

PM I - 2023-1 - Parte 2

Crie uma pasta no drive c chamada pml

Abra o terminal do windows dentro da pasta pml

Digite o comando npm i -g create-expo-app

digite o comando npx create-expo-app myapp5p --template para criar o primeiro projeto em React Native. Ao digitar o comando será exibida a janela abaixo.

```
Air-de-Luiz:preparar_aulla luizclaudio$ npx create-expo-app myapp5p --template
? Choose a template: > - Use arrow-keys. Return to submit.
  Blank
> Blank (TypeScript) - blank app with TypeScript enabled
    Navigation (TypeScript)
    Blank (Bare)
```

Escolha a opção Blank (TypeScript) e será exibida a janela abaixo

```
Air-de-Luiz:preparar_aulla luizclaudio$ npx create-expo-app myapp5p --template
✓ Choose a template: > Blank (TypeScript)
✓ Downloaded and extracted project files.
> npm install
█
```

O projeto foi criado, aguarde por alguns minutos até que os pacotes sejam instalados.

Se eles não forem instalados, automaticamente. Pressione as teclas ctrl + c para cancelar e voltar o prompt no cmd. Entre na pasta myapp5p digitando o comando cd myapp5p e digite o comando npm install para instalar os pacotes. Veja abaixo o print.

```
Air-de-Luiz:preparar_aulla luizclaudio$ cd myapp5p/
Air-de-Luiz:myapp5p luizclaudio$ npm install
(( [REDACTED] )) " idealTree:tempy: sill placeDep node_modules/@expo/cli ty
```

A janela abaixo mostra que os pacotes foram instalados

```
Air-de-Luiz:preparar_aulla luizclaudio$ cd myapp5p/
Air-de-Luiz:myapp5p luizclaudio$ npm install
npm WARN deprecated @npmcli/move-file@1.1.2: This functionality has been moved to @npmcli/fs
npm WARN deprecated source-map-url@0.4.1: See https://github.com/lydell/source-map-url#deprecated
npm WARN deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprecated
npm WARN deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#deprecated
npm WARN deprecated source-map-resolve@0.5.3: See https://github.com/lydell/source-map-resolve#deprecated
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random
  circumstances, which is known to be problematic. See https://v8.dev/blog/math-random for details.
npm WARN deprecated uglify-es@3.3.9: support for ECMAScript is superseded by `uglify-js` as of v3.13.0

added 1660 packages, removed 995 packages, changed 164 packages, and audited 1830 packages in 46s

61 packages are looking for funding
  run `npm fund` for details

5 high severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

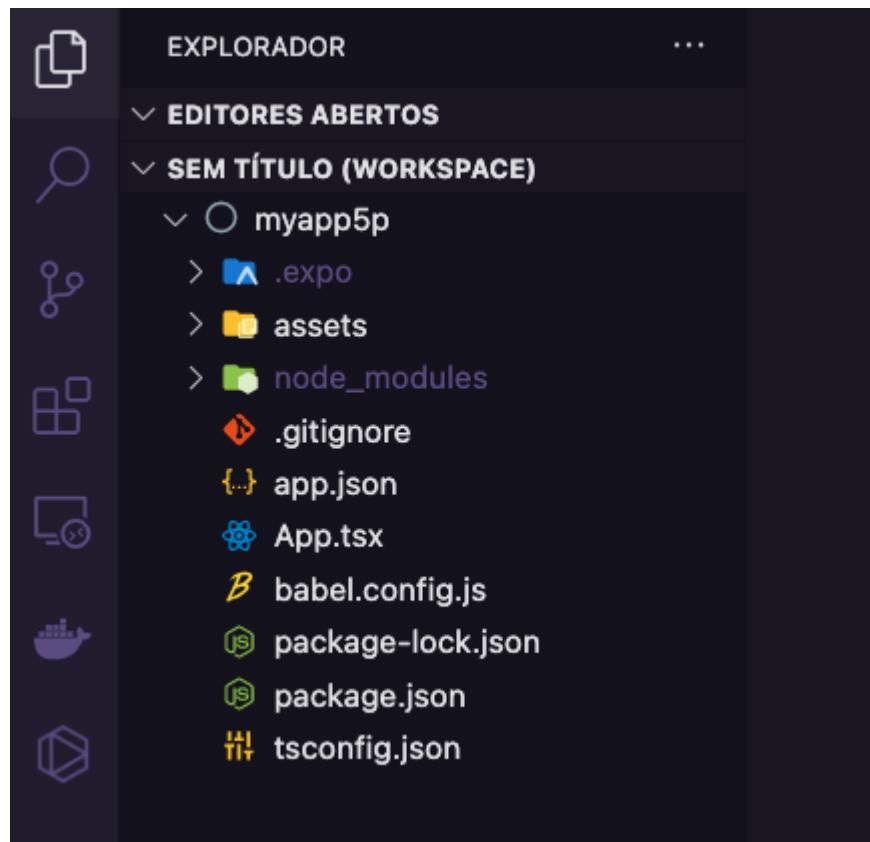
Run `npm audit` for details.
Air-de-Luiz:myapp5p luizclaudio$ █
```

Pontos de atenção

- a) No caminho do usuário e nome da pasta não utilize caracteres especiais. (ex: :/Área de Trabalho/projeto). **Evite criar projetos na área de trabalho.**
- b) **Evite também espaços em branco. Evite caracteres especiais**, porque sempre quebra o build do Gradle no Android. O local mais comum de acontecer isso é no Windows quando vc cria na área de trabalho ou possui um nome com acento e por algum motivo o Windows buga e cria o usuário com acento. (ex.: c:/Users/João). **Se for nome composto utilize underline (_)**
- c) Não crie o projeto dentro de pastas sincronizadas (ex: OneDrive, Google Drive) porque isso acabe quebrando o código ou impedindo os scripts de executar. **É para isso que tem o GitHub para manter o código num local seguro.**

Estruturas de pastas de um projeto

Abra o VsCode e adicione o seu projeto



.expo - Armazena informações compartilhada do expo

assets - Armazena arquivos de imagens, visuais que serão utilizados na aplicação, por exemplo, um ícone de exemplo, um ícone de splash. A tela que fica carregando.

node_modules - contém todas as bibliotecas instaladas na aplicação. Não precisa se preocupar com ela. Se for apagada por algum motivo e caso queira retornar, digite o comando **npm i** ou **npm install**. Ela é gerada novamente porque tem o arquivo package.json que armazena as dependências do seu projeto.

package.json -

```
⚙️ package.json myapp5p ×
● myapp5p > ⚙️ package.json > ...
8   "ios": "expo start --ios",
9   "web": "expo start --web"
10 },
11   "dependencies": {
12     "expo": "~47.0.12",
13     "expo-status-bar": "~1.4.2",
14     "react": "18.1.0",
15     "react-dom": "18.1.0",
16     "react-native": "0.70.5",
17     "react-native-web": "~0.18.9"
18 },
19   "devDependencies": {
20     "@babel/core": "^7.12.9",
21     "@types/react": "~18.0.14",
22     "@types/react-native": "~0.70.6",
23     "typescript": "^4.6.3"
24 },
25   "private": true
26 }
```

package-lock.json - É um arquivo de cache dos pacotes que são instalados no seu projeto.

.gitignore - Contém uma lista de pastas e arquivos que devem ser ignorados pelo controle de versão do GitHub incluindo a pasta node_modules

```
❖ .gitignore myapp5p ×  
● myapp5p > ❖ .gitignore  
1   node_modules/  
2   .expo/  
3   dist/  
4   npm-debug.*  
5   *.jks  
6   *.p8  
7   *.p12  
8   *.key  
9   *.mobileprovision  
10  *.orig.*  
11  web-build/  
12  
13  # macOS  
14  .DS_Store
```

app.json - Contém informações da aplicação como o nome e versão.

```
..} app.json myapp5p X
● myapp5p > ..} app.json > ...
1  {
2    "expo": {
3      "name": "myapp5p",
4      "slug": "myapp5p",
5      "version": "1.0.0",
6      "orientation": "portrait",
7      "icon": "./assets/icon.png",
8      "userInterfaceStyle": "light",
9      "splash": {
10        "image": "./assets/splash.png",
11        "resizeMode": "contain",
12        "backgroundColor": "#fffffff"
13      },
14      "updates": {
15        "fallbackToCacheTimeout": 0
16      },

```

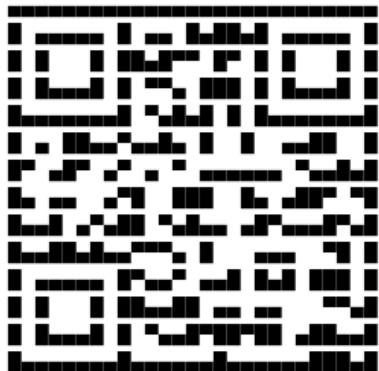
tsconfig.json - É o arquivo que vc coloca as configurações de como o TypeScript deve se comportar.

App.tsx - É o arquivo de entrada (inicialização) do seu projeto.

Executando o projeto

Dentro da pasta myapp5p digite o comando **npx expo start** para iniciar o projeto e será exibida a janela abaixo.

Starting Metro Bundler



```
> Metro waiting on exp://10.0.0.205:19000
> Scan the QR code above with Expo Go (Android) or the Camera app (iOS)

> Press a | open Android
> Press i | open iOS simulator
> Press w | open web

> Press j | open debugger
> Press r | reload app
> Press m | toggle menu

> Press ? | show all commands
```

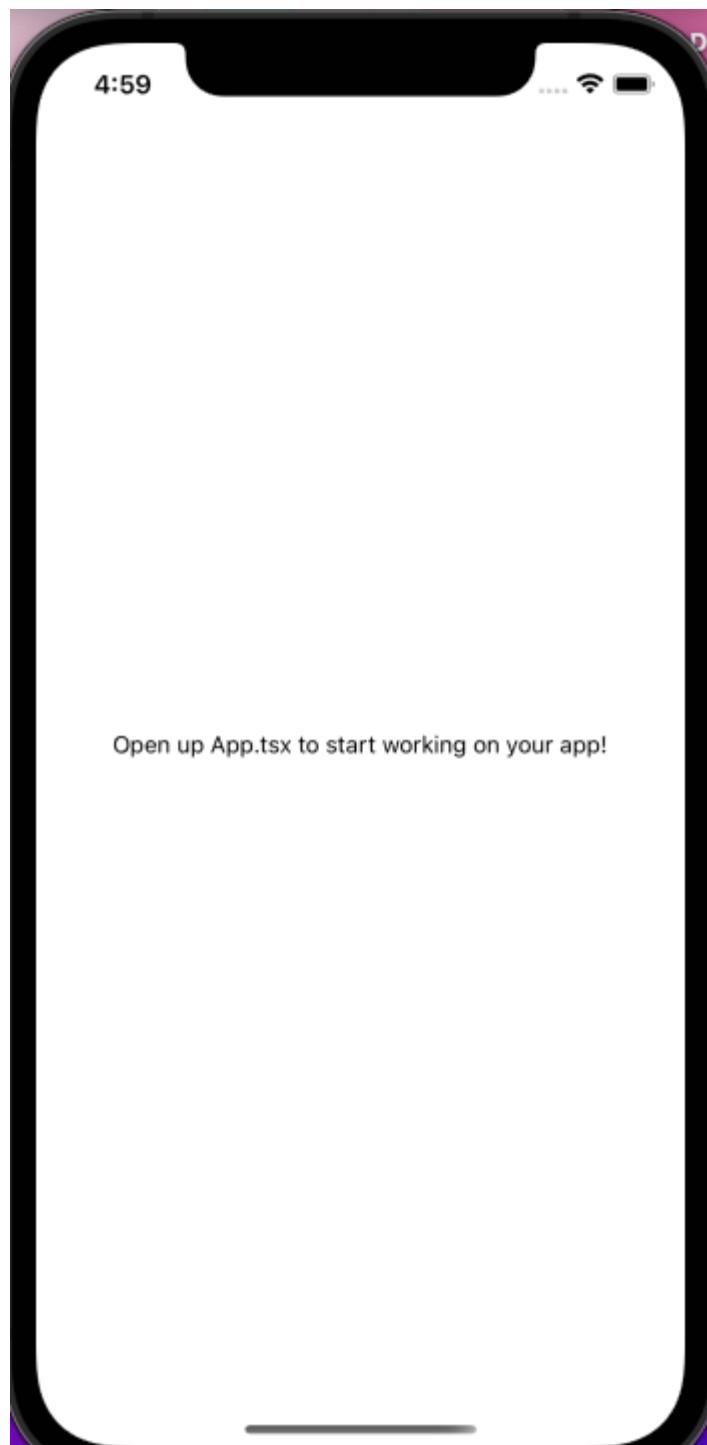
Logs for your project will appear below. Press Ctrl+C to exit.

Com o aplicativo **Expo Go** instalado no seu smartphone ou iphone.

A screenshot of the Expo Go mobile application. At the top, there are download links for the App Store and Google Play. Below that, the main interface shows a "Development servers" section with a text input field containing "expo start". A note says "Select the local server when it appears here." Underneath, there's a "Projects" section listing two items: "Expo APIs" and "Bacon Buddy". Each project entry includes its name, icon, and a note about its SDK version. The "Expo APIs" entry notes "native-component-list" and "SDK 40: Not supported". The "Bacon Buddy" entry notes "bacon-buddy" and "SDK 37: Not supported".

abra-o e scaneia o Qr code para que o aplicativo seja iniciado. Veja abaixo o print.

Obs: O seu smartphone deve estar conectado na mesma rede (mesma faixa de ip do seu computador/notebook).



Você também tem as opções para abrir com o simulador do Android pressionando **letra a** ou simulado do IOS com **letra i** ou até mesmo abrir no navegador pressionando a **tecla w**. Quando for necessário reiniciar o app, pressione a **tecla r**. Para cancelar este menu pressione **ctrl + c**

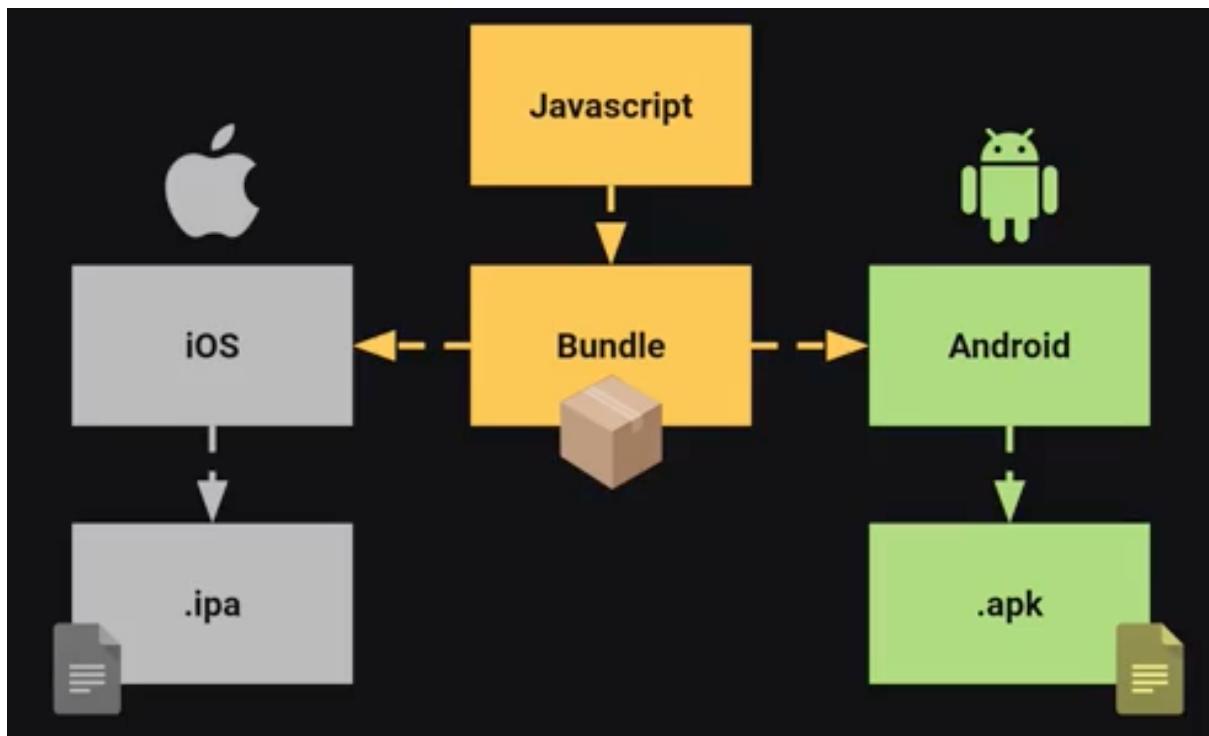
Funcionamento do React Native

Desenvolvimento tradicional, você desenvolve para os dois ambientes. IOS e Android.



Como funciona o React Native

Ele passa por um processo de empacotamento chamado Bundle. Ele pode ter um destino, ambiente IOS ou Android. Dentro do bundle ele tem toda a lógica e interfaces.

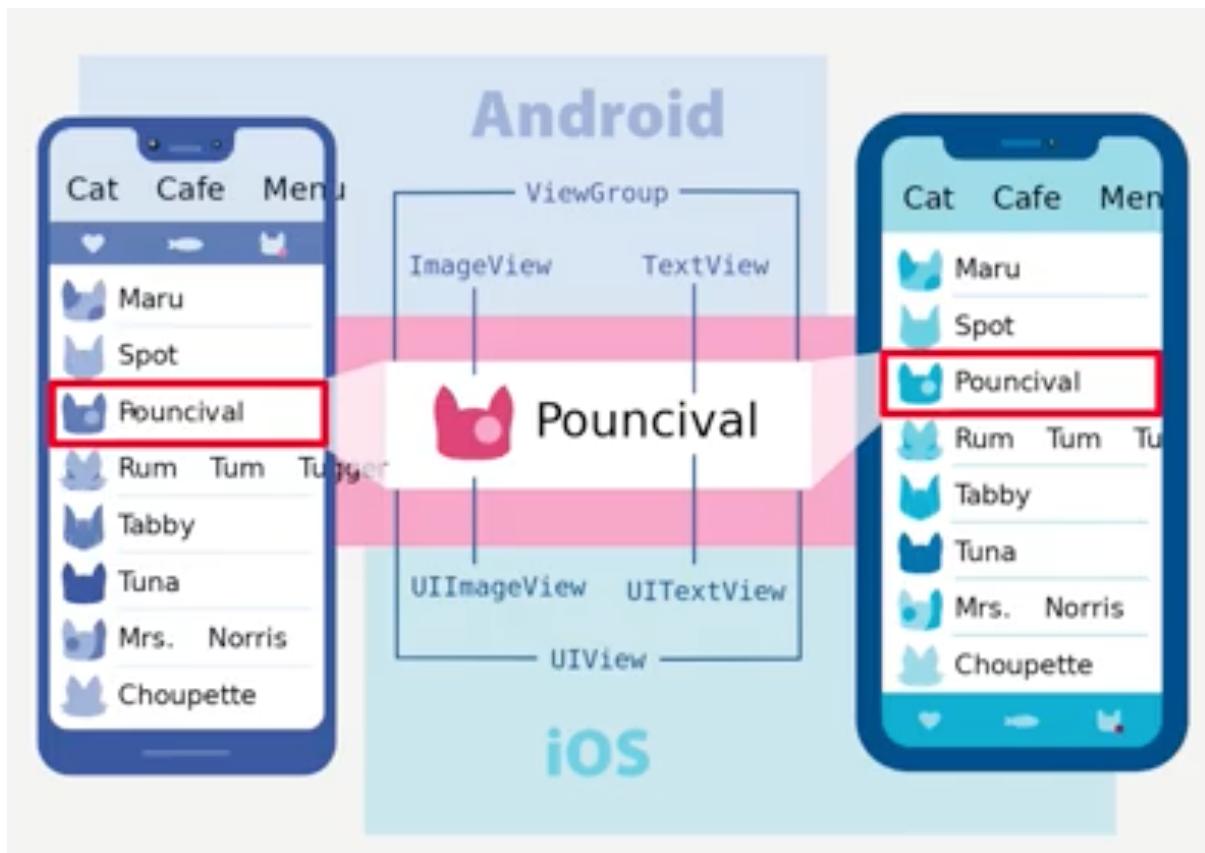


A interface é construída baseada em Interface declarativa com JSX.

Interface declarativa com JSX

O JSX é a sintaxe utilizada para desenhar a interface de forma declarativa

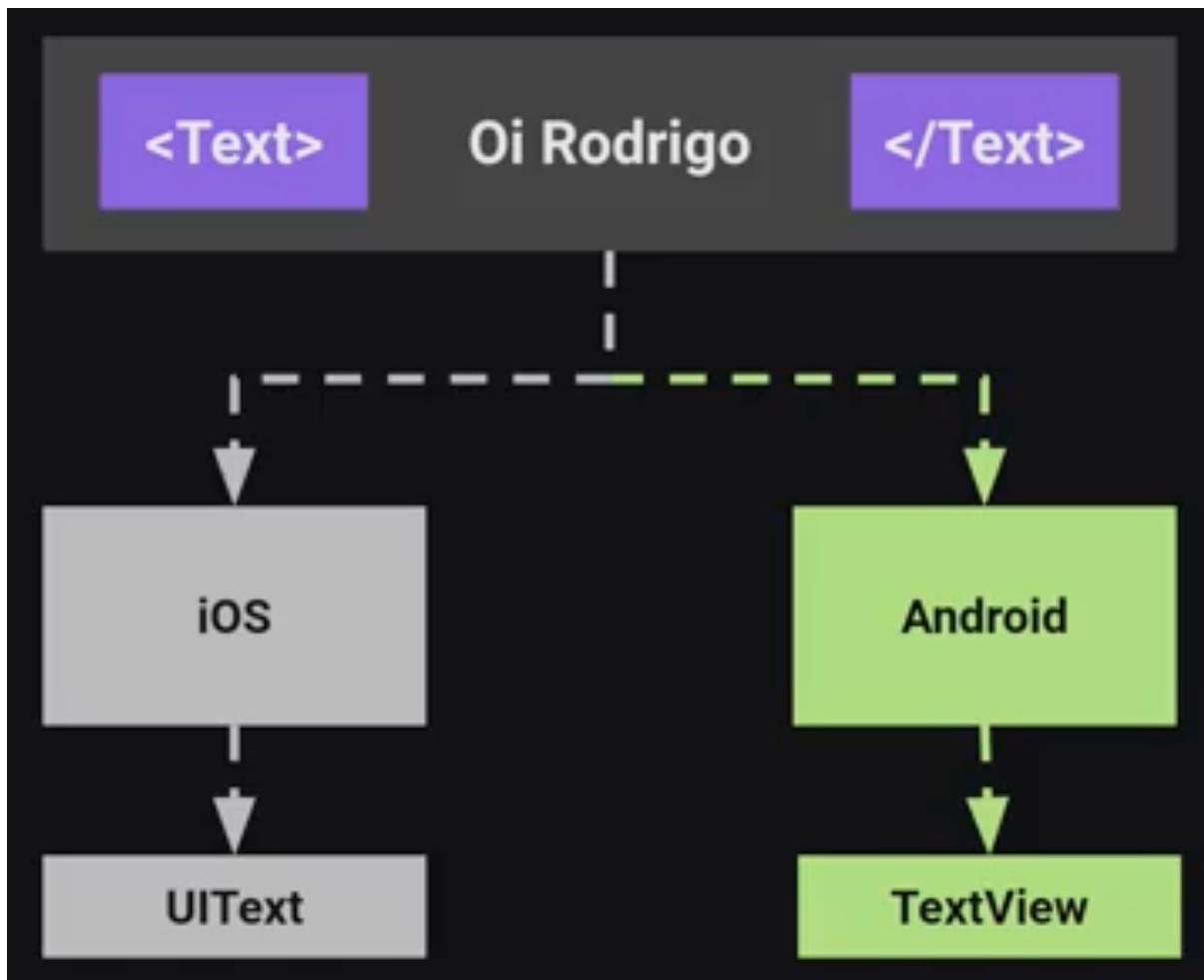
Por exemplo. Mesmo aplicativo sendo executado no Android ou IOS



No RN será utilizado Image para desenhar Imagem e Text para texto. No android ele será renderizada como ImageView e TextView.

No IOS será renderizado com o UIImageView e UITextView. Será declarado o que você quer desenvolver e eles serão renderizados no ambiente Android ou IOS.

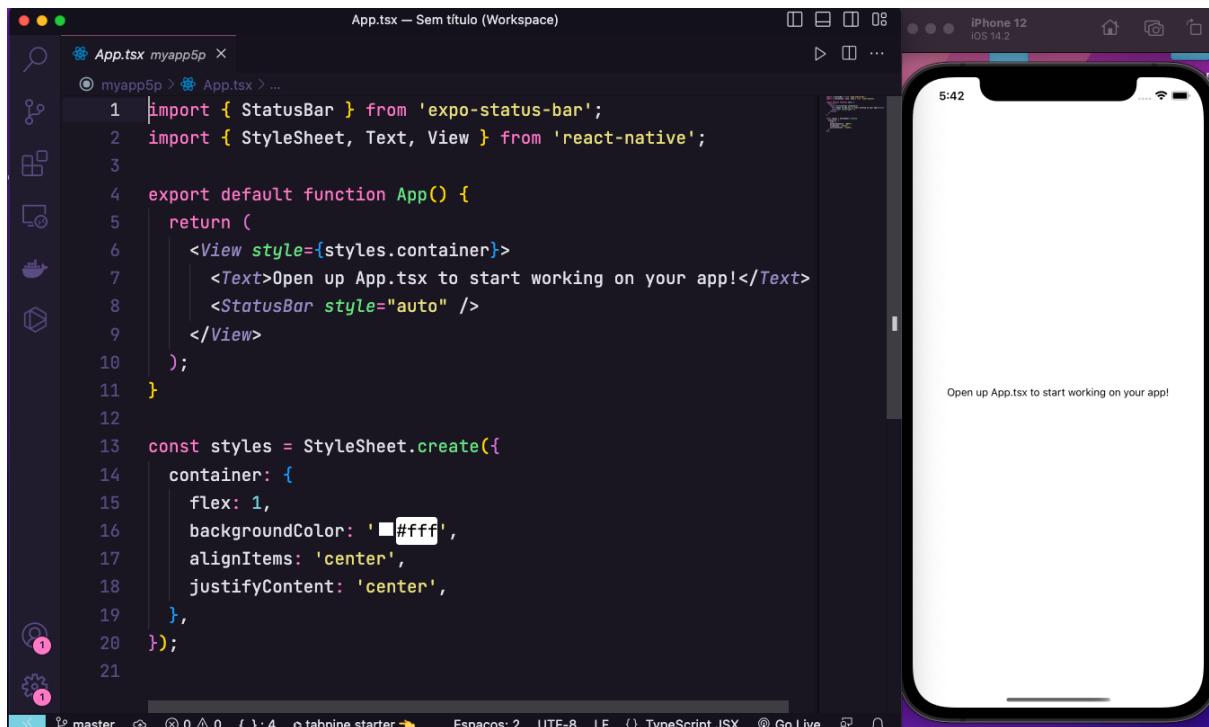
Por exemplo, no RN para definir um texto, vc utiliza o componente Text, isso é feito de forma declarativa. Quando ele for executado, será utilizado o componente específico o Android e o IOS. Veja abaixo o print.



