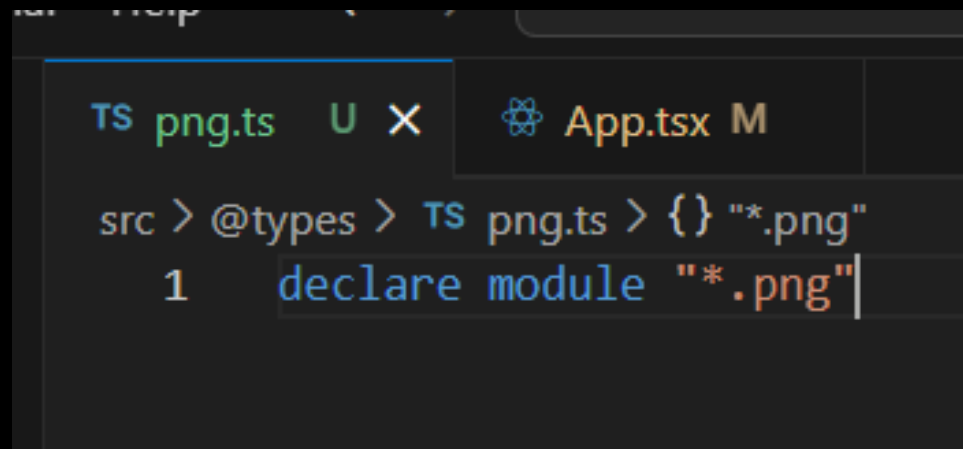
# Guarde suas imagens no Assets



```
src
└─ assets
   ├── login-48.png
   └── login-96.png
```

Pode ter subpastas para icons, img, etc...

Na pasta @types crie uma página png.ts – isso é para evitar erros de renderizar as imagens



```
src > @types > TS png.ts > {} "*.png"
1 declare module "*.png"
```

Se tiver outros tipos de extensão de imagens coloque também

# Página Login

```
index.tsx U X TS styles.ts U App.tsx M
src > pages > login > index.tsx > ...
1  import React from "react";
2  import {
3    Text
4  } from 'react-native';
5
6  export default function Login(){
7
8    return(
9      <Text>Olá aluno!!!</Text>
10
11    )
12  }
```

Ambos os Text tem o T em maiúsculo

Mas para funcionar  
devemos mudar o  
App.tsx

```
App.tsx > styles
1  import { StatusBar } from 'expo-status-bar';
2  import { StyleSheet, Text, View } from 'react-native';
3  import Login from './src/pages/login';
4
5  export default function App() {
6    return (
7      <View style={styles.container}>
8        <StatusBar style="auto" />
9        <Login />
10      </View>
11    );
12  }
```

Atualize e olhe no celular

Se não atualizar verifique o console de erros!!!!

# Bora para começar um projeto!!!

- Temos nosso arquivo `index.tsx`, esse será nosso esqueleto do projeto. Ele deve ter várias conexões, mas aqui faremos somente nossa tela com a conexão com o estilo.
- -----
- Temos um arquivo dentro da pasta `login` de nome `styles.ts`
- Lá colocaremos todos os nossos estilos.
- Colocarei a imagem inteira, mas lembre que temos que ter um conexão com nosso arquivo `index.tsx`

TS styles.ts U

index.tsx U X

src &gt; pages &gt; login &gt; index.tsx &gt; ...

```
1 import React from "react";
2 import {
3   Text,
4   View,
5   Image,
6   TextInput
7 } from 'react-native';
8 import {style} from './styles';
9 import Logo from '../../assets/login-96.png';
10 export default function Login(){
11   return(
12     <View style={style.container}>
13       <View style={style.boxTop}>
14         <Image source={Logo} />
15         <Text>Bem Vindo!!!</Text>
16       </View>
17       <View style={style.boxMid}>
18         <Text>Endereço de E-mail</Text>
19         <TextInput style={style.boxInput}/>
20         <Text>Coloque sua Senha</Text>
21         <TextInput style={style.boxInput}/>
22       </View>
23       <View style={style.boxBottom}>
24         <Text>Bottom</Text>
25       </View>
26     </View>
27   )
28 }
```

TS styles.ts U X

index.tsx U

src &gt; pages &gt; login &gt; TS styles.ts &gt; style &gt; boxBottom

```
1 import { Dimensions, StyleSheet } from "react-native";
2
3 export const style = StyleSheet.create({
4   container:{
5     flex:1,
6     width:"100%",
7     paddingTop:50,
8     justifyContent:"center",
9     backgroundColor:"#a4c639",
10  },
11  boxTop:{
12    alignItems: 'center',
13    justifyContent: 'center',
14    height:Dimensions.get("window").height/3,
15    width:"100%"
16  },
17  boxMid:{
18    height:Dimensions.get("window").height/3,
19    width:"100%",
20    marginLeft: 20,
21  },
22  boxBottom:[]
23    alignItems: 'center',
24    height:Dimensions.get("window").height/3,
25    width:"100%"
26  ],
27  boxInput:{
28    backgroundColor:"#Ffffff",
29    marginRight: 40,
30  }
31 })
```



# Resultado