Fast Refresh

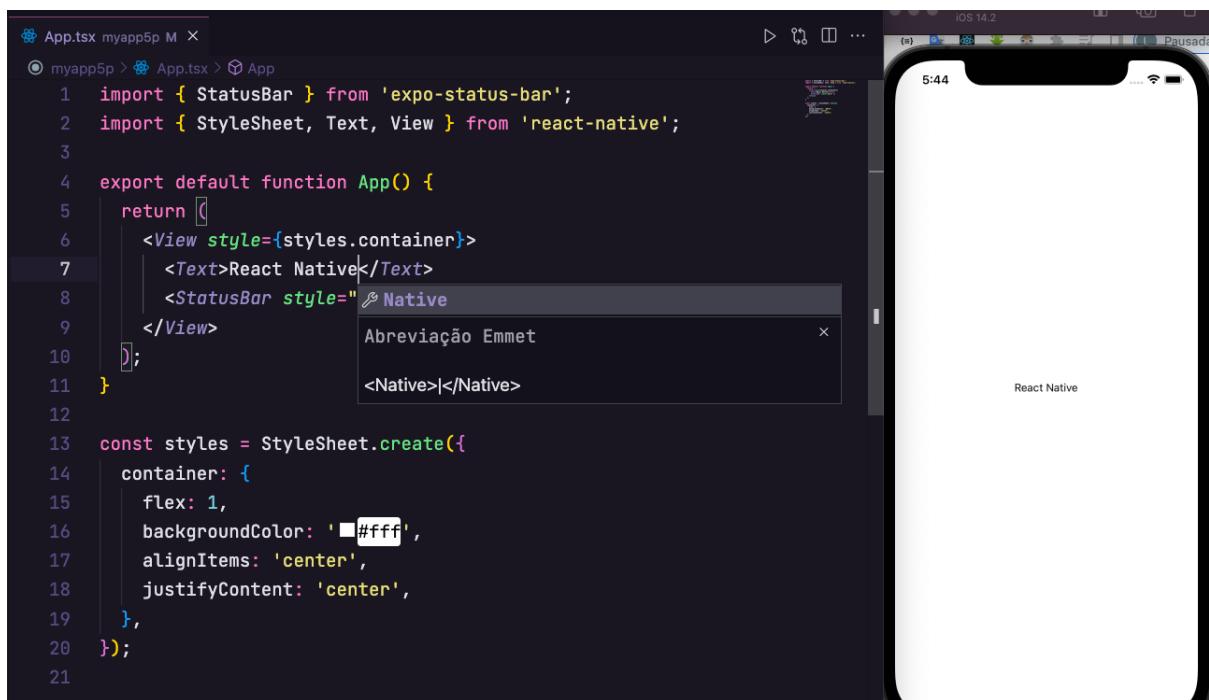
É o recurso que muda o conteúdo da sua tela automaticamente quando é feita uma alteração e a mesma é gravada.

Abra o arquivo App.tsx do seu projeto, conforme abaixo



```
App.tsx myapp5p X
myapp5p > App.tsx > App
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>Open up App.tsx to start working on your app!</Text>
8       <StatusBar style="auto" />
9     </View>
10  );
11}
12
13 const styles = StyleSheet.create({
14   container: {
15     flex: 1,
16     backgroundColor: '#ffff',
17     alignItems: 'center',
18     justifyContent: 'center',
19   },
20 });
21
```

Altere o texto da linha 7 para React Native e grave a alteração. Você verá que automaticamente texto foi alterado no seu smartphone ou iphone.



```
App.tsx myapp5p M
myapp5p > App.tsx > App
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>React Native</Text>
8       <StatusBar style="Native" />
9     </View>
10   ];
11}
12
13 const styles = StyleSheet.create({
14   container: {
15     flex: 1,
16     backgroundColor: '#ffff',
17     alignItems: 'center',
18     justifyContent: 'center',
19   },
20 });
21
```

Se por algum motivo vc quiser reiniciar o seu app, abra o terminal e pressione a letra r, ele vai recarregar o bundle e renderizar a tela novamente.

```
myapp5p — node -e npm exec expo start TERM_PROGRAM=Apple_Terminal ANDROID_HOME=/Users/luizc...
```

QR code

```
> Metro waiting on exp://10.0.0.205:19000
> Scan the QR code above with Expo Go (Android) or the Camera app (iOS)

> Press a | open Android
> Press i | open iOS simulator
> Press w | open web

> Press j | open debugger
> Press r | reload app
> Press m | toggle menu

> Press ? | show all commands

Logs for your project will appear below. Press Ctrl+C to exit.
> Opening on iOS...
> Opening exp://10.0.0.205:19000 on iPhone 12
> Press ? | show all commands
ios Bundling complete 53918ms
> Reloading apps
ios Bundling complete 365ms
```

React Native

Entendendo o JSX

O JSX é a sintaxe que o React utiliza para que a gente possa criar interfaces de forma declarativa. O React permite criar interfaces (web, mobile, watch).

O React Native disponibiliza os elementos específicos do contexto mobile para ser renderizado de forma nativa no Android ou iOS.

Na linha 7 está declarando o texto React Native, ele é muito parecido com o HTML, tem uma tag de abertura e fechamento.

```
App.tsx myapp5p M X

myapp5p > App.tsx > styles > container

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>React Native</Text>
8       <StatusBar style="auto" />
9     </View>
10   );
11 }
```

Apague o conteúdo do arquivo App.tsx para que a gente possa criá-lo do zero.

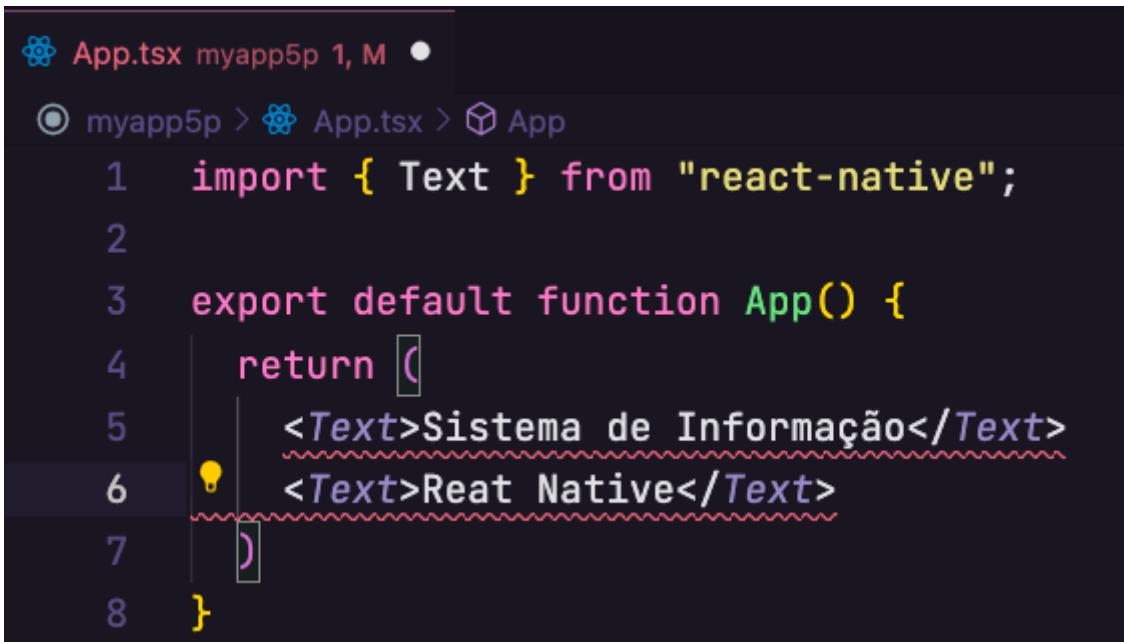
```
App.tsx myapp5p M X
myapp5p > App.tsx > ...
1 import { Text } from "react-native";
2
3 export default function App() {
4     return (
5         <Text>Sistema de Informação</Text>
6     )
7 }
```

Salve as alterações e veja o resultado conforme abaixo.



Observe que o texto ficou na parte superior do lado esquerdo, pois não tem nenhuma estilização.

Altere o App para inserir dois textos. Insira a linha 6 conforme abaixo.



```
App.tsx myapp5p 1, M ●
myapp5p > App.tsx > App
1 import { Text } from "react-native";
2
3 export default function App() {
4     return (
5         <Text>Sistema de Informação</Text>
6         <Text>React Native</Text>
7     )
8 }
```

Salve as alterações e veja o resultado.



Elementos JSX devem ser embrulhados em uma tag de fechamento. Você não pode retornar mais de um elemento. É uma regra do JSX. Uma interface não é feita de um elemento, ela pode ter vários, desde que vc retorne embrulhada por um elemento. Para resolver vc pode usar um fragment, que é um sinal de menor e maior para abertura e fechamento. (<> </>). Ela serve única e exclusivamente para embrulhar elementos. Insira as linhas 5 e 8.

```
⚙️ App.tsx myapp5p M X
● myapp5p > ⚙️ App.tsx > ...
1 import { Text } from "react-native";
2
3 export default function App() {
4     return (
5         <>
6             <Text>Sistema de Informação</Text>
7             <Text>React Native</Text>
8         </>
9     )
10 }
```

Salve as alterações e veja o resultado.



Observe que a aplicação voltou a funcionar.

Outra forma que você pode utilizar para embrulhar os elementos é utilizar uma View.

Altere a linha 1 para importar a View

Altere as linhas 5 e 9

```
⚙️ App.tsx myapp5p M ✘
● myapp5p > ⚙️ App.tsx > ...
1 import { Text, View } from "react-native";
2
3 export default function App() {
4   return (
5     <View>
6       <Text>Sistema de Informação</Text>
7       <Text>React Native</Text>
8     </View>
9   )
10 }
```

Salve e veja que o app continua funcionando da mesma forma.

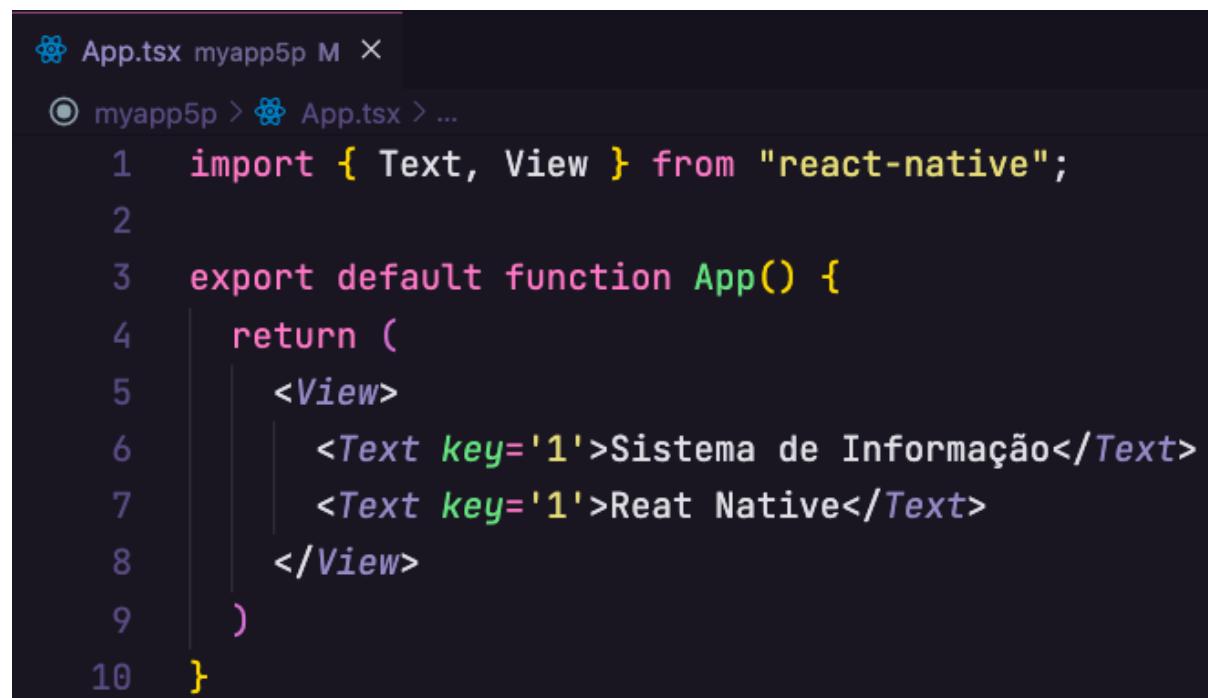


Entendendo a LogBox

Uma coisa que vc vai ver ao longo do desenvolvimento da sua aplicação são as LogBoxs. Elas serão exibidas na cor amarela (são erros não tratados e não quebram a aplicação) ou vermelha (são os erros que quebram a aplicação).

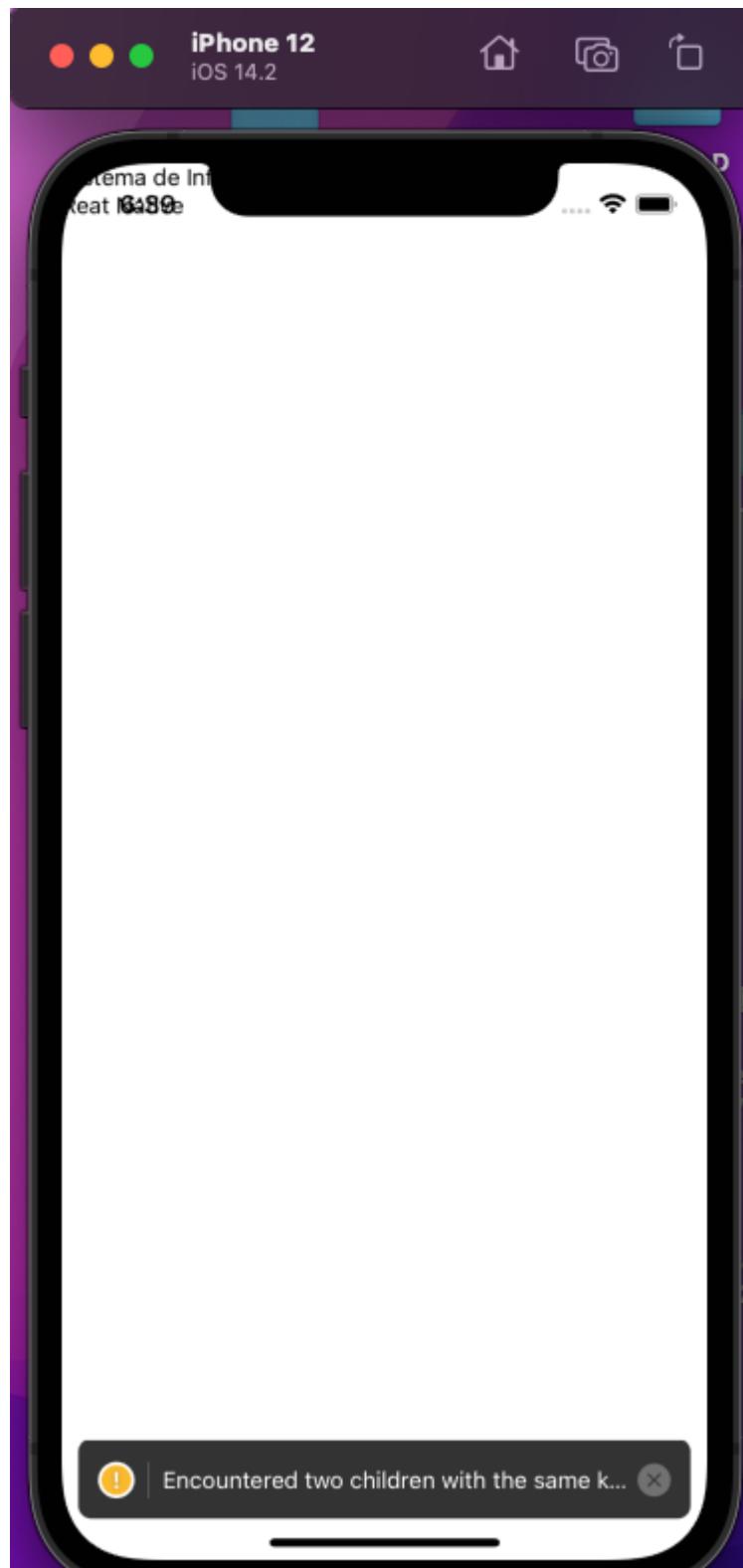
Simulando o LogBox - cor amarela - warning.

Altere as linhas 6 e 7 inserindo a propriedade key com o mesmo conteúdo, ou seja, a mesma chave, porém no RN cada elemento deve ter uma chave única.

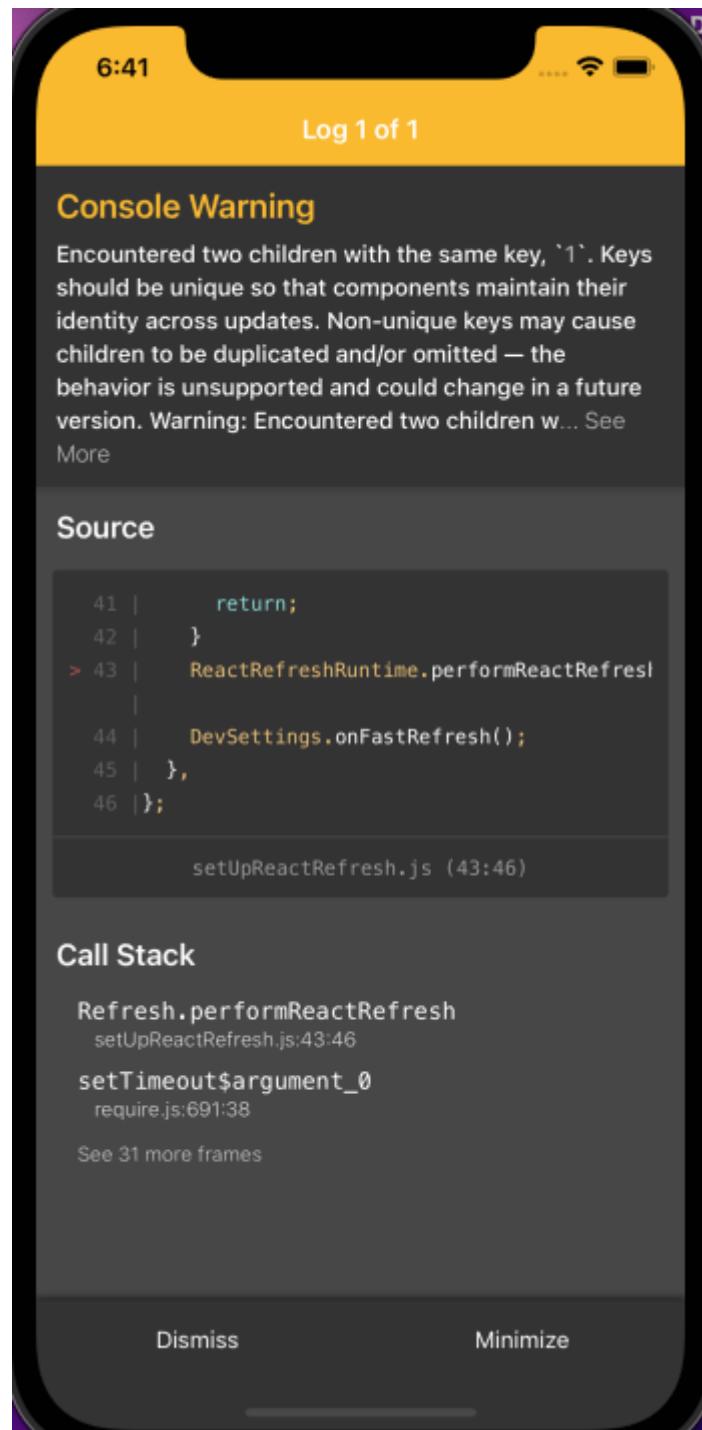


```
App.tsx myapp5p M X
myapp5p > App.tsx > ...
1 import { Text, View } from "react-native";
2
3 export default function App() {
4     return (
5         <View>
6             <Text key='1'>Sistema de Informação</Text>
7             <Text key='1'>React Native</Text>
8         </View>
9     )
10 }
```

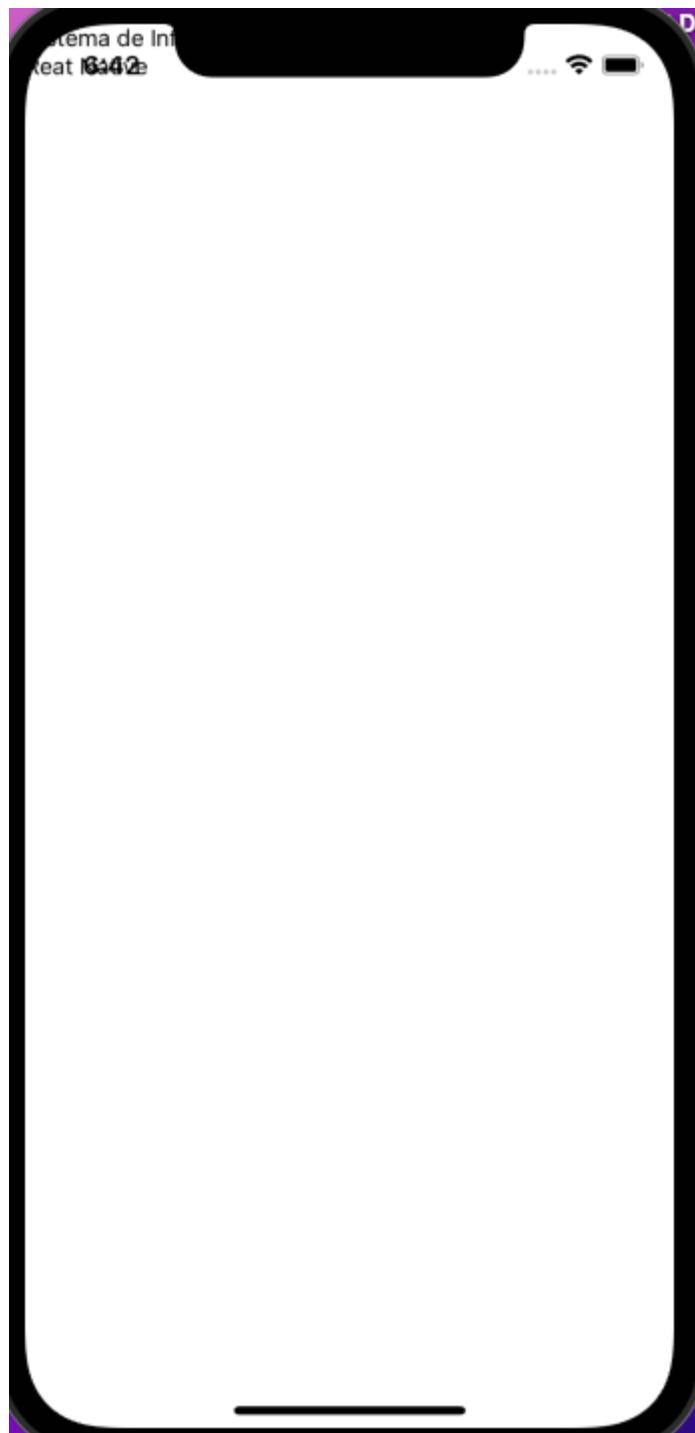
Salve as alterações e veja o que acontece.



Observe no final da tela é exibido um alerta de forma resumida. Foi encontrado dois filhos com a mesma chave. Se vc clicar poderá ver de uma forma detalhada.



Não é um erro que quebra a aplicação, porém é um alerta e vc tem que cuidar. Você pode ignorar clicando em Dismiss. Veja abaixo.

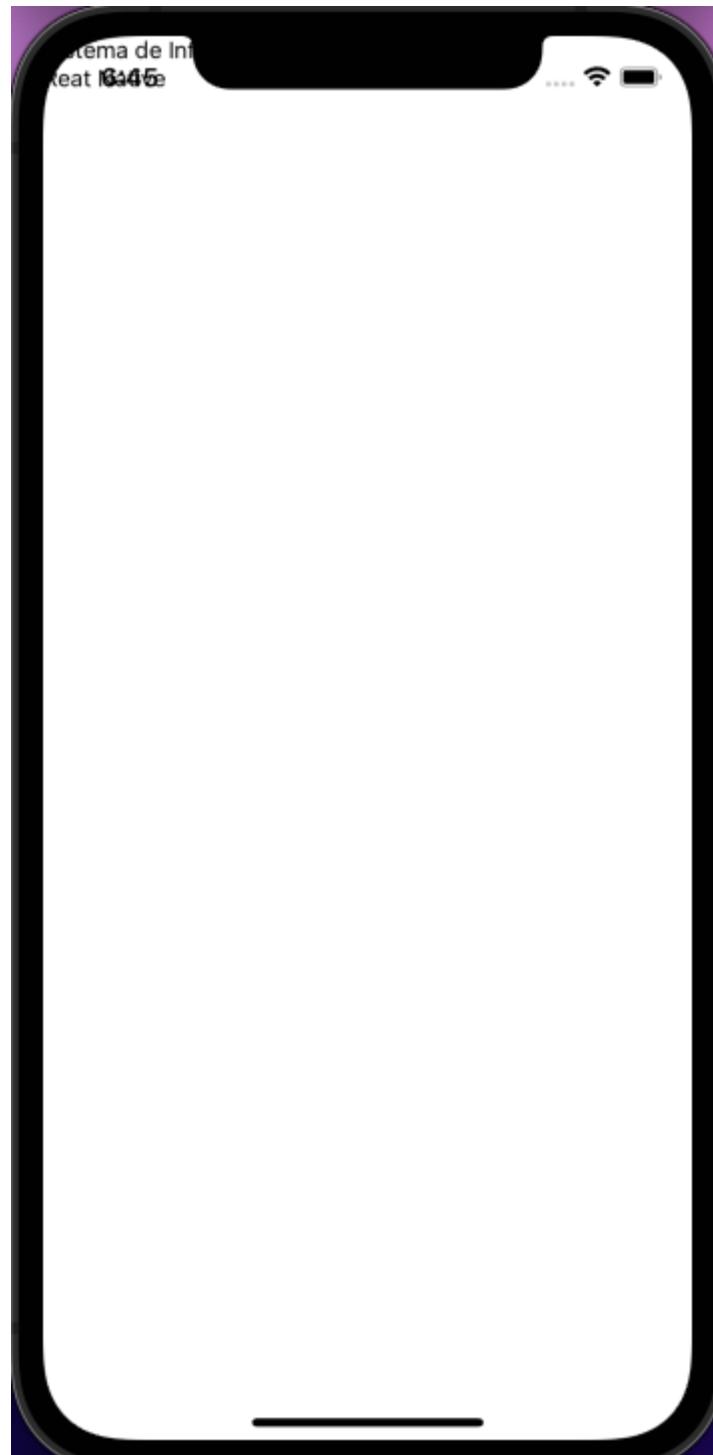


Porém se vc salvar o código será executado o fast refresh e volta a mensagem de erro alerta novamente.

Para corrigir altere a key do texto React Native na linha 7 conforme abaixo.

```
⚙️ App.tsx myapp5p M X
● myapp5p > ⚙️ App.tsx > ⚑ App
  1 import { Text, View } from "react-native";
  2
  3 export default function App() {
  4   return (
  5     <View>
  6       <Text key='1'>Sistema de Informação</Text>
  7       <Text key='2'>React Native</Text>
  8     </View>
  9   )
 10 }
```

Salve as alterações e veja o resultado.



Simulando o LogBox - cor vermelha - erro.

Ela acontece quando ocorre um erro que quebra a aplicação.

Altere a linha 6 retirando a letra t do componente Text conforme abaixo.

```
⚙ App.tsx myapp5p 2, M ●
● myapp5p > ⚙ App.tsx > ...
1 import { Text, View } from "react-native";
2
3 export default function App() {
4     return (
5         <View>
6             <Text key='1'>Sistema de Informação</Text>
7             <Text key='2'>React Native</Text>
8         </View>
9     )
10 }
```

Salve as alterações e veja o resultado.

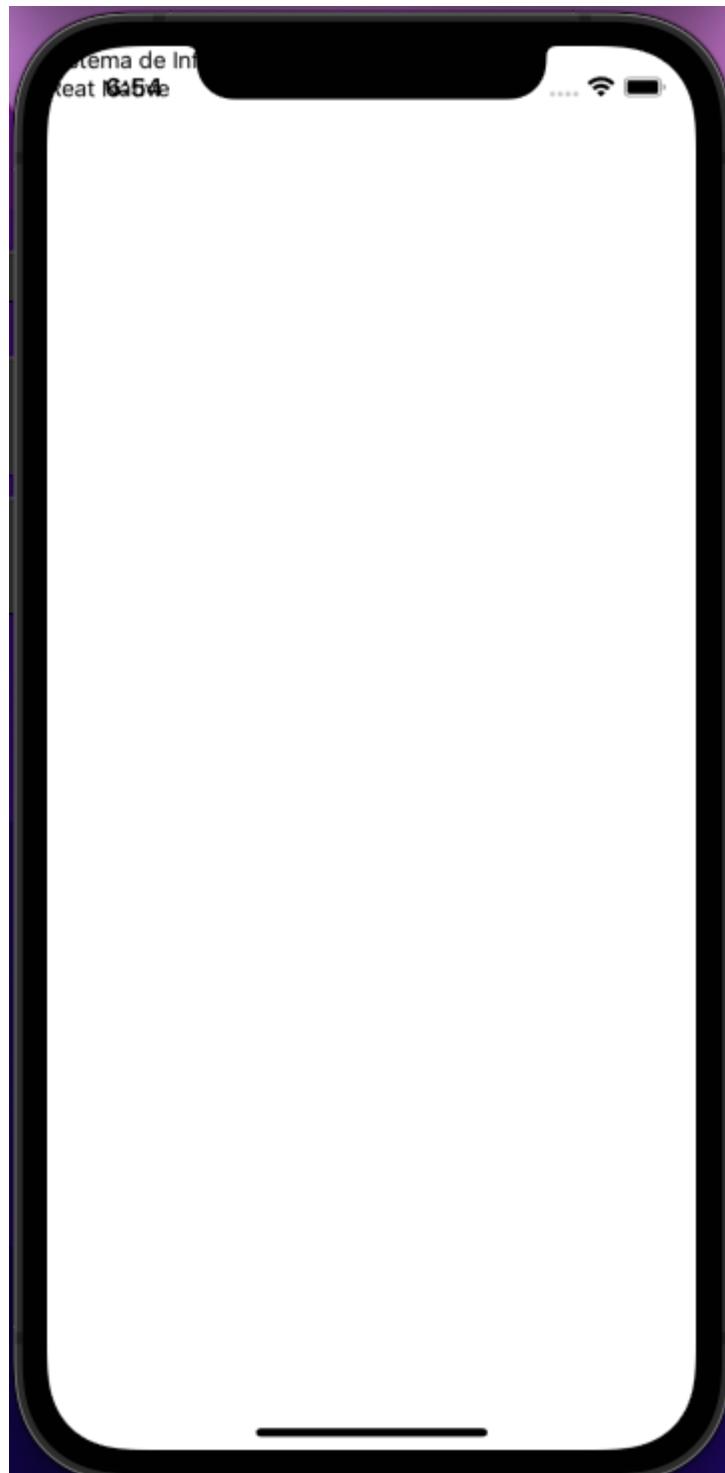


Ele mostra a mensagem de erro e sinaliza a linha 6 com o problema. Observe que não tem a opção para ignorar o erro, neste caso será necessário corrigir para o seu app voltar a funcionar.

Altere a linha 6 conforme abaixo.

```
⚙ App.tsx myapp5p M ●
● myapp5p > ⚙ App.tsx > ...
1 import { Text, View } from "react-native";
2
3 export default function App() {
4     return (
5         <View>
6             <Text key='1'>Sistema de Informação</Text>
7             <Text key='2'>React Native</Text>
8         </View>
9     )
10 }
```

Salve as alterações e veja o resultado.



Estas são as duas possibilidades que vc tem e vai ser interessante, pois irá te auxiliar no desenvolvimento do app.

Estilizando Elementos

A forma para estilizar é muito parecido com o css da web. A primeira forma é através da propriedade style dentro do próprio elemento. No React Native ele já vem com o flex box ativado por padrão.

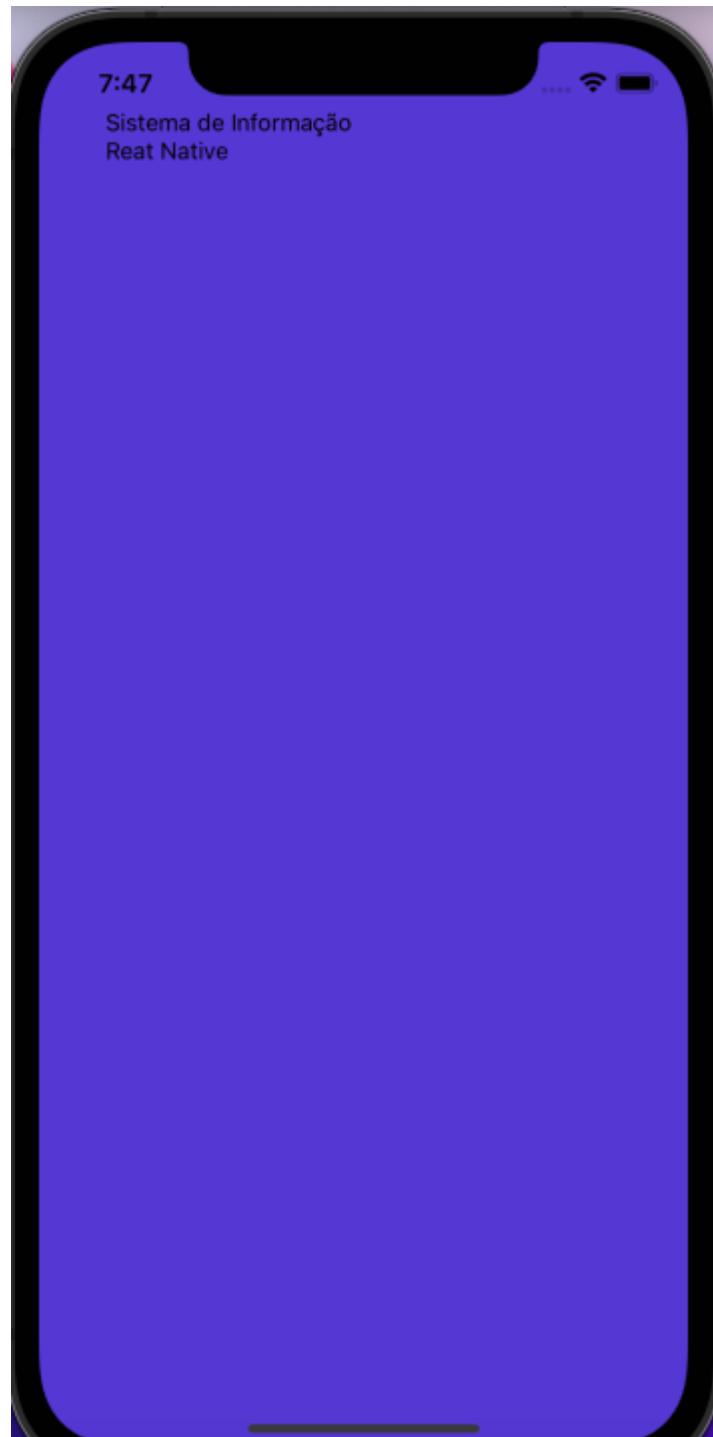
Insira a linha 6, o flex: 1 permite definir a estilização ocupar a tela inteira

Insira a linha 7 para definir a cor de fundo.

Insira a linha 8 para definir o espaçamento interno de 40

```
⚙️ App.tsx myapp5p M X
● myapp5p > ⚙️ App.tsx > ...
1 import { Text, View } from "react-native";
2
3 export default function App() {
4   return (
5     <View style={{{
6       flex: 1,
7       backgroundColor: '#5636D3',
8       padding: 40
9     }}>
10    <Text key='1'>Sistema de Informação</Text>
11    <Text key='2'>React Native</Text>
12  </View>
13)
14}
```

Salve e veja o resultado



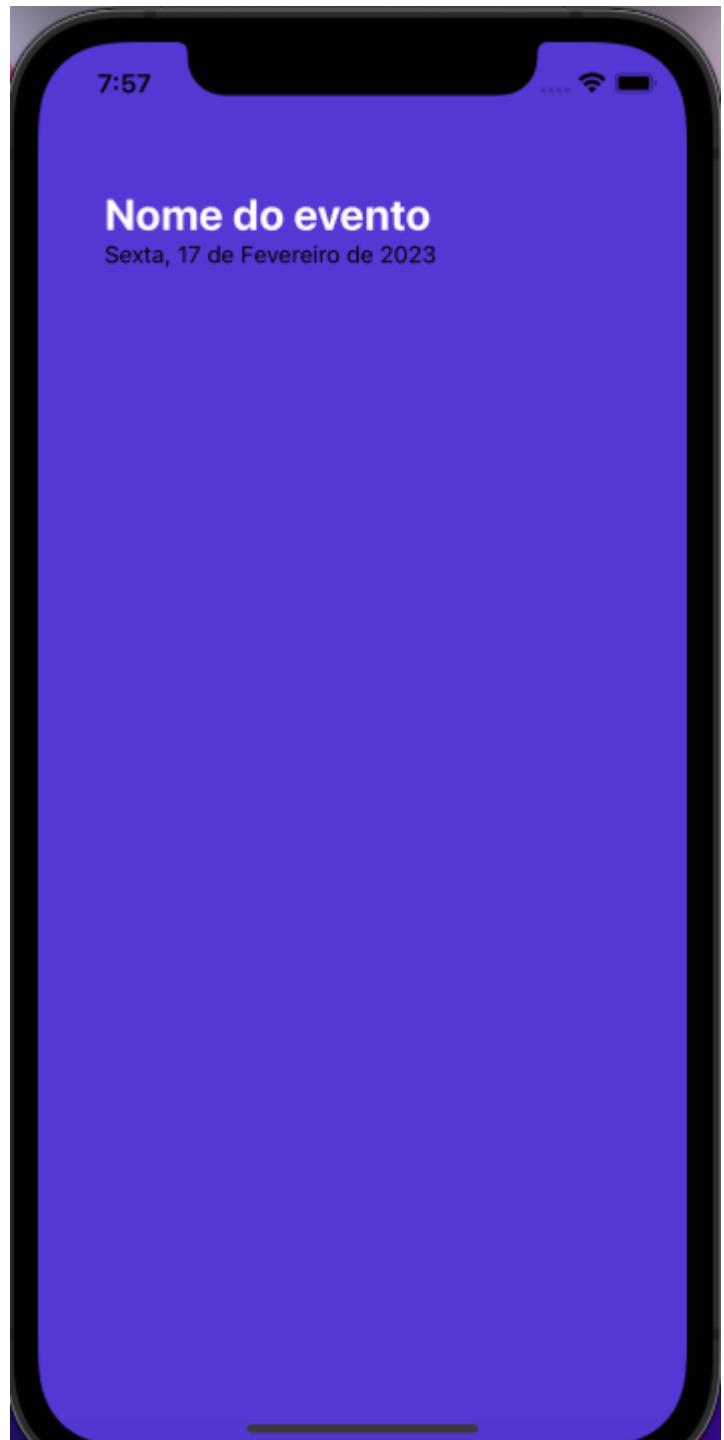
Agora defina uma estilização para o texto.

Altere a linha 10

Insira da linha 11 até 14 e altere as linha 15 e 16

```
⚙️ App.tsx myapp5p M X
    ⚙️ myapp5p > ⚙️ App.tsx > ⚡ App
3   export default function App() {
4     return (
5       <View style={{
6         flex: 1,
7         backgroundColor: '#5636D3',
8         padding: 40
9       }}>
10      <Text style={{
11        color: '#FDFCFE',
12        fontSize: 26,
13        fontWeight: 'bold',
14        marginTop: 48
15      }}>
16        Nome do evento
17      </Text>
18      <Text>
19        Sexta, 17 de Fevereiro de 2023
20      </Text>
21      </View>
22    )
23  }
```

Salve e veja o resultado

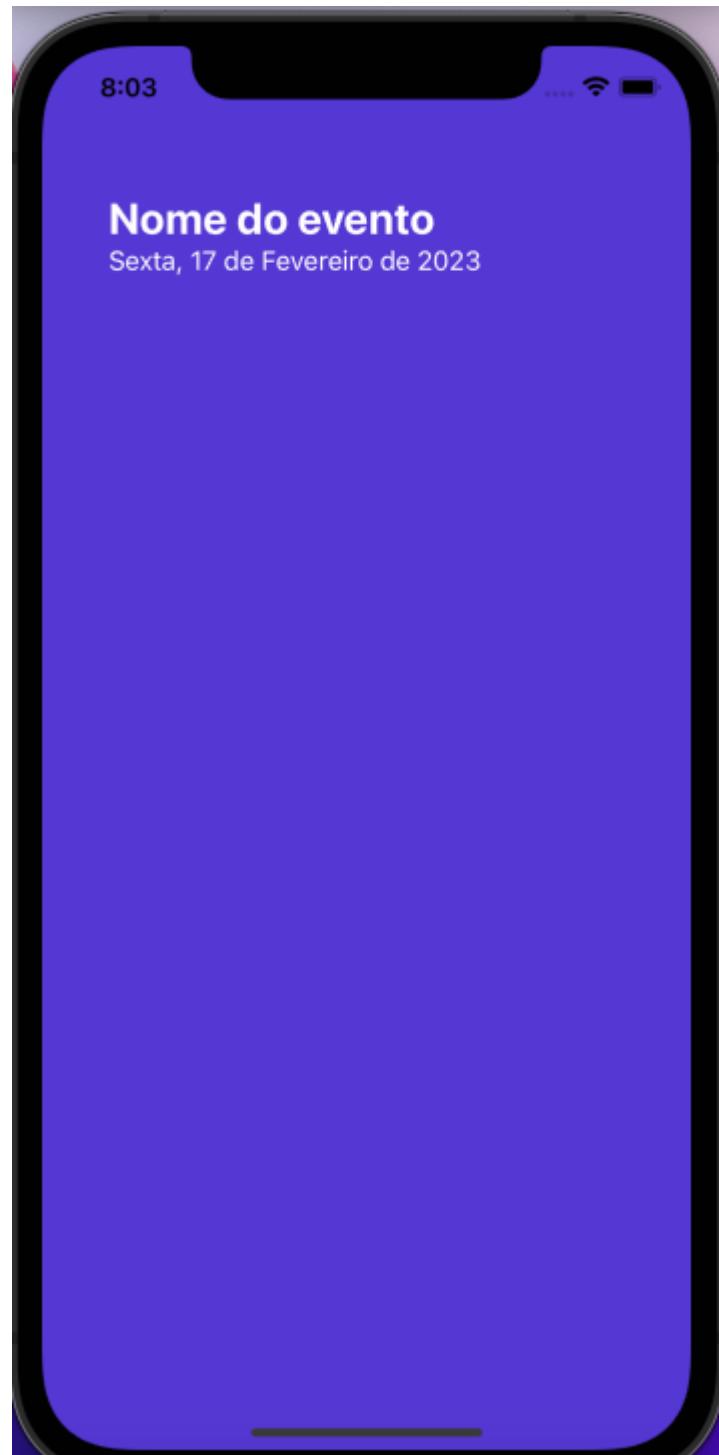


E estilize o próximo texto

Insira da linha 19 até 22

```
App.tsx myapp5p M X
myapp5p > App.tsx > ...
10   <Text style={{ ...
11     color: '#FDCCFE',
12     fontSize: 26,
13     fontWeight: 'bold',
14     marginTop: 48
15   }>
16   Nome do evento
17 </Text>
18 <Text
19   style={{ ...
20     color: '#FFF',
21     fontSize: 16
22   }>
23   Sexta, 17 de Fevereiro de 2023
24 </Text>
25 </View>
26 )
27 }
```

Salve e veja o resultado



Entendendo densidade de pixel

No RN vc não coloca uma unidade de media para definir o tamanho, por exemplo, `fontSize: 18`, vc não colocar `fontSize: 18px`. Por padrão a gente não coloca.

Por que a gente não coloca a unidade de medida?

Existem dois tipos de pixels

O Pixel de hardware é um ponto de luz na tela. Já o pixel de software é dinâmico de acordo com a densidade de pixels da tela.

O que é Densidade de Pixel?

É o número total de pixels que existe dentro de uma área física da tela (PPI - pixels per inch, ou pixel por polegada).

Densidade de Pixel

Inserir a imagem mostrando um botão, como se tivesse dado uma lupa nele.

maior densidade, existe muito mais pixels naquela tela.

menor densidade, existe muito menos pixels naquela tela.

Maior a densidade de pixels, melhor é a definição.

Densidade de Pixel

1 PPI

2 PPI

4 PPI

16 PPI

inserir uma imagem

Quanto mais perto do dispositivo mais detalhes de visualização será requerido, ou seja, será necessário uma maior densidade de pixels (smartphone e tablet). Quanto mais distante, menor poderá ser a densidade de pixels (TV, Projetores).

Quando mais perto do dispositivo, um smartphone, ele vai ver numa distância muito próxima, mais detalhe de visualização vai ser requerido, será necessário uma maior densidade de pixel.

Quanto mais distante vc pode ter uma menor quantidade de pixel.

Existem uma variedade muito grande de dispositivos e densidade de tela. Para resolver este problema foi criado uma medida independente da quantidade de pixels.

Independência de Densidade

inserir uma imagem

Renderiza os elementos de uma forma independente das características e densidades de cada dispositivo. Android - DPI (dots per inch - pontos por polegada) e IOS (points)

Ela renderiza os elementos de uma forma independente da característica de densidade de pixel do dispositivo da tela, ele vai renderizar os elementos de forma proporcional. Ela vai tentar deixar mais equivalente o tamanho dos elementos em tela em diferentes dispositivos.

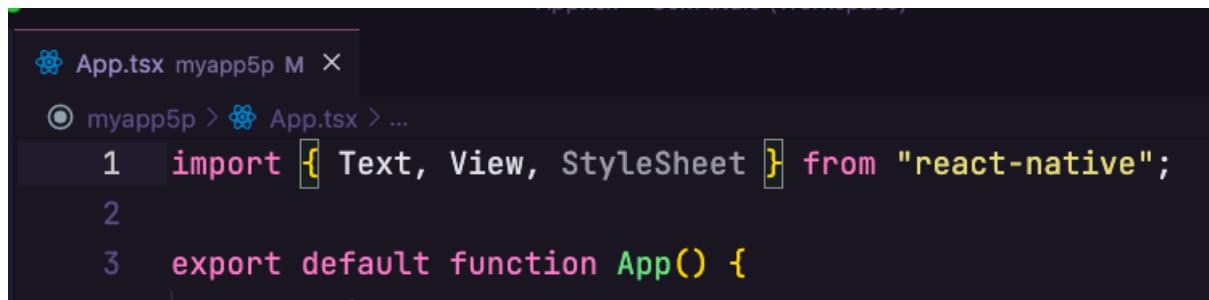
É por isso que a gente não coloca uma unidade de medida, por exemplo, px. Pq cada ambiente levando em considerando a unidade de medida que utiliza, ele vai renderizar dentro do seu ambiente de forma automática.

Utilizando StyleSheet

Ele permite separar a lógica de estilização dentro do RN.

Edite o arquivo App.tsx

Altere a linha 1 para importar o StyleSheet



```
App.tsx myapp5p M X
myapp5p > App.tsx > ...
1 import { Text, View, StyleSheet } from "react-native";
2
3 export default function App() {
```

A screenshot of a code editor showing the beginning of an App.tsx file. The file starts with an import statement for 'Text', 'View', and 'StyleSheet' from 'react-native'. The 'StyleSheet' part is highlighted in yellow. Below the imports, there is a blank line and then the start of the 'App()' function definition.

Insira no final do arquivo da linha 27 até 33. Ele a cria uma constante chamada styles que terá um objeto chamado container com as estilizações.

```
⚙️ App.tsx myapp5p M X
● myapp5p > ⚙️ App.tsx > ...
21           Sexta, 17 de Fevereiro de 2023
22           </Text>
23           </View>
24       )
25   }
26
27   const styles = StyleSheet.create({
28     container: {
29       flex: 1,
30       backgroundColor: '#5636D3',
31       padding: 40
32     }
33   })

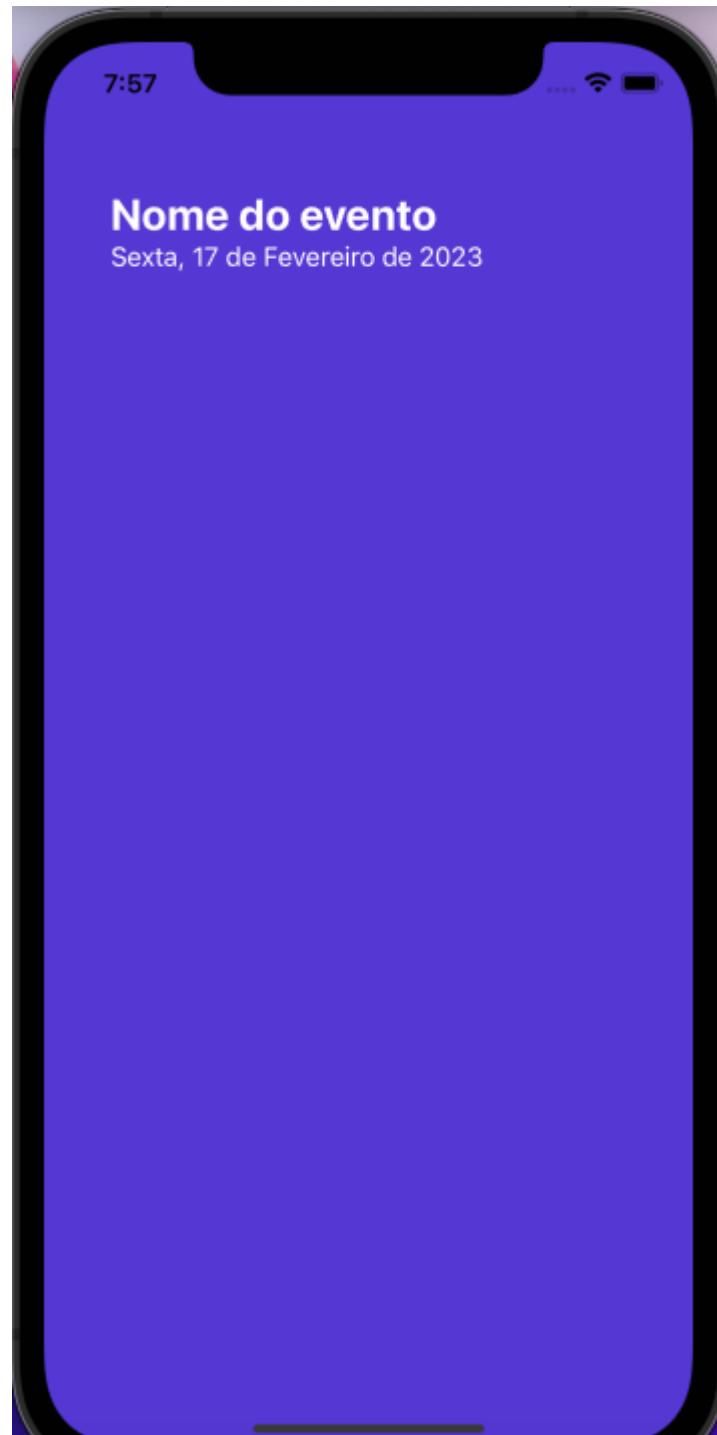
```

Para utilizar o objeto de estilização container, altere a linha 5 conforme abaixo. Observe que foi utilizado a constante styles e o objeto de estilização container.

```
⚙️ App.tsx myapp5p M X
● myapp5p > ⚙️ App.tsx > ↗ App
1 import { Text, View, StyleSheet } from "react-native";
2
3 export default function App() {
4   return [
5     <View style={styles.container}>
6       <Text style={{
7         color: '#FDFCFE',

```

Salve as alterações e veja que a aplicação da estilização continua da mesma forma.



A vantagem de utilizar o StyleSheet é deixar separada a estilização
Insira da linha 26 até 31 para criar o objeto de estilização eventName

```
⚙️ App.tsx myapp5p M X
● myapp5p > ⚙️ App.tsx > ...
18   }
19
20   const styles = StyleSheet.create({
21     container: {
22       flex: 1,
23       backgroundColor: '#5636D3',
24       padding: 40
25     },
26     eventName: {
27       color: '#FDFCCE',
28       fontSize: 26,
29       fontWeight: 'bold',
30       marginTop: 48
31     }
32   })

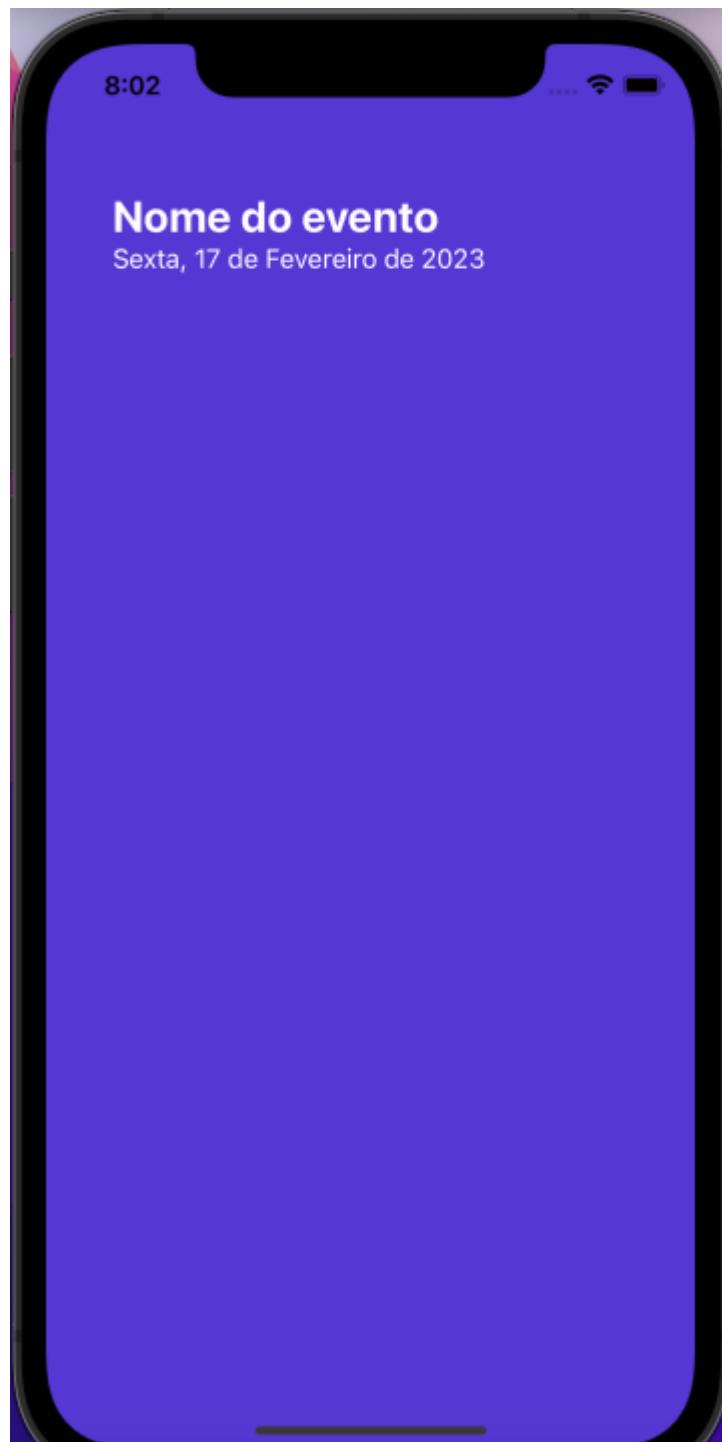
```

Altere a linha 6 para utilizar o objeto eventName

```
⚙️ App.tsx myapp5p M X
● myapp5p > ⚙️ App.tsx > ⌂ App
1 import { Text, View, StyleSheet } from "react-native";
2
3 export default function App() {
4   return (
5     <View style={styles.container}>
6       <Text style={styles.eventName}>
7         Nome do evento
8       </Text>

```

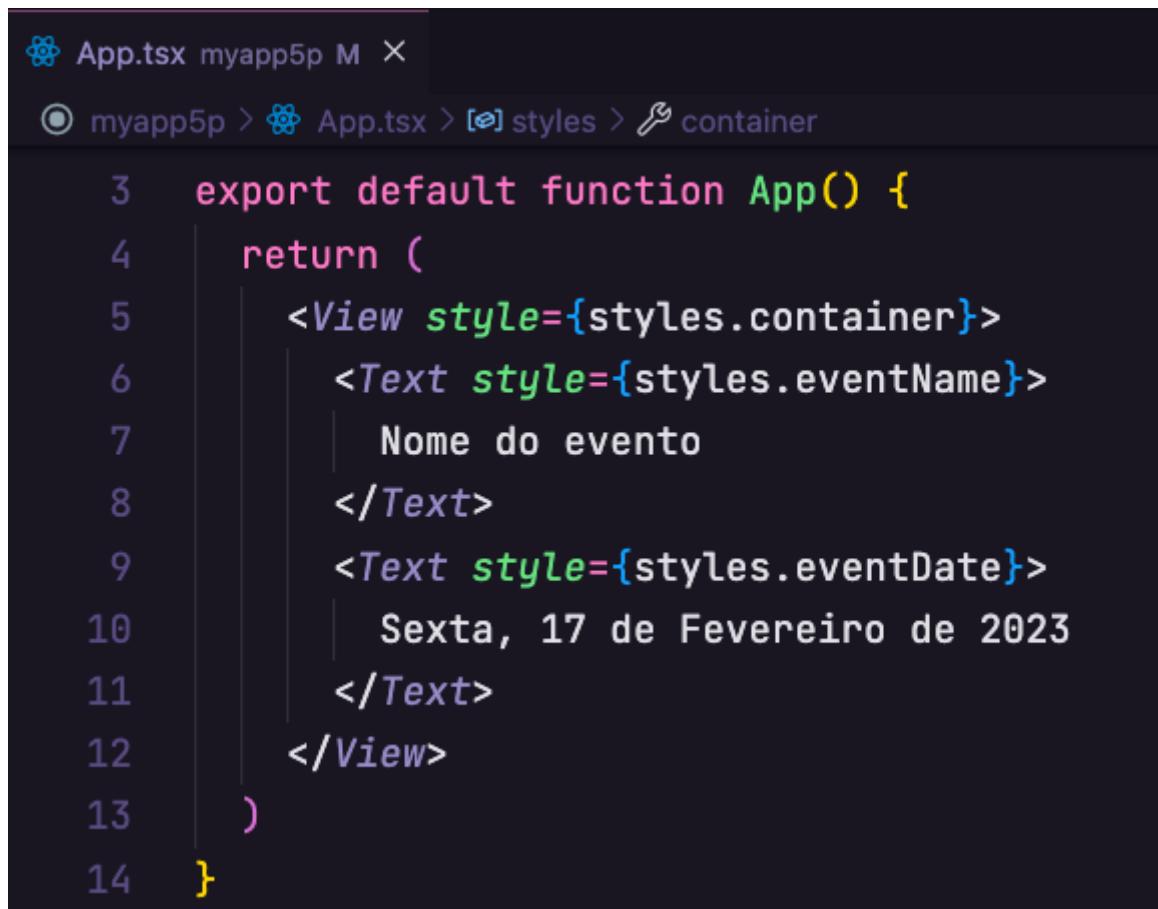
Salve as alterações e veja o resultado do seu app.



Insira da linha 29 até 32 para criar o objeto eventDate

```
⚙️ App.tsx myapp5p M X
⌚ myapp5p > ⚙️ App.tsx > ...
20   backgroundColor: '#5636D3',
21   padding: 40
22 },
23   eventName: {
24     color: '#FDFCCE',
25     fontSize: 26,
26     fontWeight: 'bold',
27     marginTop: 48
28 },
29   eventDate: {
30     color: '#FFF',
31     fontSize: 16
32 }
33 })
```

Altere a linha 9 para utilizar o objeto de estilização eventDate

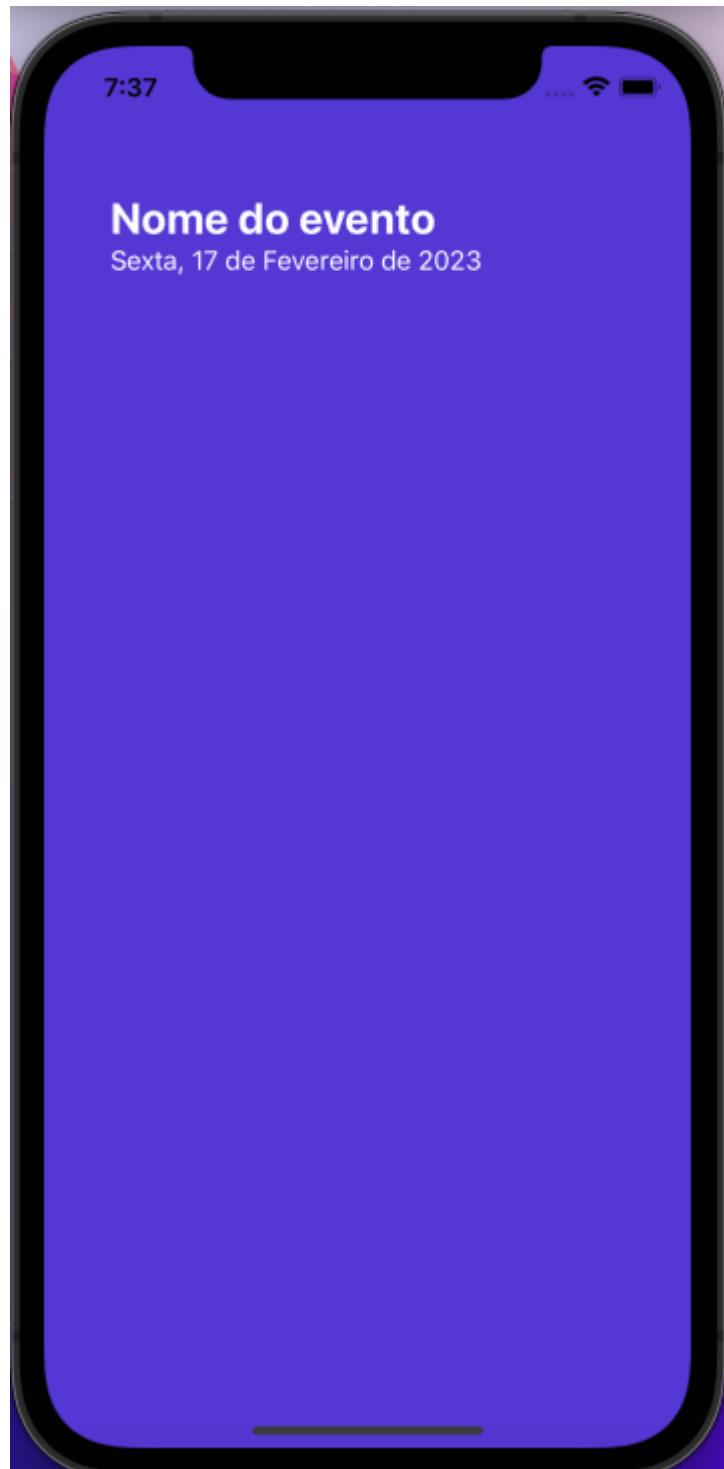


The screenshot shows a code editor window with the following details:

- File: App.tsx
- Project: myapp5p
- Editor tab: styles
- Code content:

```
3  export default function App() {
4    return (
5      <View style={styles.container}>
6        <Text style={styles.eventName}>
7          Nome do evento
8        </Text>
9        <Text style={styles.eventDate}>
10          Sexta, 17 de Fevereiro de 2023
11        </Text>
12      </View>
13    )
14 }
```

Salve as alterações e observe que o app continua funcionando da mesma forma.



Com o StyleSheet vc consegue separar as regras de estilizações deixando o seu código mais limpo.

Organizando o projeto

O próximo passo é separar o código em arquivos.

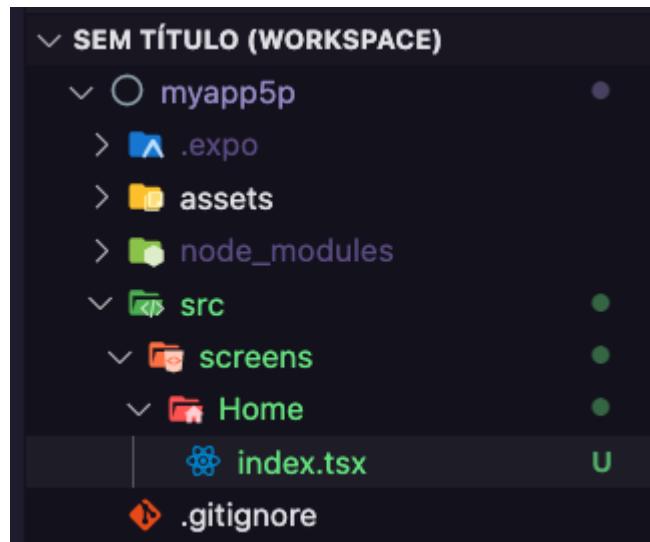
Crie a pasta src no mesmo nível de estrutura da pasta assets

Crie a pasta src/screens

Crie a pasta src/screens/Home

Crie o arquivo src/screens/Home/index.tsx

Veja abaixo o print



O arquivo .tsx vc irá usar a sintaxe e JSX, ele será composto pelo código TypeScript e o return para retornar uma interface declarativa.

Deixe o seu arquivo src/screens/Home/index.tsx conforme abaixo

```
index.tsx Home 3, U X
myapp5p > src > screens > Home > index.tsx > ...
1 import { Text, View, StyleSheet } from "react-native";
2
3 export function Home() {
4   return (
5     <View style={styles.container}>
6       <Text style={styles.eventName}>
7         Nome do evento
8       </Text>
9       <Text style={styles.eventDate}>
10        Sexta, 17 de Fevereiro de 2023
11      </Text>
12    </View>
13  )
14}
```

Crie o arquivo src/screens/Home/styles.ts com o conteúdo abaixo.



```
ts styles.ts Home 1, U ×
myapp5p > src > screens > Home > ts styles.ts > ...
1 const styles = StyleSheet.create({
2   container: {
3     flex: 1,
4     backgroundColor: '#5636D3',
5     padding: 40
6   },
7   eventName: {
8     color: '#FDFCFE',
9     fontSize: 26,
10    fontWeight: 'bold',
11    marginTop: 48
12  },
13  eventDate: {
14    color: '#FFF',
15    fontSize: 16
16  }
17})
```

Observe que gerou o erro na linha 1 com o create pois o StyleSheet não foi importado. Insira a linha 1 conforme print abaixo.



```
styles.ts Home U ×
myapp5p > src > screens > Home > styles.ts > ...
1 import { StyleSheet } from "react-native"
2
3 const styles = StyleSheet.create({
4   container: {
5     flex: 1,
6     backgroundColor: '#5636D3',
7     padding: 40
8   },
9   eventName: {
10    color: '#FDFFC6',
11    fontSize: 26,
12    fontWeight: 'bold',
13    marginTop: 48
14  },
15   eventDate: {
16    color: '#FFF',
17    fontSize: 16
18  }
19 })
```

O próximo passo será exportar a constante styles que está no arquivo styles.ts para que ela seja importada no arquivo index.tsx.

Altere a linha 3 conforme print abaixo

```
ts styles.ts Home U X
● myapp5p > src > screens > Home > ts styles.ts > ...
1 import { StyleSheet } from "react-native"
2
3 export const styles = StyleSheet.create({
4   container: {
5     flex: 1,
6     backgroundColor: '#5636D3',
7     padding: 40
8   },
9   eventName: {
10    color: '#FDFFC1',
11    fontSize: 26,
12    fontWeight: 'bold',
13    marginTop: 48
14  },
15   eventDate: {
16    color: '#FFF',
17    fontSize: 16
18  }
19 })
```

O próximo passo é importar a constante styles dentro do arquivo index.tsx

Edite o arquivo src/screens/Home/index.tsx

Altere a linha 1 para apagar a importação do StyleSheet

Insira a linha 3 conforme print abaixo.

```
index.tsx Home U X
myapp5p > src > screens > Home > index.tsx > ...
1 import { Text, View } from "react-native";
2
3 import { styles } from './styles'
4
5 export function Home() {
6     return (
7         <View style={styles.container}>
8             <Text style={styles.eventName}>
9                 Nome do evento
10            </Text>
11            <Text style={styles.eventDate}>
12                Sexta, 17 de Fevereiro de 2023
13            </Text>
14        </View>
15    )
16 }
```

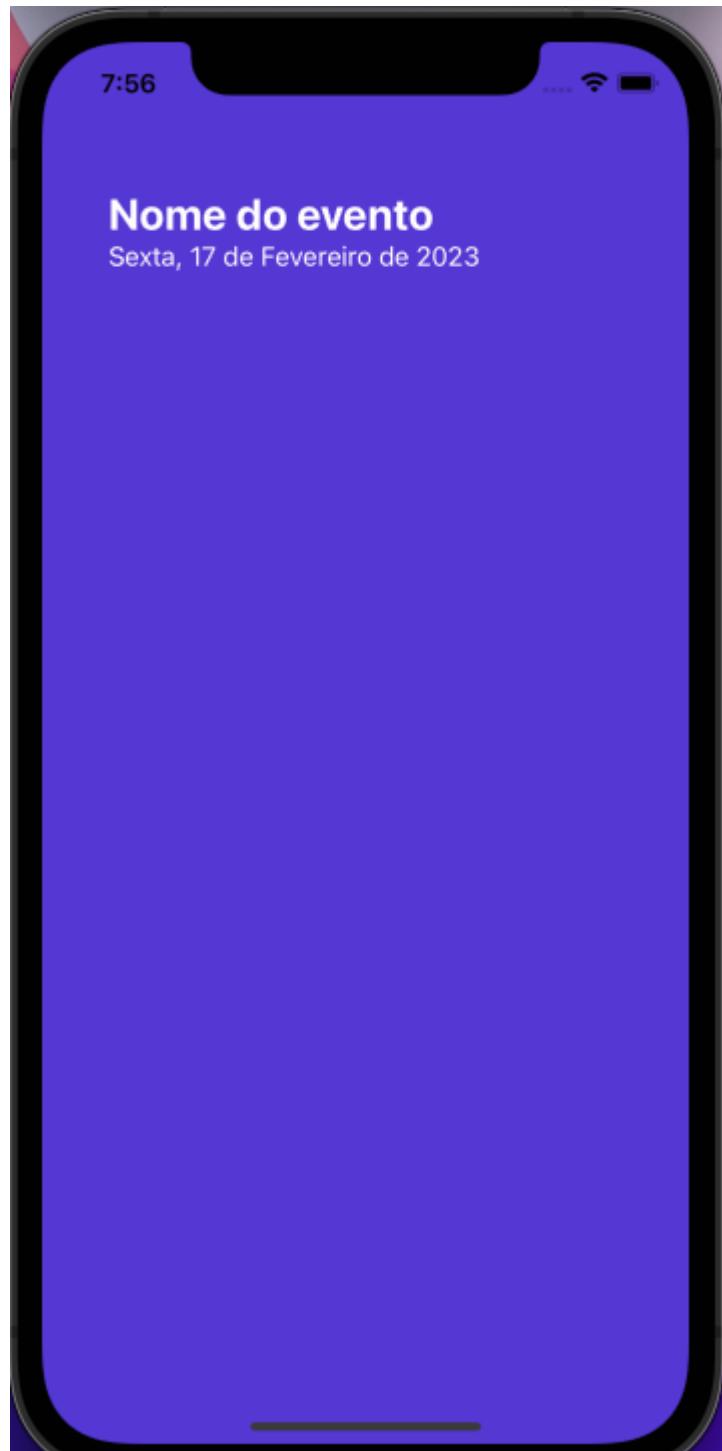
e o próximo passo é importar a screen Home no arquivo App.tsx

Insira a linha 1 para importar o Home

Altere a linha 5 para exibir o componente Home

```
⚙️ App.tsx myapp5p M X
● myapp5p > ⚙️ App.tsx > ...
1 import { Home } from "./src/screens/Home";
2
3 export default function App() {
4     return (
5         <Home />
6     )
7 }
```

Salve as alterações e veja que o App continua funcionando da mesma forma



Export e Export Default

Export

Quando vc exporta com o export function Home(), vc deve importá-lo como
import { Home } from './src/screens/Home'. Veja o abaixo o exemplo

Edite o arquivo src/screens/Home/index.tsx

Na linha 5 está exportando com o export function.

```
index.tsx Home U X
myapp5p > src > screens > Home > index.tsx > Home
1 import { Text, View } from "react-native";
2
3 import { styles } from './styles'
4
5 export function Home() {
6   return [
7     <View style={styles.container}>
8       <Text>Home</Text>
9     </View>
10    <Text>Home</Text>
11  ]
12}
```

Edite o arquivo App.tsx

Observe que na linha 1 o arquivo Home está sendo importado entre chaves

```
App.tsx myapp5p M X
myapp5p > App.tsx > ...
1 import { Home } from "./src/screens/Home";
2
3 export default function App() {
4   return (
5     <Home />
6   )
7 }
```

a vantagem do `export function` é que vc pode exportar uma constante ou outra função por exemplo, `export const CONFIG = ''` e para importar vc utilizar

```
import { Home, CONFIG } from './src/screens/Home'
```

Edite o arquivo src/screens/Home/index.tsx

Insira a linha 5, observe que neste arquivo tem duas exportações

```
⚙️ index.tsx Home U X
● myapp5p > src > screens > Home > ⚙️ index.tsx > ...
1 import { Text, View } from "react-native";
2
3 import { styles } from './styles'
4
5 export const CONFIG = '';
6
7 export function Home() {
8   return (
9     <View style={styles.container}>
```

Edite o arquivo App.tsx

Altere a linha 1 para importar a constante CONFIG

```
⚙️ App.tsx myapp5p M X
● myapp5p > ⚙️ App.tsx > ...
1 import { Home, CONFIG } from "./src/screens/Home";
2
3 export default function App() {
4   return (
5     <Home />
6   )
7 }
```

obs: O export é chamada de exportação nomeada

Export Default

Se vc exportar como **export default function Home()**, vc deve importá-lo como **import Home from './src/screens/Home'**

Edite o arquivo src/screens/Home/index.tsx

Altere a linha 5 conforme abaixo, observe que a exportação da constante CONFIG foi apagada.

```
index.tsx Home U X
myapp5p > src > screens > Home > index.tsx > Home
1 ~ import { Text, View } from "react-native";
2
3   import { styles } from './styles'
4
5 ~ export default function Home() {
6   ~ return (
7     ~   <View style={styles.container}>
8       ~   <Text style={styles.eventName}>
```

Edite o arquivo App.tsx

Observe que está gerando um erro na linha 1, no Home é porque ele foi exportado com o export default e para importá-lo, não pode usar as chaves. O erro do CONFIG é porque ele foi apagado do arquivo src/screens/Home/index.tsx

```
App.tsx myapp5p 2, M X
myapp5p > App.tsx > ...
1   import { Home, CONFIG } from "./src/screens/Home";
2
3   export default function App() {
4     return (
5       <Home />
6     )
7 }
```

Altere a linha 1 conforme abaixo

Toda exportação com export default, obrigatoriamente vc tem que importar sem as chaves

```
App.tsx myapp5p M X
myapp5p > App.tsx > ...
1 import Home from "./src/screens/Home";
2
3 export default function App() {
4     return (
5         <Home />
6     )
7 }
```

No export default vc pode mudar o nome ao importar, por exemplo.

export default function Home()

import Batata from './src/screens/Home'

Se vc usar o export ele vai te obrigar a usar o nome correto, por exemplo

export function Home()

import { Batata } from './src/screens/Home'

Neste exemplo ele vai gerar um erro.

Eu particularmente eu prefiro utilizar a exportação com o export ao invés de export default, exceto o arquivo App que obrigatoriamente tem que ser o export default
Você pode combinar as duas exportas.

export default function Home()

export const CONFIG = ""

import Home, { CONFIG } from './src/screens/Home';

Input de dados

Para capturar os dados digitados pelo usuário será utilizado o componente TextInput

Altere a linha 1 para inserir o TextInput

```
index.tsx Home 1, U ×  
myapp5p > src > screens > Home > index.tsx > Home  
1 import { Text, View, TextInput } from "react-native";  
2  
3 import { styles } from './styles'  
4
```

Insira a linha 15, observe que está gerando o erro pois não existe a estilização input

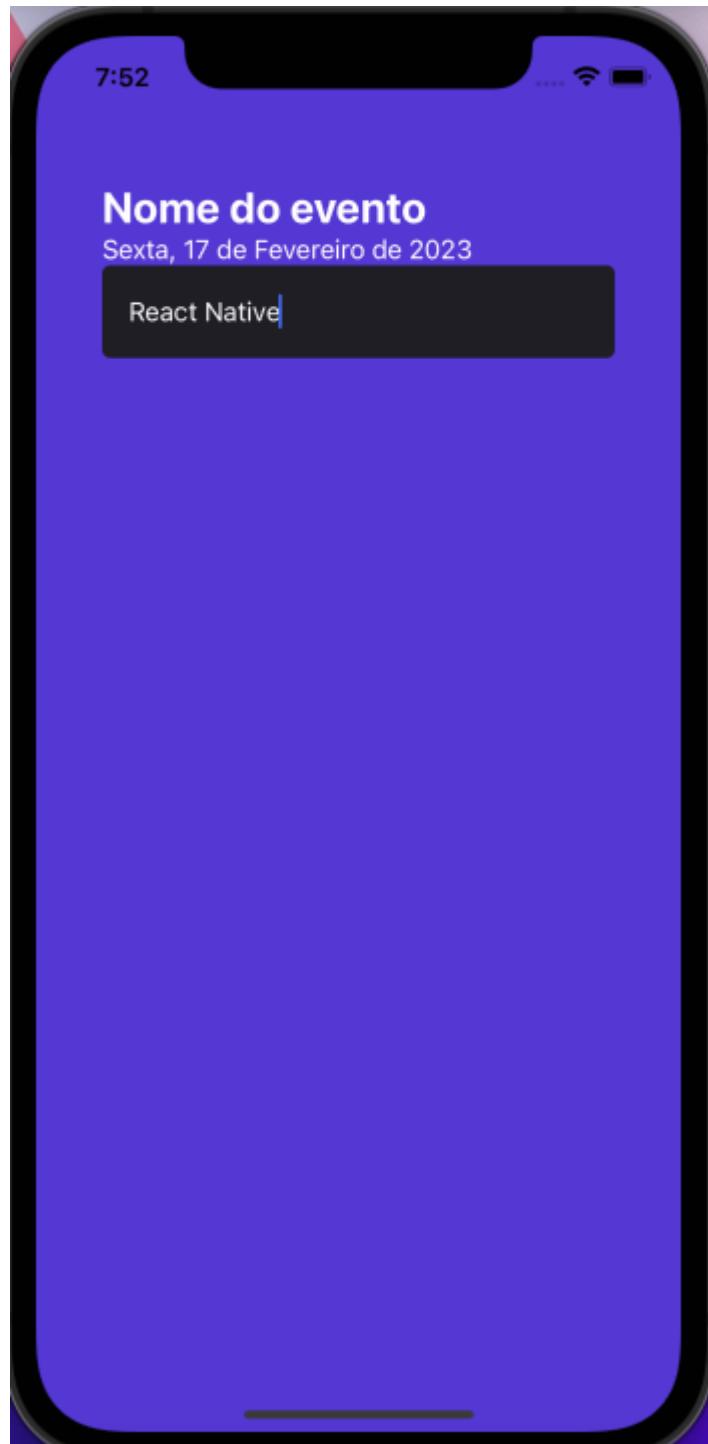
```
index.tsx Home 1, U ×  
myapp5p > src > screens > Home > index.tsx > ...  
5 export default function Home() {  
6   return (  
7     <View style={styles.container}>  
8       <Text style={styles.eventName}>  
9         Nome do evento  
10        </Text>  
11        <Text style={styles.eventDate}>  
12          Sexta, 17 de Fevereiro de 2023  
13        </Text>  
14  
15        <TextInput style={styles.input} />  
16      </View>  
17    )  
18  }
```

Edito o arquivo src/screens/Home/styles.ts

Insira da linha 19 até 24

```
ts styles.ts Home U X
● myapp5p > src > screens > Home > ts styles.ts > ...
15   eventDate: {
16     color: '#FFF',
17     fontSize: 16
18   },
19   input: {
20     height: 56,
21     backgroundColor: '#1F1E25',
22     borderRadius: 5,
23     color: '#FFF',
24     padding: 16,
25     fontSize: 16
26   }
27 })
```

Salve e veja o resultado. Veja abaixo um exemplo digitando o texto React Native



Definindo propriedades para o TextInput

Edite o arquivo src/screens/Home/index.tsx

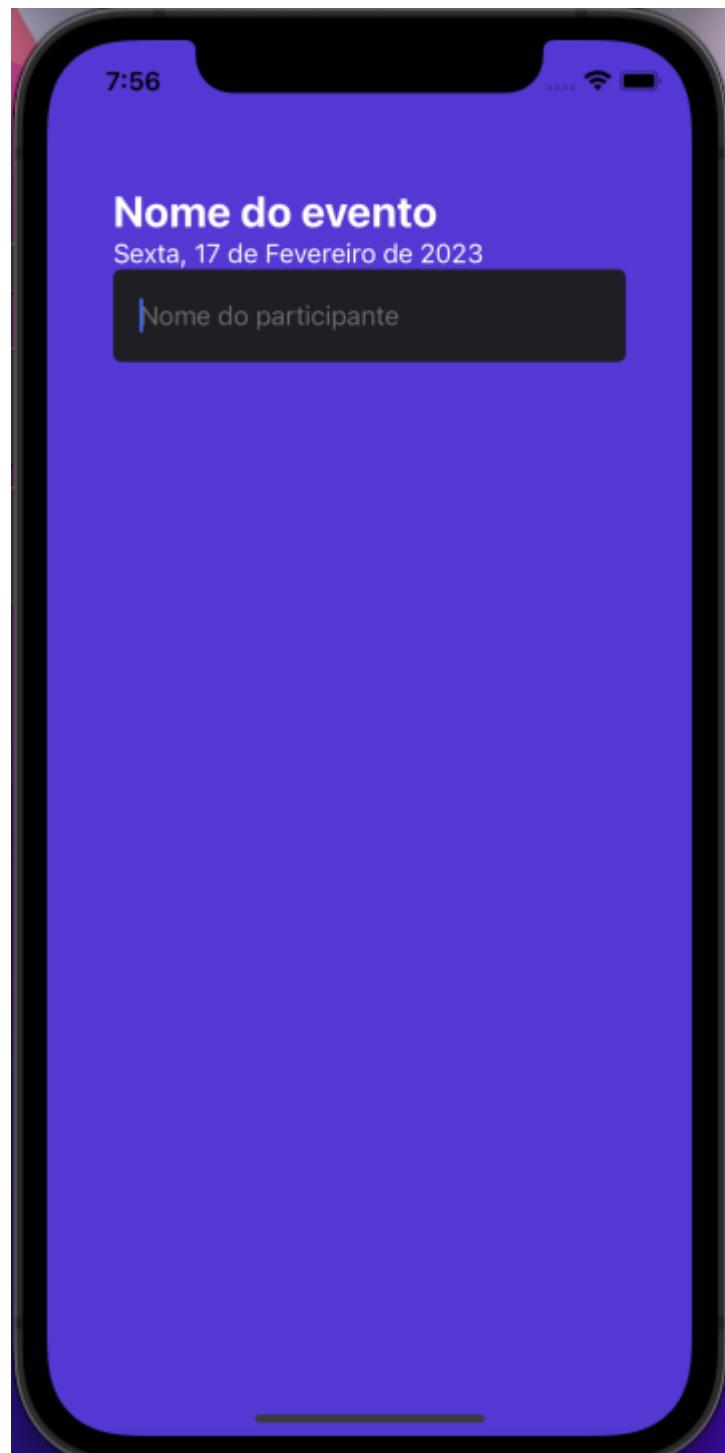
Insira da linha 15 até 19

A propriedade placeholder na linha 17 define o texto do input

A propriedade placeholderTextColor na linha 18 define a cor do texto do input

```
⚙️ index.tsx Home ✘ ×
● myapp5p > src > screens > Home > ⚙️ index.tsx > ...
11   <Text style={styles.eventDate}>
12     Sexta, 17 de Fevereiro de 2023
13   </Text>
14
15   <TextInput
16     style={styles.input}
17     placeholder='Nome do participante'
18     placeholderTextColor='□#6B6B6B'
19   />
20   </View>
21 )
22 }
```

Salve e veja o resultado



Outra propriedade que vc pode definir para definir o tipo do teclado.

Insira a linha 19 e dentro das aspas pressione ctrl + barra de espaço para ver as opções.

The screenshot shows a code editor with the file `index.tsx` open. The code is part of a React Native application. A dropdown menu is open at line 19, showing various keyboard types. The menu items are:

- ascii-capable
- decimal-pad
- default
- email-address
- name-phone-pad
- number-pad
- numbers-and-punctuation
- numeric
- phone-pad
- twitter
- url
- visible-password

Define a linha 19 conforme abaixo

```
index.tsx Home U X
myapp5p > src > screens > Home > index.tsx > ...
11      <Text style={styles.eventDate}>
12          Sexta, 17 de Fevereiro de 2023
13      </Text>
14
15      <TextInput
16          style={styles.input}
17          placeholder='Nome do participante'
18          placeholderTextColor='#6B6B6B'
19          keyboardType='numeric'
20      />
21      </View>
22  )
23 }
```

Salve e veja o resultado

Para definir que todas os caracteres digitados ficarão em letras maiúsculas

Insira a linha 20

```
index.tsx Home U X
myapp5p > src > screens > Home > index.tsx > ...
14
15     <TextInput
16         style={styles.input}
17         placeholder='Nome do participante'
18         placeholderTextColor='#6B6B6B'
19         // keyboardType='numeric'
20         autoCapitalize='characters'
21     />
22     </View>
23 )
24 }
```

Salve as alterações e faça um teste

Botão Adicionar

Edite o arquivo src/screens/Home/index.tsx

Para trabalhar com botões no React Native, utilize o TouchableOpacity. Altere a linha 1 para importá-lo

```
index.tsx Home U ●
myapp5p > src > screens > Home > index.tsx > Home
1 import { Text, View, TextInput, TouchableOpacity } from "react-native";
2
3 import { styles } from './styles'
4
5 export default function Home() {
```

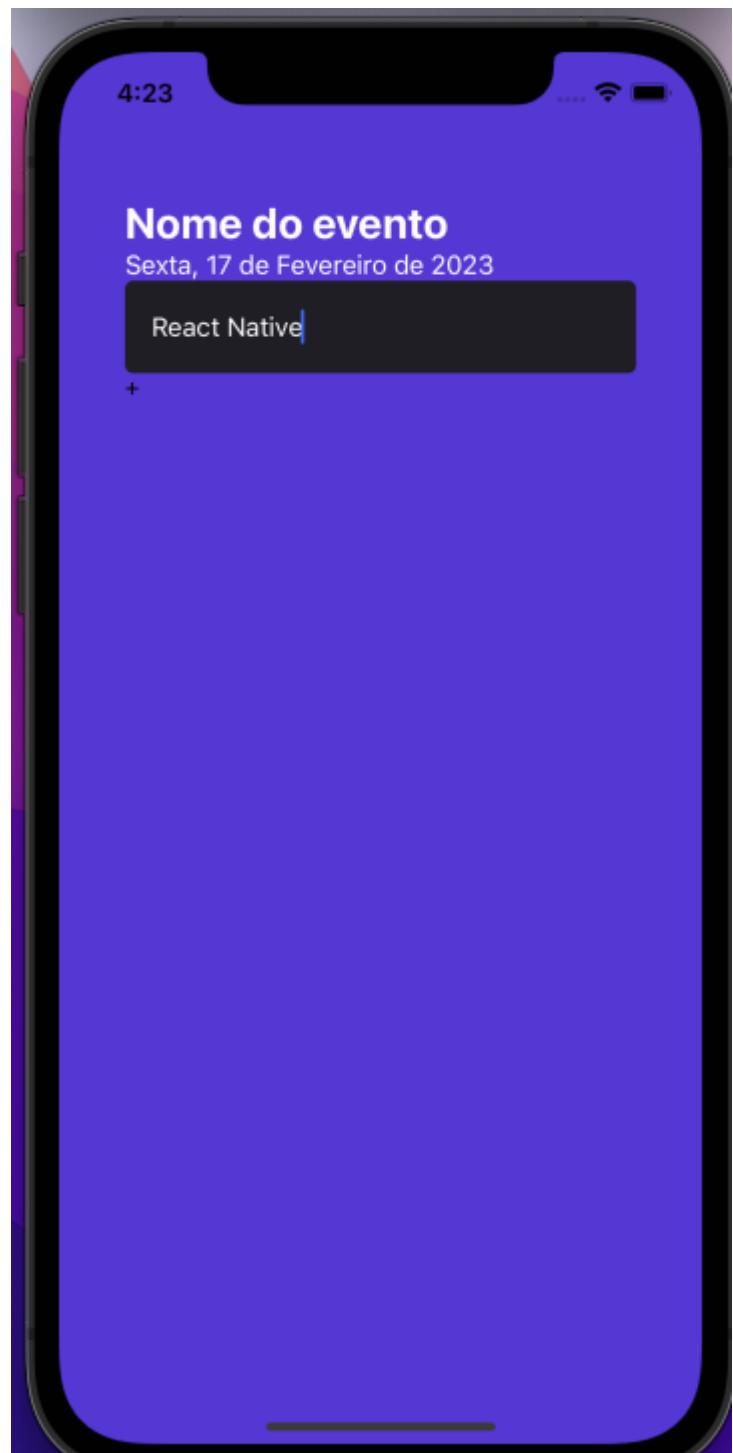
Insira da linha 21 até 25

The screenshot shows a code editor window with the following details:

- File: index.tsx
- Project: myapp5p
- Path: src > screens > Home > index.tsx

```
15     <TextInput
16         style={styles.input}
17         placeholder='Nome do participante'
18         placeholderTextColor='#6B6B6B'
19     />
20
21     <TouchableOpacity>
22         <Text>
23             +
24         </Text>
25     </TouchableOpacity>
26
27     </View>
28 )
29 }
```

Salve as alterações e veja o resultado. Observe que será exibido um sinal de mais.



Altere a linha 22 para definir o objeto de estilização buttonText

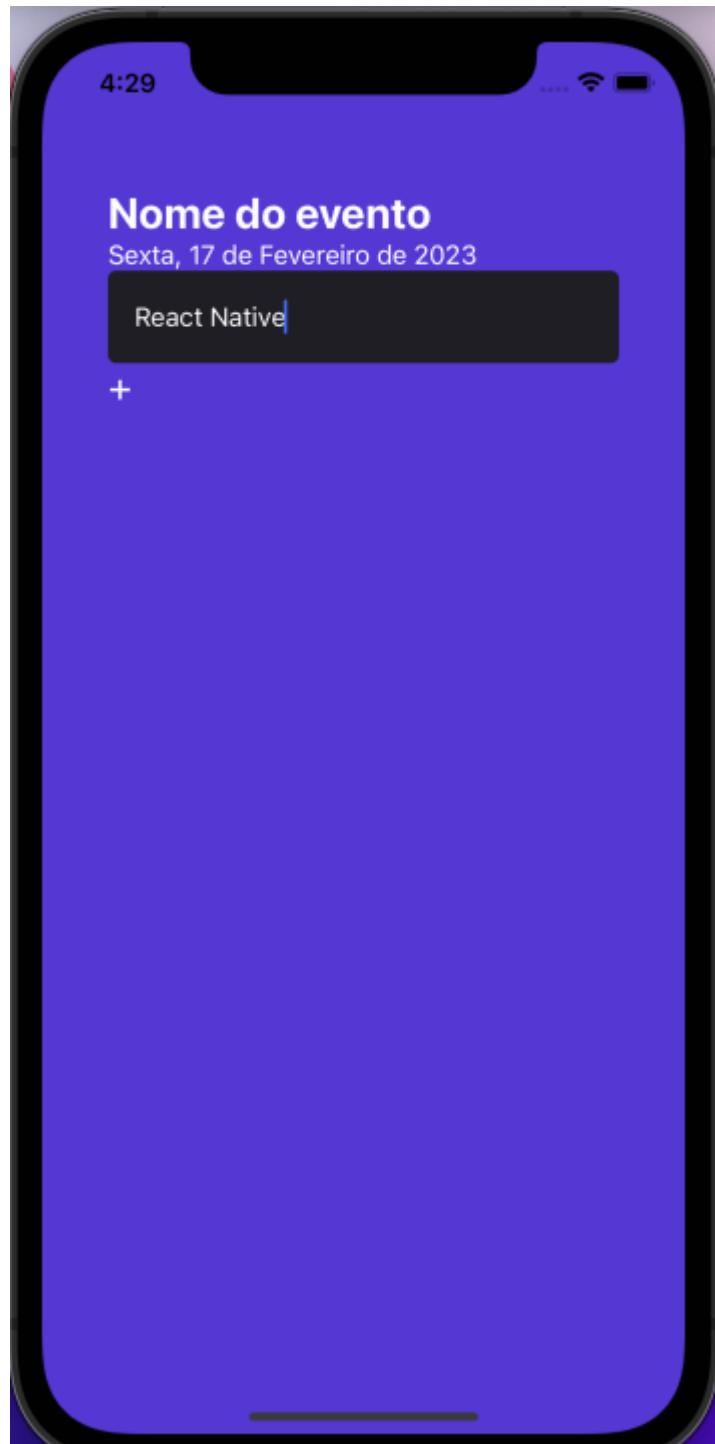
```
index.tsx Home 1, U ●
myapp5p > src > screens > Home > index.tsx > ...
15     <TextInput
16         style={styles.input}
17         placeholder='Nome do participante'
18         placeholderTextColor='#6B6B6B'
19     />
20
21     <TouchableOpacity>
22         <Text style={styles.buttonText}>
23             +
24         </Text>
25     </TouchableOpacity>
26
27     </View>
28 )
29 }
```

Edite o arquivo src/screens/Home/styles.ts

Insira da linha 27 até 30

```
ts styles.ts Home U X
myapp5p > src > screens > Home > ts styles.ts > ...
19   input: {
20     height: 56,
21     backgroundColor: '#1F1E25',
22     borderRadius: 5,
23     color: '#FFF',
24     padding: 16,
25     fontSize: 16
26   },
27   buttonText: {
28     color: '#FFF',
29     fontSize: 24,
30   }
31 })
```

Salve as alterações e veja o resultado. Clique no botão e veja o efeito de opacidade



Edite o arquivo src/screens/Home/index.tsx

Altere a linha 21 para definir um objeto de estilização para o button

```
⚙️ index.tsx Home 1, U •  
● myapp5p > src > screens > Home > ⚙️ index.tsx > ...  
15     <TextInput  
16         style={styles.input}  
17         placeholder='Nome do participante'  
18         placeholderTextColor='□#6B6B6B'  
19     />  
20  
21     <TouchableOpacity style={styles.button}>  
22         <Text style={styles.buttonText}>  
23             +  
24         </Text>  
25     </TouchableOpacity>  
26  
27     </View>  
28 )  
29 }
```

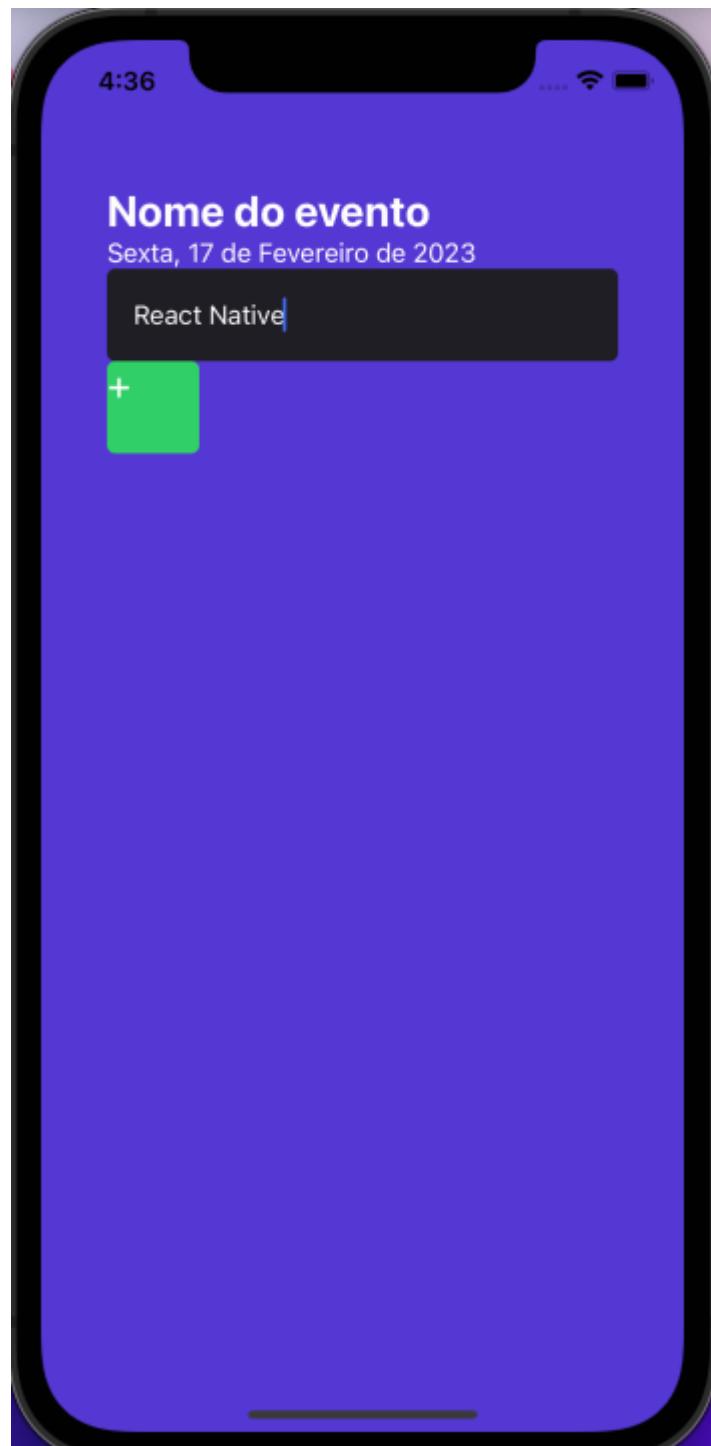
Edite o arquivo src/screens/Home/styles.ts

Insira da linha 31 até 35

The screenshot shows a code editor window with the file 'styles.ts' open. The file contains TypeScript code for a button component. The code defines two style objects: 'buttonText' and 'button'. The 'buttonText' object sets the color to '#FFF' and the font size to 24. The 'button' object sets the width to 56, height to 56, border radius to 5, and background color to '#31CF67'. A comment indicates an alternative background color '#FF872C'. The code is numbered from 27 to 38.

```
ts styles.ts Home U X
● myapp5p > src > screens > Home > ts styles.ts > ...
27   buttonText: {
28     color: '#FFF',
29     fontSize: 24,
30   },
31   button: {
32     width: 56,
33     height: 56,
34     borderRadius: 5,
35     backgroundColor: '#31CF67'
36     // backgroundColor: '#FF872C'
37   }
38 })
```

Salve as alterações e veja o resultado.



Observe que o sinal de mais ficou do lado esquerdo na parte superior. Para alinhá-lo

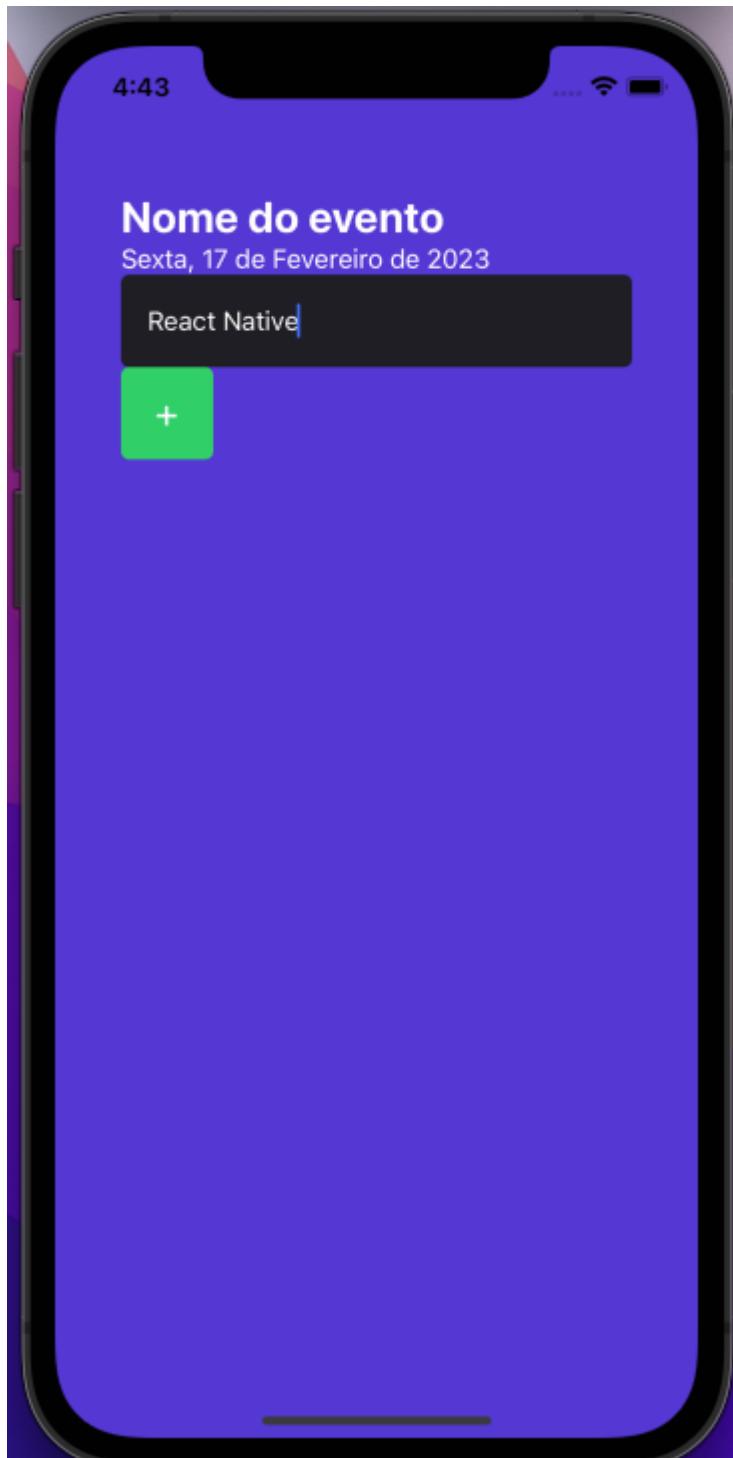
Edite o arquivo `src/screens/Home/styles.ts`

Insira as linhas 36 e 37

The screenshot shows a code editor window with the file 'styles.ts' open. The file contains TypeScript code for styling components. The code defines two objects: 'buttonText' and 'button'. The 'buttonText' object has properties for color ('#FFF') and font size (24). The 'button' object has properties for width (56), height (56), border radius (5), background color ('#31CF67'), justify content ('center'), and align items ('center'). The code is numbered from 27 to 39.

```
27  buttonText: {
28    color: '#FFF',
29    fontSize: 24,
30  },
31  button: {
32    width: 56,
33    height: 56,
34    borderRadius: 5,
35    backgroundColor: '#31CF67',
36    justifyContent: 'center',
37    alignItems: 'center'
38  }
39 })
```

Salve as alterações e veja o resultado.



O próximo passo é definir uma funcionalidade para o botão Adicionar. Quando ele for pressionado

Editie o arquivo src/screens/Home/index.tsx

Como será inserido mais de uma propriedade para o TouchableOpacity, deixe cada uma num linha. Insira a linha 23

```
index.tsx Home 1, U ●
myapp5p > src > screens > Home > index.tsx > ...
15    <TextInput
16      style={styles.input}
17      placeholder='Nome do participante'
18      placeholderTextColor='#6B6B6B'
19    />
20
21    <TouchableOpacity
22      style={styles.button}
23      onPress={handleAddParticipant}
24    >
25      <Text style={styles.buttonText}>
26        +
27      </Text>
28    </TouchableOpacity>
29
30  </View>
31)
32}
```

O erro que está gerando na linha 23 é pq a função handleAddParticipant não existe.

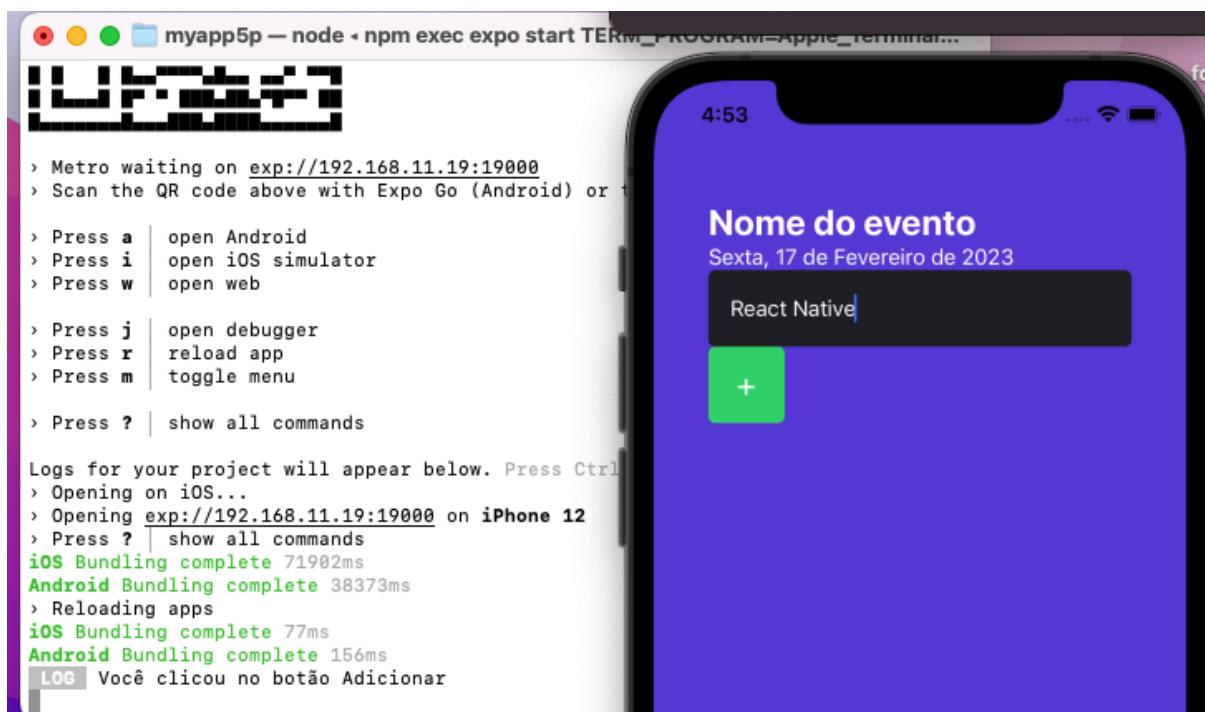
Criando a função handleAddParticipant

Insira da linha 7 até 9

```
index.tsx Home U X
myapp5p > src > screens > Home > index.tsx > Home

5  export default function Home() {
6
7      function handleAddParticipant() {
8          console.log('Você clicou no botão Adicionar')
9      }
10
11     return [
12         <View style={styles.container}>
13             <Text style={styles.eventName}>
```

Salve as alterações e faça o teste. Clique no botão Adicionar e veja que será exibida a mensagem no console do terminal



Criando o form

O flex direction funciona por padrão em coluna. Cada elemento que vc coloca fica abaixo do outro, o exemplo é o texto, input e o botao. Na aplicação o input deve ficar ao lado do botão adicionar e para isso eles devem ficar envolvidos por uma View

Edito o arquivo src/screens/Home/index.tsx

Insira a linha 19 e define a estilização form

Insira linha 34



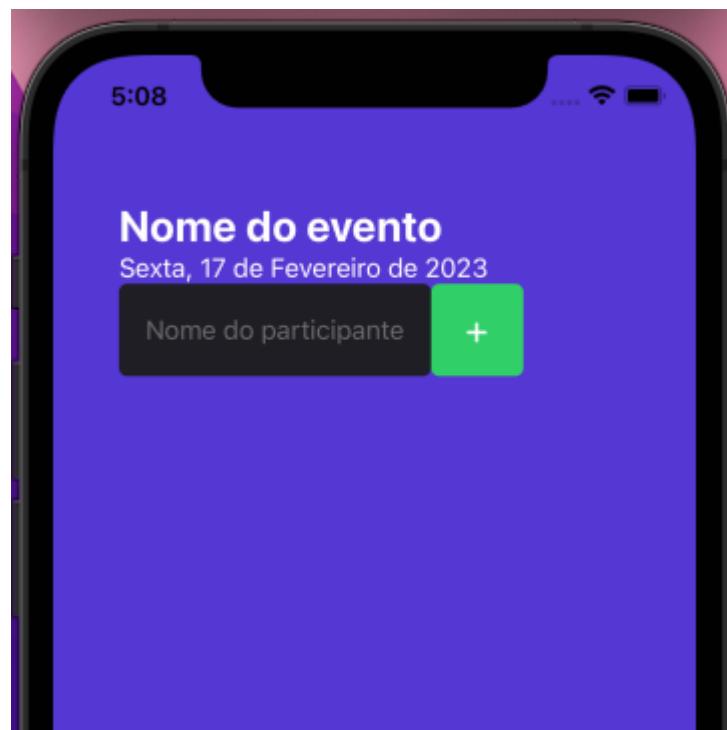
```
index.tsx Home 1, U •
myapp5p > src > screens > Home > index.tsx > Home
17     Sexta, 17 de Fevereiro de 2023
18     </Text>
19     <View style={styles.form}>
20         <TextInput
21             style={styles.input}
22             placeholder='Nome do participante'
23             placeholderTextColor='#6B6B6B'
24         />
25
26         <TouchableOpacity
27             style={styles.button}
28             onPress={handleAddParticipant}
29         >
30             <Text style={styles.buttonText}>
31                 +
32             </Text>
33         </TouchableOpacity>
34     </View>
35 </View>
36 )
```

Edito o arquivo src/screens/Home/styles.ts

Insira da linha 39 até 42

```
ts styles.ts Home U X
myapp5p > src > screens > Home > ts styles.ts > ...
31   button: {
32     width: 56,
33     height: 56,
34     borderRadius: 5,
35     backgroundColor: '#31CF67',
36     justifyContent: 'center',
37     alignItems: 'center'
38   },
39   form: {
40     width: '100%',
41     flexDirection: 'row',
42   }
43 })
```

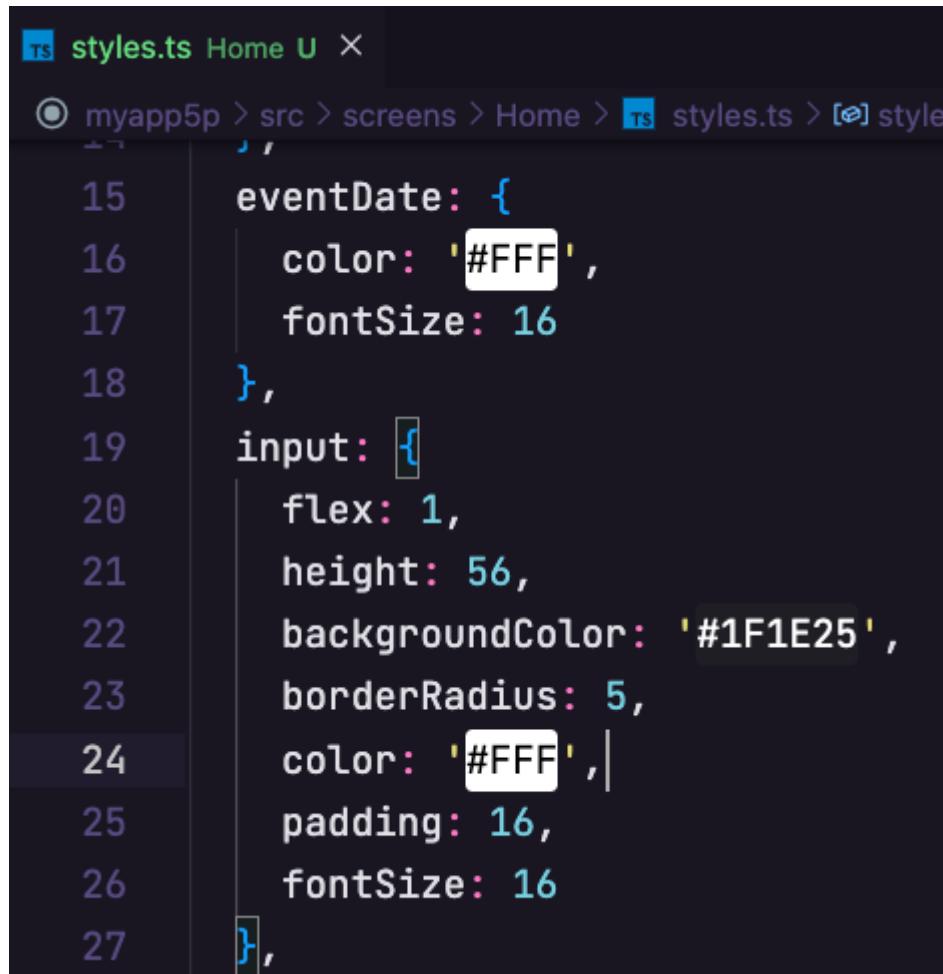
Salve as alterações e veja o resultado.



Observe que o input ficou ao lado do botão, porém a largura dele está do mesmo tamanho do conteúdo do placeholder

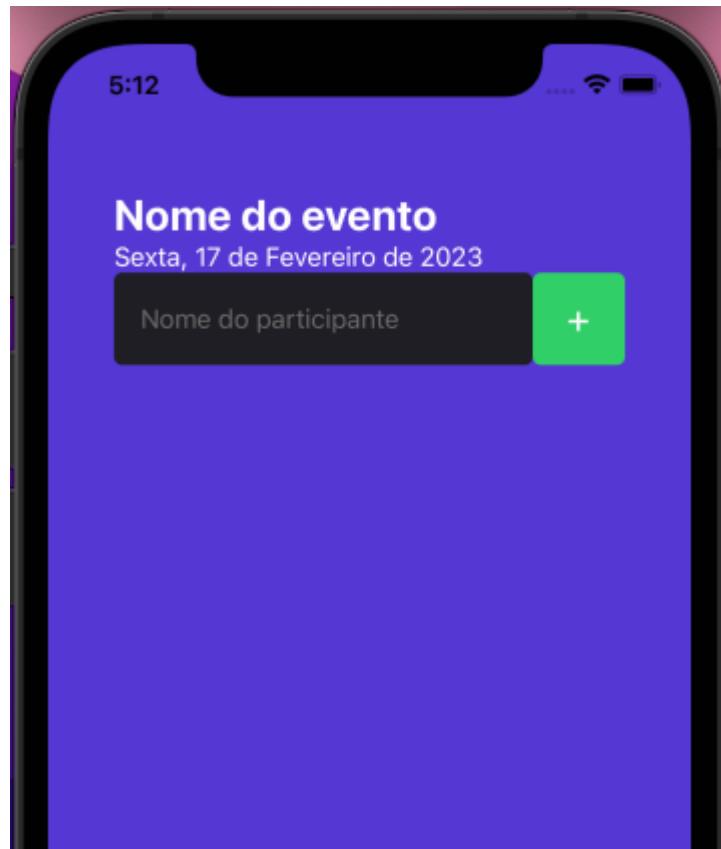
Edito o arquivo `src/screens/Home/styles.ts`

Insira a linha 20 para que o input ocupe todo o espaço disponível da tela.



```
 15   eventDate: {
 16     color: '#FFF',
 17     fontSize: 16
 18   },
 19   input: {
 20     flex: 1,
 21     height: 56,
 22     backgroundColor: '#1F1E25',
 23     borderRadius: 5,
 24     color: '#FFF',
 25     padding: 16,
 26     fontSize: 16
 27   },
}
```

Salve e veja o resultado.



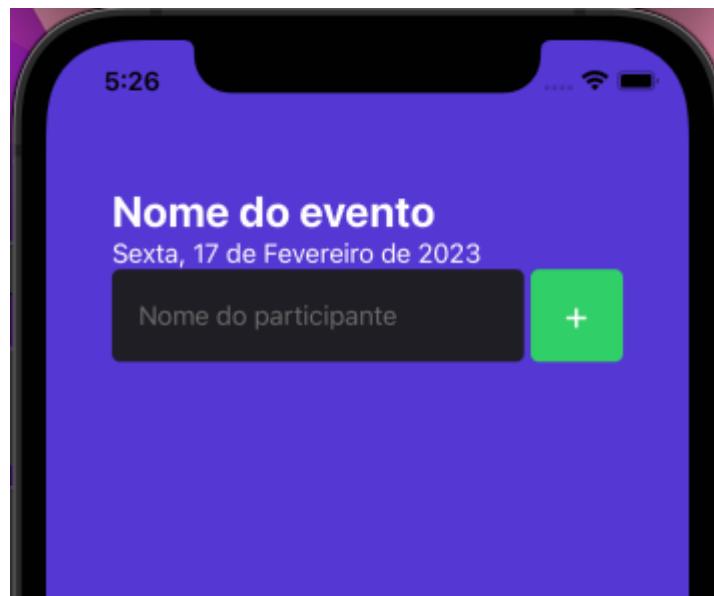
Observe que o input está colado no botão, para ajustar siga os passos

Edite o arquivo src/screens/Home/styles.ts

Insira a linha 27 para definir uma margem direita para o input.

```
TS styles.ts Home U X
● myapp5p > src > screens > Home > TS styles.ts > ...
19   input: {
20     flex: 1,
21     height: 56,
22     backgroundColor: '#1F1E25',
23     borderRadius: 5,
24     color: '#FFF',
25     padding: 16,
26     fontSize: 16,
27     marginRight: 4,
28   },
```

Salve e veja o resultado.



E para definir um espaçamento do input com a data, siga os passos

Edito o arquivo src/screens/Home/styles.ts

Insira as linhas 44 e 45

```
ts styles.ts Home U X
myapp5p > src > screens > Home > ts styles.ts >
41   form: {
42     width: '100%',
43     flexDirection: 'row',
44     marginTop: 36,
45     marginBottom: 42,
46   }
47 })
```

Salve e veja o resultado.



Componentização

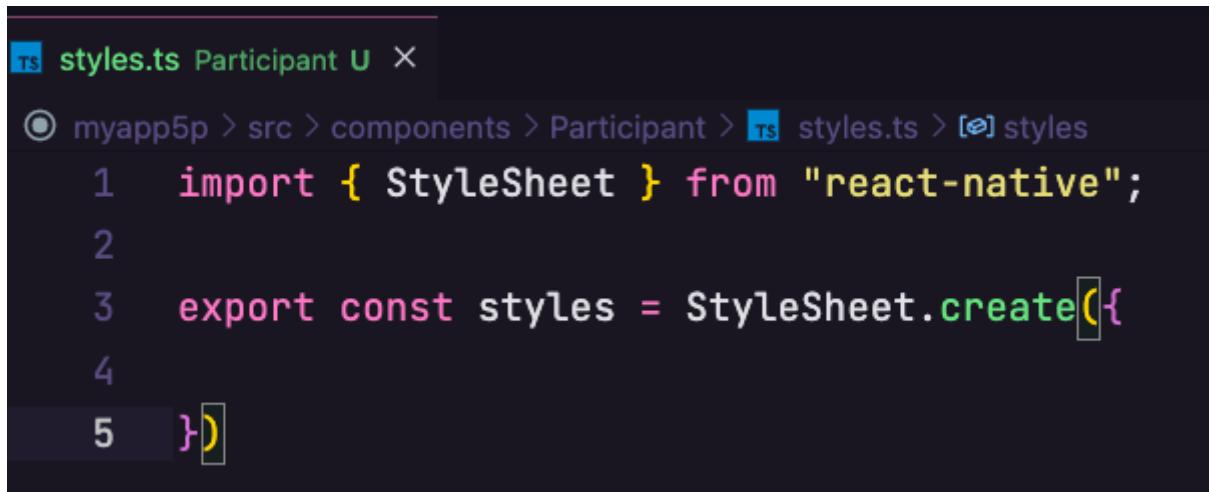
Criando Component Participant

Nesta parte será criado um componente para utilizá-lo em outras parte da aplicação. O componente a ser criado vai exibir o nome do participante e um botão para excluir.

Crie no botão novo arquivo e digite src/components/Participant/styles.ts

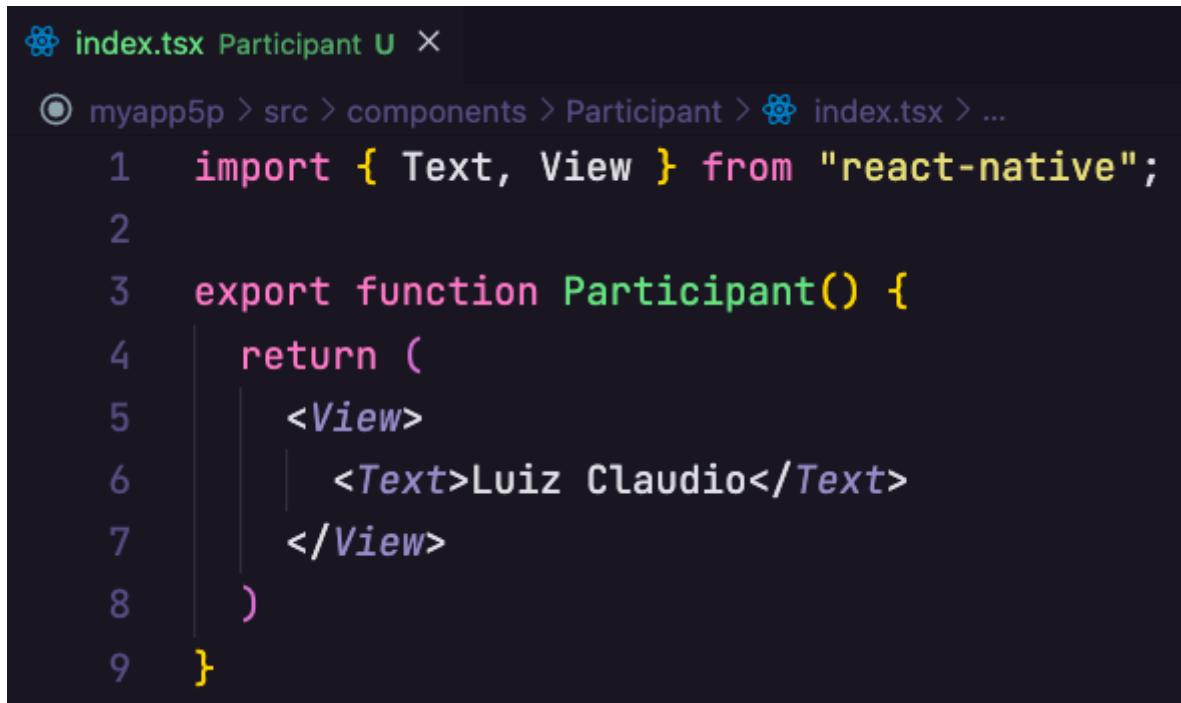
Neste exemplo será criado as pastas components e Participant e o arquivo styles.ts

Digite o conteúdo do arquivo styles.ts conforme abaixo



```
ts styles.ts Participant U X
① myapp5p > src > components > Participant > ts styles.ts > [o] styles
1 import { StyleSheet } from "react-native";
2
3 export const styles = StyleSheet.create({
4
5 })
```

Crie o arquivo src/components/Participant/index.tsx



```
blob index.tsx Participant U X
① myapp5p > src > components > Participant > blob index.tsx > ...
1 import { Text, View } from "react-native";
2
3 export function Participant() {
4     return (
5         <View>
6             <Text>Luiz Claudio</Text>
7         </View>
8     )
9 }
```

Importando o componente Participant na página Home

Edito o arquivo src/screens/Home/index.tsx

Insira a linha 2 para importar o componente Participant

```
index.tsx Home U X
myapp5p > src > screens > Home > index.tsx > Home
1 import { Text, View, TextInput, TouchableOpacity } from "react-native";
2 import { Participant } from "../../components/Participant";
3
4 import { styles } from './styles'
5
6 export default function Home() {
```

Insira a linha 37 para exibir o componente Participant, ele fica depois do form

```
index.tsx Home U X
myapp5p > src > screens > Home > index.tsx > ...
12     return (
13         <View style={styles.container}>
14             <Text style={styles.eventName}>
15                 Nome do evento
16             </Text>
17             <Text style={styles.eventDate}>
18                 Sexta, 17 de Fevereiro de 2023
19             </Text>
20         <View style={styles.form}>...
21             <Formik ...>
22                 <Form...>
23             </Form>
24         </View>
25     )
26
27     <Participant />
28
29 )
30 }
```

Salve e veja o resultado



Edite o arquivo src/components/Participant/styles.ts

Insira da linha 4 até 7

```
ts styles.ts Participant U X
● myapp5p > src > components > Participant > ts styles.ts > [o] styles
1 import { StyleSheet } from "react-native";
2
3 export const styles = StyleSheet.create({
4   container: {
5     width: '100%',
6     backgroundColor: '#1F1E25',
7   }
8 })
```

Edite o arquivo src/components/Participant/index.tsx

Insira a linha 3 para importar o styles

Altere a linha 7 para definir a estilização do container para a View

```
⚙️ index.tsx Participant U X
● myapp5p > src > components > Participant > ⚙️ index.tsx > ...
1 import { Text, View } from "react-native";
2
3 import { styles } from './styles'
4
5 export function Participant() {
6   return (
7     <View style={styles.container}>
8       <Text>Luiz Claudio</Text>
9     </View>
10  )
11}
```

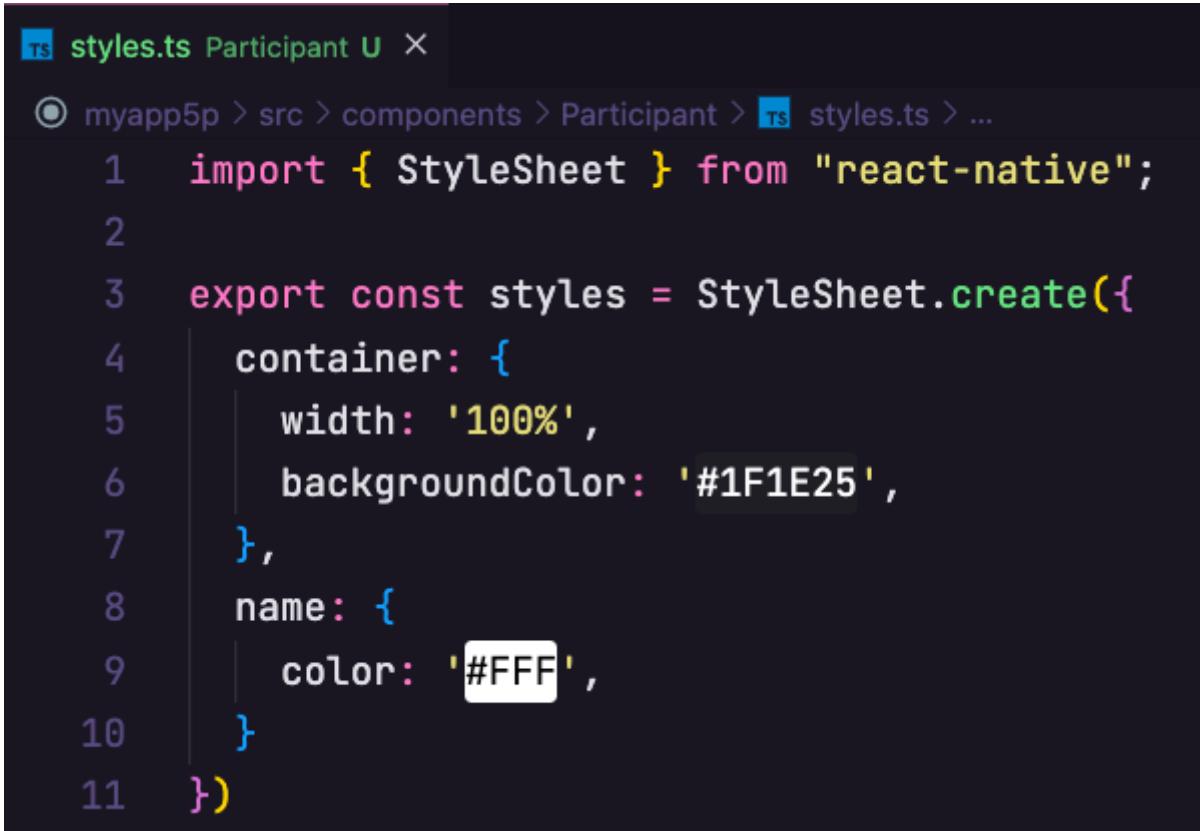
Salve as alterações e veja o resultado.



Estilizando o texto do participante

Edito o arquivo src/components/Participant/styles.ts

Insira da linha 8 até 10 para criar o componente de estilização name



The screenshot shows a code editor window with the file 'styles.ts' open. The file is part of a project structure under 'myapp5p'. The code defines a StyleSheet named 'styles' using the 'StyleSheet.create' method. It contains two style objects: 'container' and 'name'. The 'container' style has a width of '100%' and a background color of '#1F1E25'. The 'name' style has a color of '#FFF'. The code is numbered from 1 to 11.

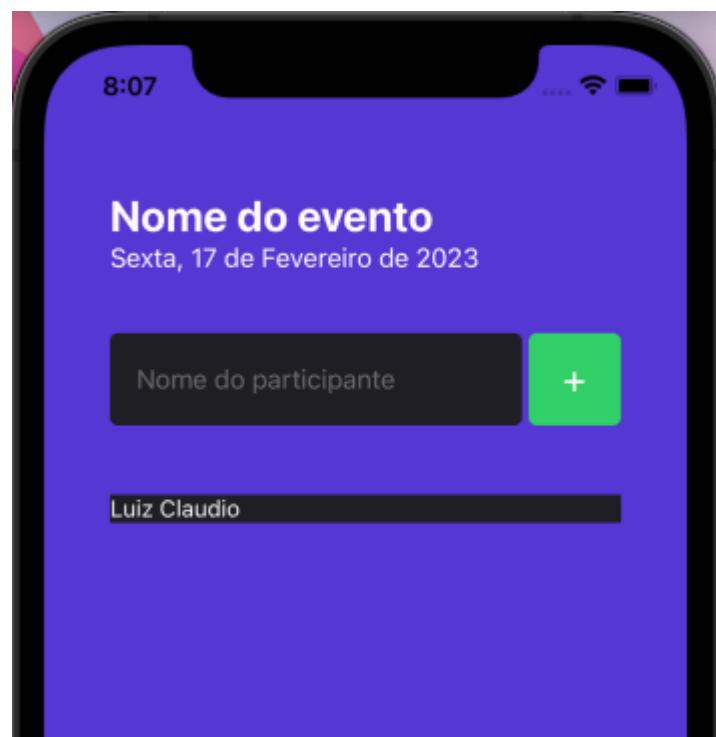
```
ts styles.ts Participant U X
① myapp5p > src > components > Participant > ts styles.ts > ...
1 import { StyleSheet } from "react-native";
2
3 export const styles = StyleSheet.create({
4   container: {
5     width: '100%',
6     backgroundColor: '#1F1E25',
7   },
8   name: {
9     color: '#FFF',
10 }
11 })
```

Edito o arquivo src/components/Participant/index.tsx

Altere a linha 8 para definir o componente de estilização name

```
index.tsx Participant U X
● myapp5p > src > components > Participant > index.tsx > ...
1 import { Text, View } from "react-native";
2
3 import { styles } from './styles'
4
5 export function Participant() {
6   return (
7     <View style={styles.container}>
8       <Text style={styles.name}>
9         Luiz Claudio
10      </Text>
11    </View>
12  )
13}
```

Salve as alterações e veja o resultado.



Criando o botão de excluir participante

Edite o arquivo src/components/Participant/index.tsx

Altere a linha 1 para importar o TouchableOpacity

```
index.tsx Participant 2, U X
myapp5p > src > components > Participant > index.tsx > Participant
1 import { Text, View, TouchableOpacity } from "react-native";
2
3 import { styles } from './styles'
4
5 export function Participant() {
```

Insira da linha 12 até 18 conforme abaixo.

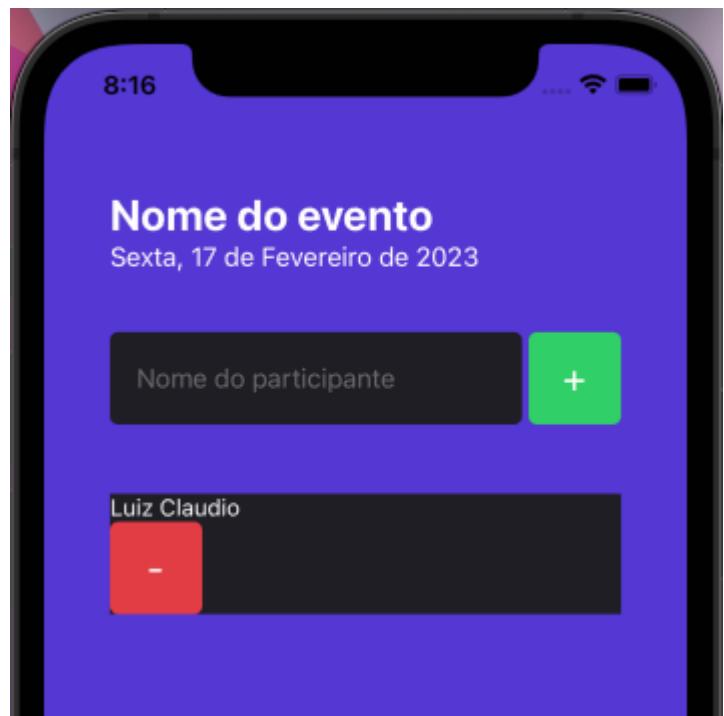
```
⚙️ index.tsx Participant 2, U X
● myapp5p > src > components > Participant > ⚙️ index.tsx > ...
5   export function Participant() {
6     return (
7       <View style={styles.container}>
8         <Text style={styles.name}>
9           Luiz Claudio
10          </Text>
11
12          <TouchableOpacity
13            style={styles.button}
14          >
15            <Text style={styles.buttonText}>
16              -
17            </Text>
18          </TouchableOpacity>
19        </View>
20      )
21    }
```

Edite o arquivo src/components/Participant/styles.ts

Insira da linha 11 até 22 conforme abaixo.

```
ts styles.ts Participant X
● myapp5p > src > components > Participant > ts styles.ts > ...
6     backgroundColor: '#1F1E25',
7 },
8     name: {
9         color: '#FFF',
10    },
11    buttonText: {
12        color: '#FFF',
13        fontSize: 24,
14    },
15    button: {
16        width: 56,
17        height: 56,
18        borderRadius: 5,
19        backgroundColor: '#E23C44',
20        justifyContent: 'center',
21        alignItems: 'center',
22    },
23 })
```

Salve as alterações e veja o resultado.



O próximo passo é ajustar os espaços

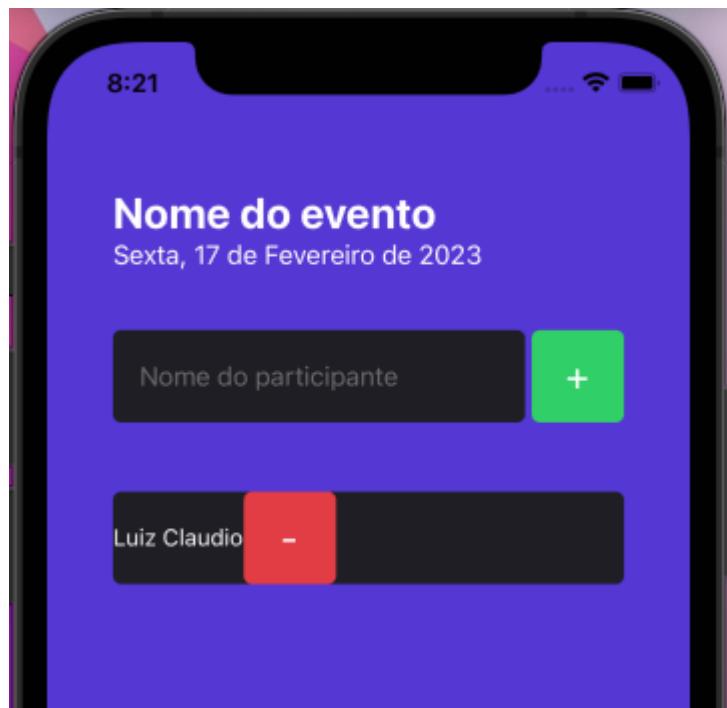
Edite o arquivo `src/components/Participant/styles.ts`

Insira da linha 7 até 9

```
ts styles.ts Participant U X
● myapp5p > src > components > Participant > ts styles.ts > ...
1 import { StyleSheet } from "react-native";
2
3 export const styles = StyleSheet.create({
4   container: {
5     width: '100%',
6     backgroundColor: '#1F1E25',
7     borderRadius: 5,
8     flexDirection: 'row',
9     alignItems: 'center'
10 },
11   name: {
12     color: '#FFF',
13   },

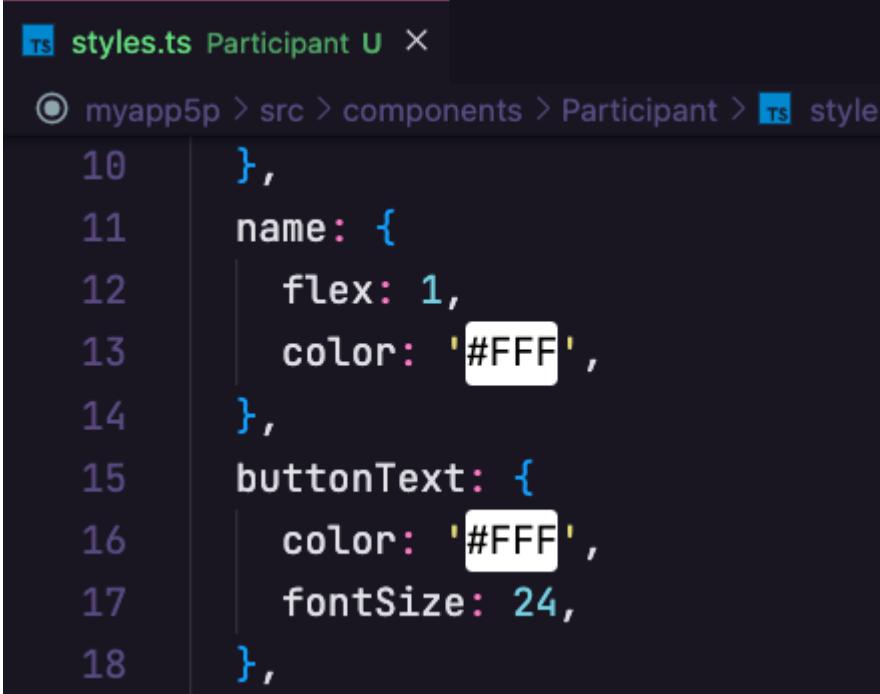
```

Salve as alterações e veja o resultado.



Edito o arquivo src/components/Participant/styles.ts

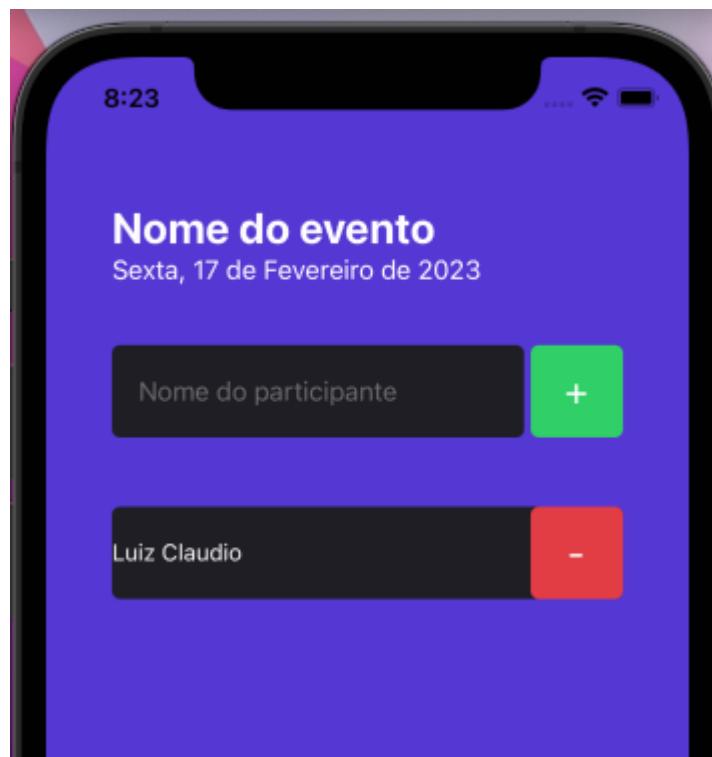
Insira a linha 12 para ocupar todo o espaço e deslocar o botão para direita



The screenshot shows a code editor window with the file 'styles.ts' open. The code is written in TypeScript and defines styles for a 'Participant' component. The relevant part of the code is as follows:

```
10  },
11  name: {
12    flex: 1,
13    color: '#FFF',
14  },
15  buttonText: {
16    color: '#FFF',
17    fontSize: 24,
18  },
```

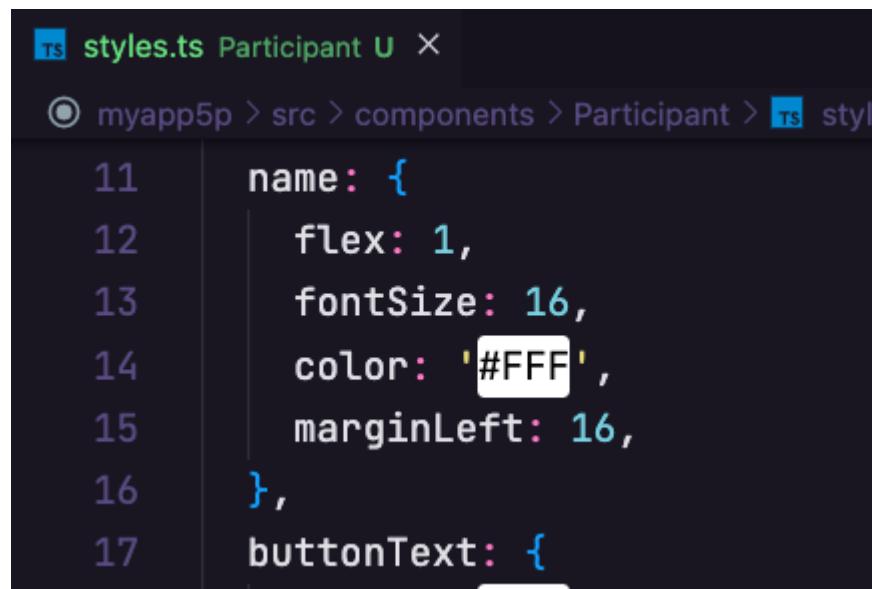
Salve as alterações e veja o resultado



O próximo passo é aumentar a fonte do nome do participante e inserir um espaçamento do lado esquerdo

Edito o arquivo src/components/Participant/styles.ts

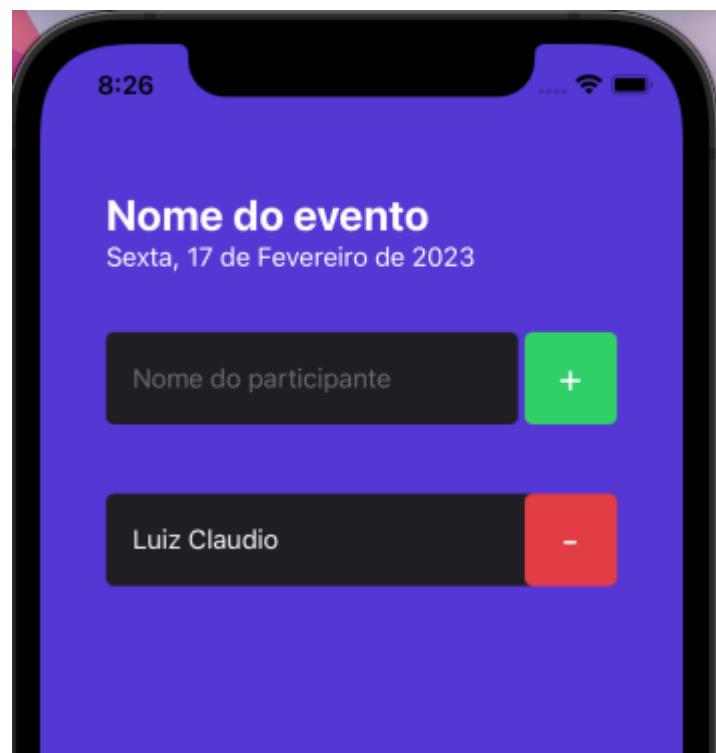
Insira as linhas 13 e 15



The screenshot shows a code editor window with the file 'styles.ts' open. The code is written in TypeScript and defines a style object for the 'name' property of the 'Participant' component. The 'fontSize' property is set to 16, and the 'color' property is set to '#FFF'. The 'marginLeft' property is also set to 16. The code is numbered from 11 to 17.

```
11  name: {
12    flex: 1,
13    fontSize: 16,
14    color: '#FFF',
15    marginLeft: 16,
16  },
17  buttonText: {
```

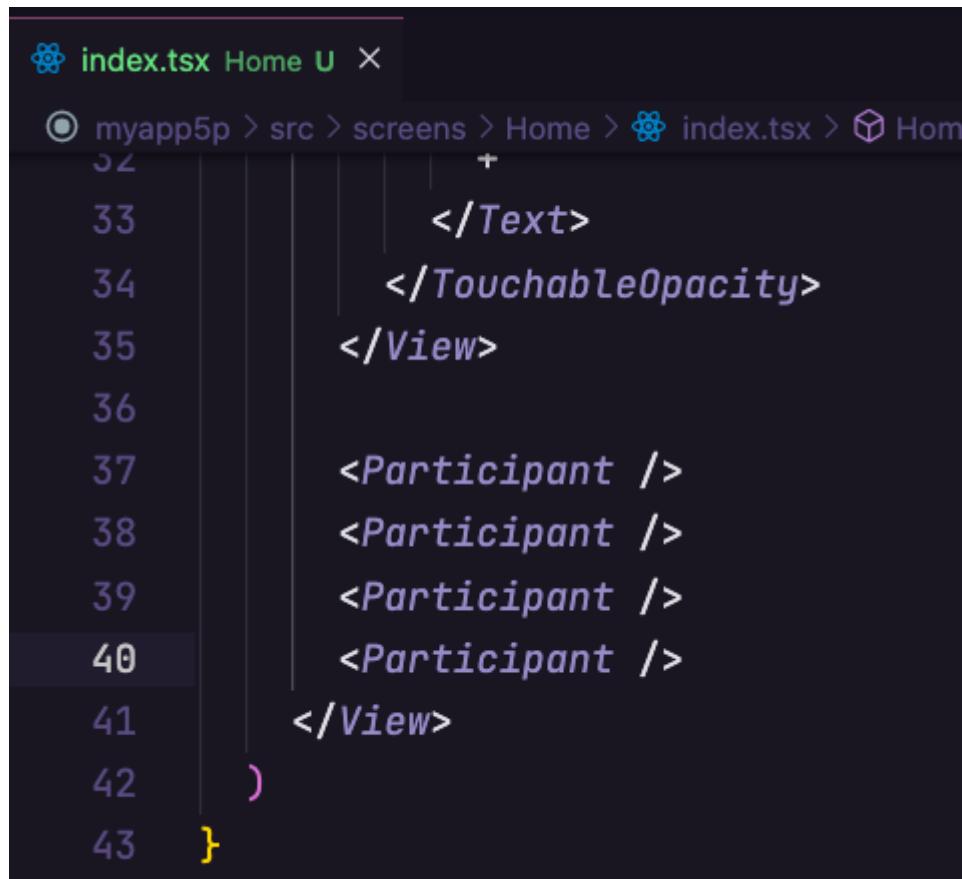
Salve as alterações e veja o resultado.



Outro recurso é aproveitar o componente Participant.

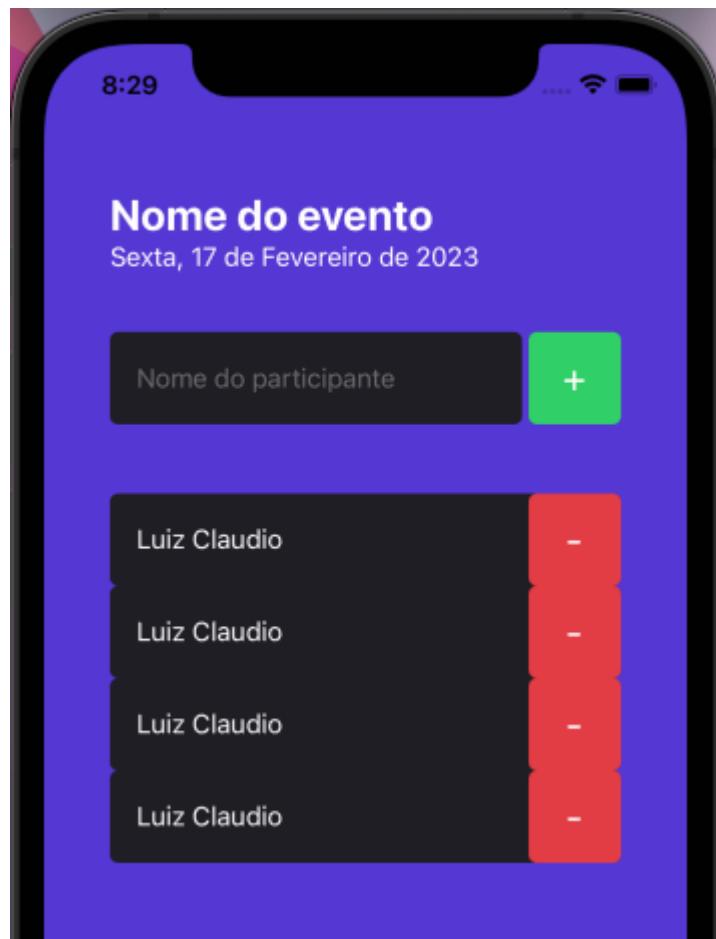
Edite o arquivo src/screens/Home/index.tsx

Insira da linha 38 até 40



```
index.tsx Home U X
myapp5p > src > screens > Home > index.tsx > Home
33           </Text>
34           </TouchableOpacity>
35           </View>
36
37           <Participant />
38           <Participant />
39           <Participant />
40           <Participant />
41           </View>
42       )
43 }
```

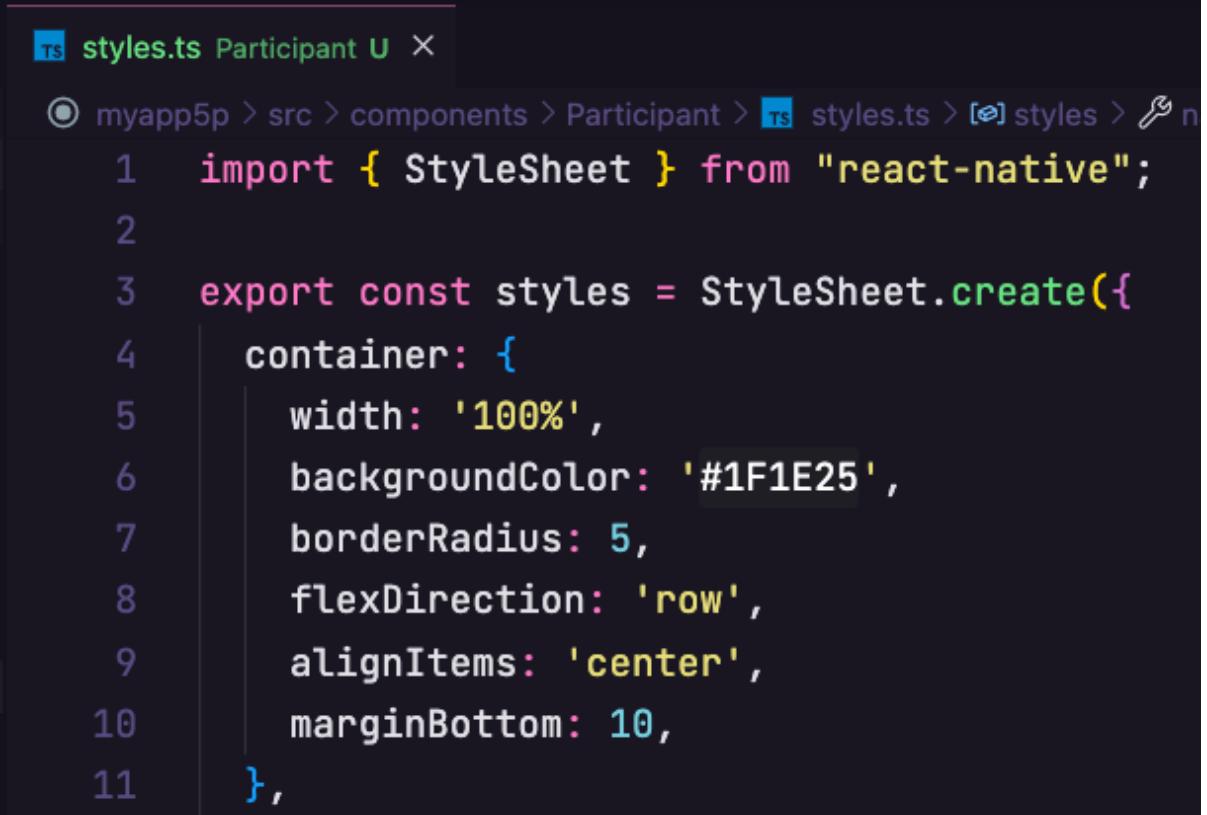
Salve as alterações e veja o resultado.



Observe que ficou muito próximo um componente do outro. Para ajustar insira um espaço na parte inferior dele.

Edite o arquivo src/components/Participant/styles.ts

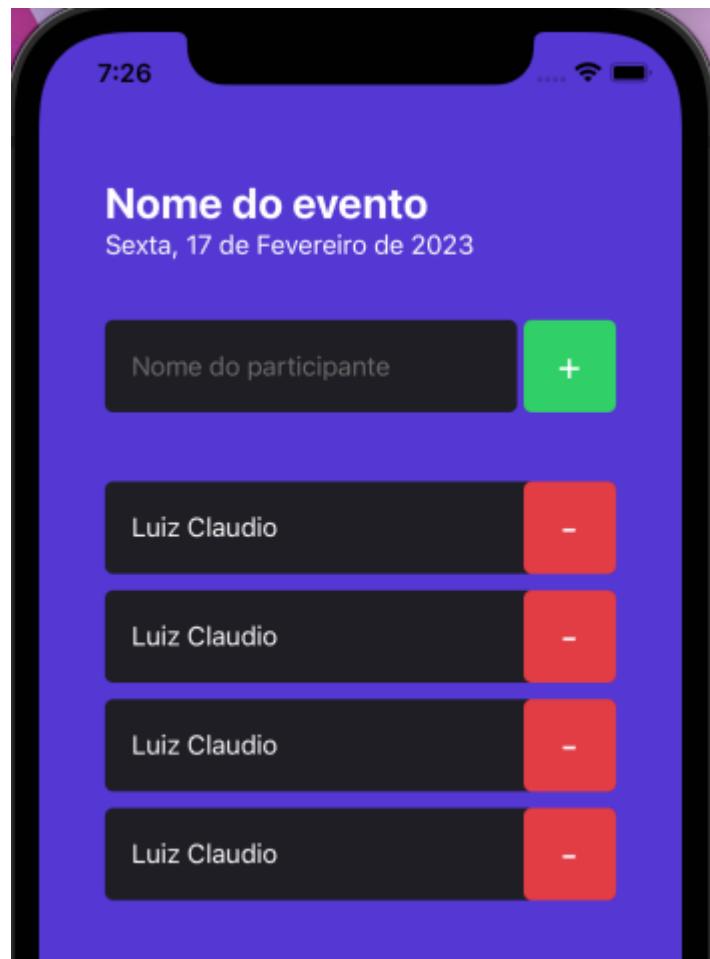
Insira a linha 10 conforme abaixo



The screenshot shows a code editor window with a dark theme. The file is named "styles.ts" and is located in the "Participant" component directory. The code defines a StyleSheet object with a single container style.

```
1 import { StyleSheet } from "react-native";
2
3 export const styles = StyleSheet.create({
4   container: {
5     width: '100%',
6     backgroundColor: '#1F1E25',
7     borderRadius: 5,
8     flexDirection: 'row',
9     alignItems: 'center',
10    marginBottom: 10,
11  },
12})
```

Salve e veja o resultado



Utilizando Propriedades nos componentes

Observe que ao exibir o nome do participante, ele está fixo. Você pode passar este nome como uma propriedade

Edito o arquivo src/screens/Home/index.tsx

Altere da linha 37 até 40 conforme abaixo, para passar a propriedade name para o componente Participant.

```
index.tsx Home 4, U ●
myapp5p > src > screens > Home > index.tsx > ...
  ↗ style={styles.buttonText}
    29   onPress={handleAddParticipant}
    30   >
    31     <Text style={styles.buttonText}>
    32       +
    33     </Text>
    34   </TouchableOpacity>
    35 </View>
    36
    37   <Participant name='Luiz Claudio' />
    38   <Participant name='Thiaguinho' />
    39   <Participant name='Maria' />
    40   <Participant name='Joaquim' />
    41 </View>
    42 )
  43 }
```

Está gerando o erro pois a propriedade name não existe no componente Participant. Definindo a propriedade name neste componente.

Edite o arquivo src/componentes/Participant/index.tsx

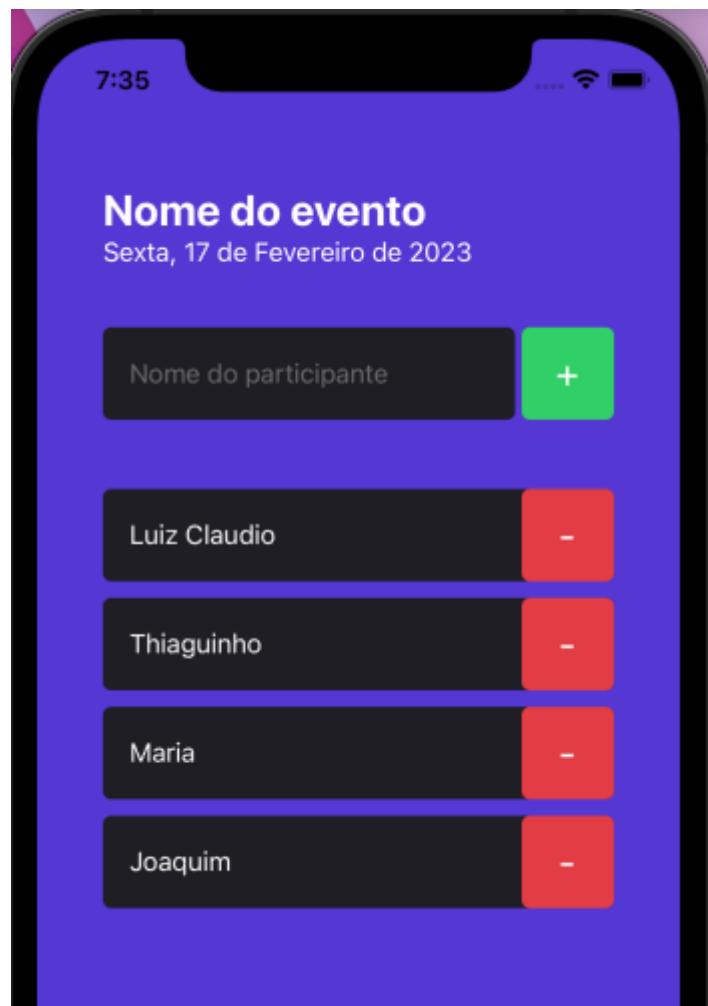
Altere a linha 5 para inserir o props

Altere a linha 9 para acessar a propriedade name dentro do props

```
index.tsx Participant 1, U X
myapp5p > src > components > Participant > index.tsx > Participant
```

```
5  export function Participant(props) {
6    return (
7      <View style={styles.container}>
8        <Text style={styles.name}>
9          {props.name}
10         </Text>
11
12         <TouchableOpacity
```

Salve as alterações e veja o resultado.

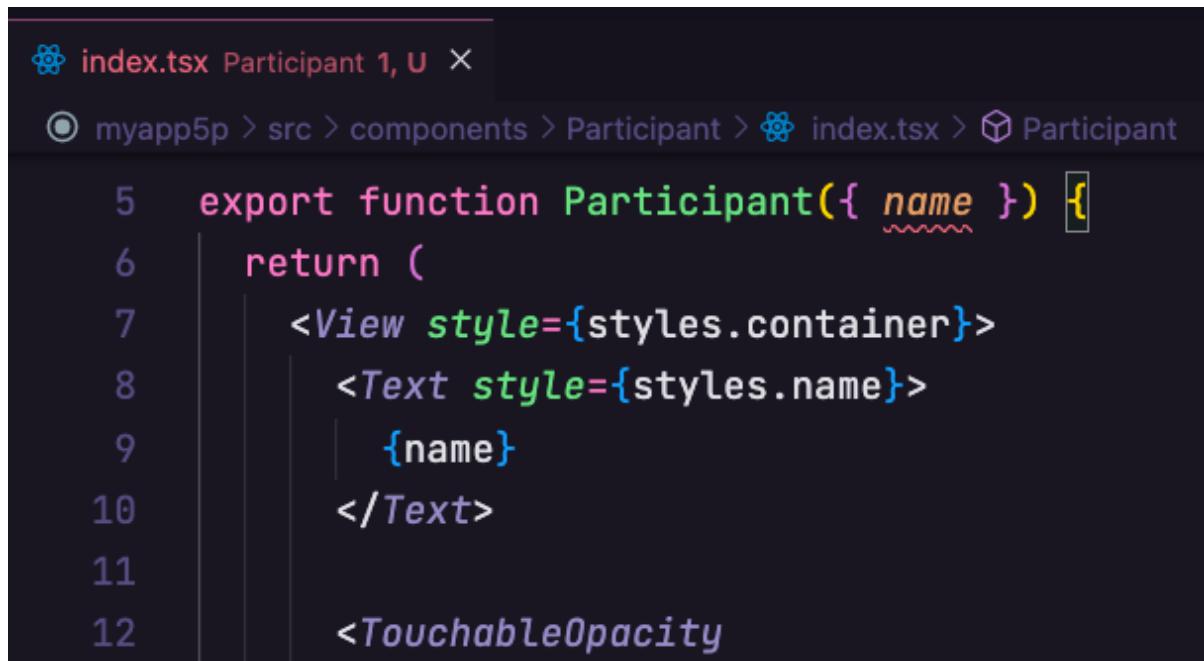


Outra forma para acessar propriedade name é desestruturar de props.

Edite o arquivo src/componentes/Participant/index.tsx

Altere a linha 5 para desestruturar a propriedade name

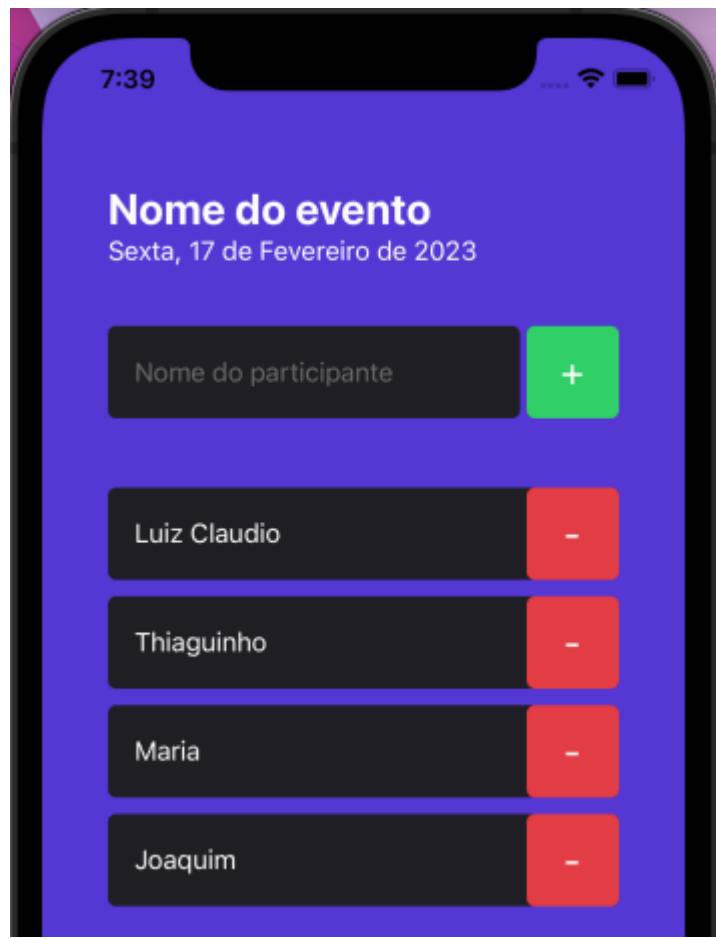
Altere a linha 9 para exibir a propriedade name.



The screenshot shows a code editor with the file 'index.tsx' open. The file path is 'myapp5p > src > components > Participant > index.tsx'. The code is as follows:

```
5  export function Participant({ name }) {  
6    return (  
7      <View style={styles.container}>  
8        <Text style={styles.name}>  
9          {name}  
10       </Text>  
11     <TouchableOpacity  
12       ...  
13     </TouchableOpacity>  
14   )  
15 }  
16  
17 const styles = {  
18   container: {  
19     flex: 1,  
20     padding: 10,  
21     alignItems: 'center',  
22     justifyContent: 'center',  
23     backgroundColor: '#f0f0f0'  
24   },  
25   name: {  
26     color: 'black',  
27     fontSize: 24,  
28     margin: 10  
29   }  
30 }  
31  
32 interface ParticipantProps {  
33   name: string  
34 }  
35  
36 const App = () => {  
37   return (  
38     <View style={styles.container}>  
39       <Text style={styles.name}>  
40         <Participant name="John" />  
41       </Text>  
42     </View>  
43   )  
44 }  
45  
46 export default App;
```

Salve as alterações e veja que continue funcionando da mesma forma.



Observe que está gerando um erro na linha 5 do arquivo **src/componentes/Participant/index.tsx**, pois está sendo utilizado o TS e ele exige uma tipagem. Para ajustar vc deve criar um tipo e define a propriedade name.

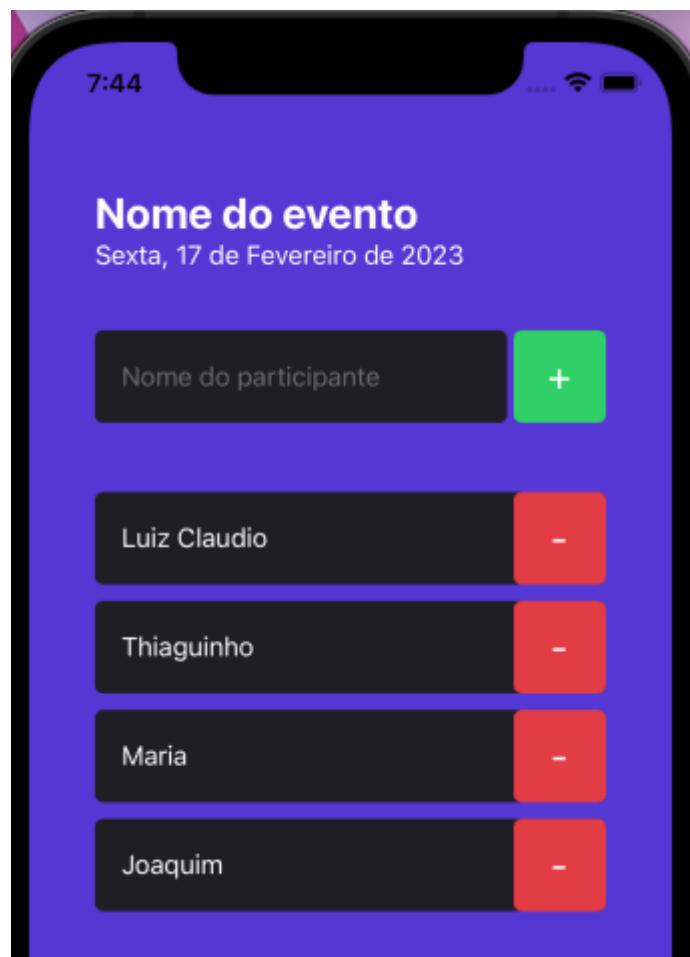
Edito o arquivo src/componentes/Participant/index.tsx

Insira da linha 5 até 7

Altere a linha 9 para desestruturar o name de Props

```
index.tsx Participant ✘ ×
myapp5p > src > components > Participant > index.tsx > Participant
1 import { Text, View, TouchableOpacity } from "react-native";
2
3 import { styles } from './styles'
4
5 type Props = {
6   name: string
7 }
8
9 export function Participant({ name }: Props) {
10   return (
11     <View style={styles.container}>
12       <Text style={styles.name}>
```

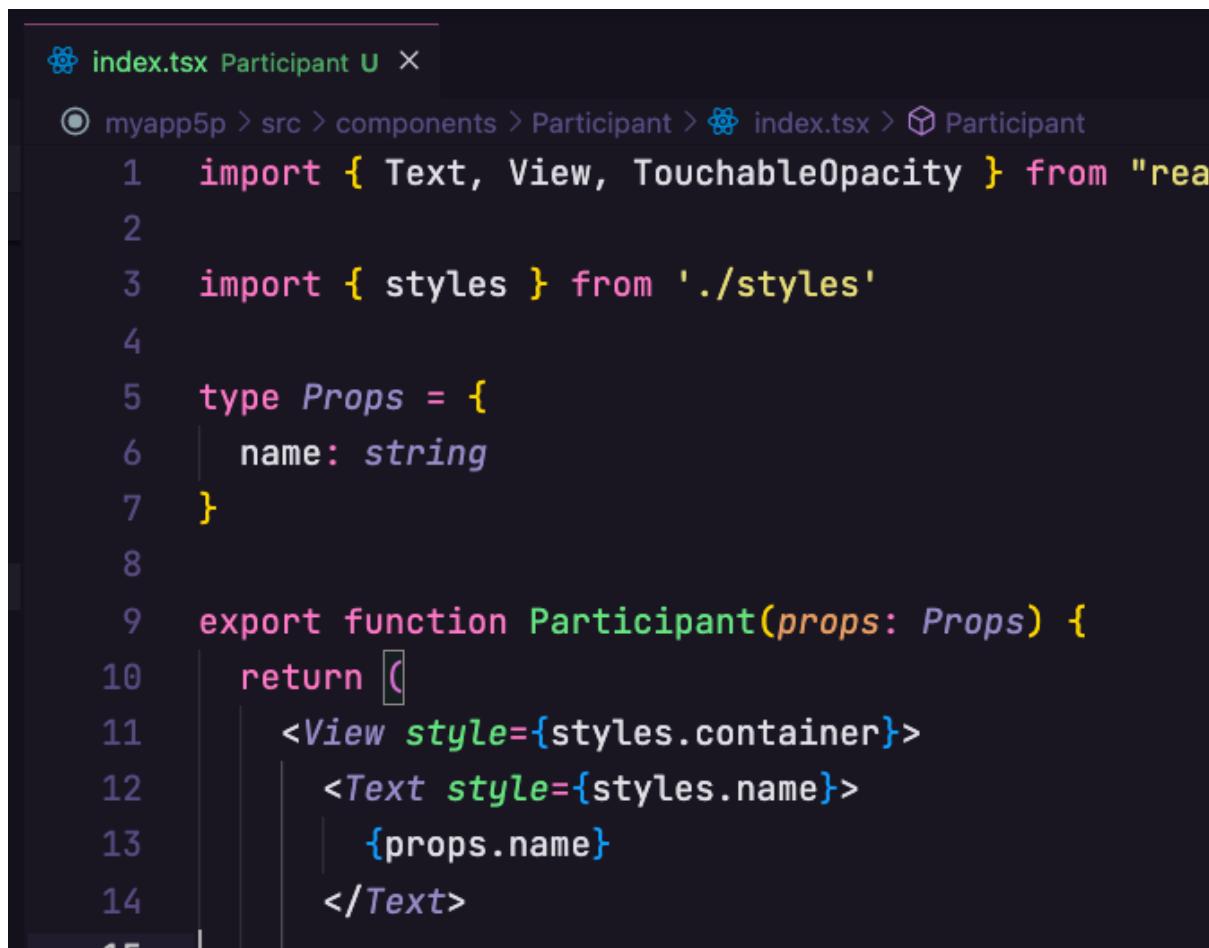
Salve as alterações e veja o resultado.



Se vc quiser utilizar o props, pode fazer da seguinte forma

Altere a linha 9 para definir o props

Altere a linha 13 para acessar a propriedade name de props



```
index.tsx Participant U X
myapp5p > src > components > Participant > index.tsx > Participant
1 import { Text, View, TouchableOpacity } from "react-native"
2
3 import { styles } from './styles'
4
5 type Props = {
6   name: string
7 }
8
9 export function Participant(props: Props) {
10   return [
11     <View style={styles.container}>
12       <Text style={styles.name}>
13         {props.name}
14       </Text>
15     </View>
16   ]
17 }
```

Salve as alterações e veja que tudo continua funcionando normalmente.

Função Remover

Para implementar a função de remover, passe-a como propriedade para o componente Participant

Edito o arquivo src/screens/Home/index.tsx

Insira da linha 12 até 14

```
index.tsx Home U X
myapp5p > src > screens > Home > index.tsx > Home
6  export default function Home() {
7
8      function handleAddParticipant() {
9          console.log('Você clicou no botão Adicionar')
10     }
11
12     function handleRemoveParticipant() {
13         console.log('Você clicou em remover o participante')
14     }
15
16     return [
```

Insira a linha 43 para passar a propriedade onRemove com a função handleRemoveParticipant para o componente Participant.

```
index.tsx Home 1, U ●
myapp5p > src > screens > Home > index.tsx > ...
37             </Text>
38             </TouchableOpacity>
39         </View>
40
41         <Participant
42             name='Thiaguinho'
43             onRemove={handleRemoveParticipant} />
44         </View>
45     )
46 }
```

Observe que está gerando um erro na linha 43 pois a propriedade onRemove não existe no componente Participant.

Edite o arquivo src/componentes/Participant/index.tsx

Insira a linha 7 para definir a propriedade onRemove como uma função

Altere a linha 10 para desestruturar as propriedades name e onRemove

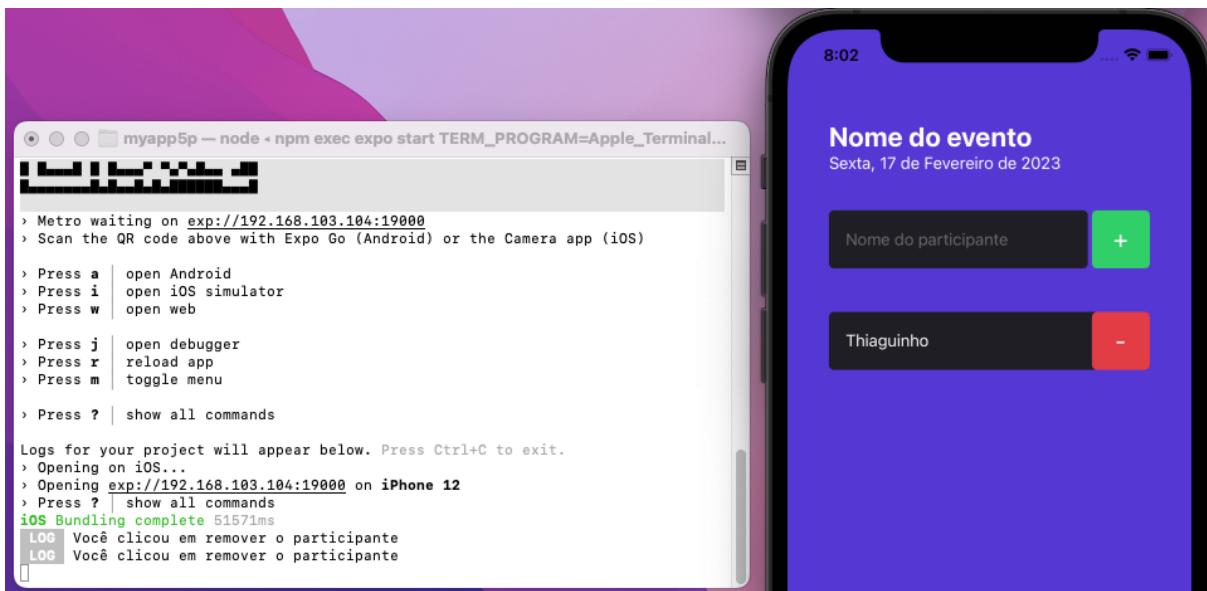
```
index.tsx Participant U ×
myapp5p > src > components > Participant > index.tsx > Participant
1 import { Text, View, TouchableOpacity } from "react-native";
2
3 import { styles } from './styles'
4
5 type Props = {
6   name: string;
7   onRemove: () => void;
8 }
9
10 export function Participant({ name, onRemove }: Props) {
11   return (
12     <View style={styles.container}>
```

Insira a linha 19 para que a propriedade onRemove seja executada ao pressionar o botão (onPress)

```
index.tsx Participant U ×
myapp5p > src > components > Participant > index.tsx > Participant
10 export function Participant({ name, onRemove }: Props) {
11   return (
12     <View style={styles.container}>
13       <Text style={styles.name}>
14         {name}
15       </Text>
16
17       <TouchableOpacity
18         style={styles.button}
19         onPress={onRemove}
20       >
21         <Text style={styles.buttonText}>
```

Observe que os erros sumiram. Salve as alterações e veja resultado. Quando clicar no botão remover, deve exibir uma mensagem Você clicou em remover o

participante no console conforme print abaixo.



Edite o arquivo src/screens/Home/index.tsx

Observe que na linha 43 foi chamada a função `handleRemoveParticipant` sem abrir e fechar parêntese (), pois ela não está recebendo nenhum parâmetro.

```
index.tsx Home U X
● myapp5p > src > screens > Home > index.tsx > ...
36
37           +
38           </Text>
39           </TouchableOpacity>
40
41           <Participant
42             name='Thiaguinho'
43             onRemove={handleRemoveParticipant} />
44           </View>
45         )
46     }
```

A screenshot of a code editor showing the file 'index.tsx' under the 'Home' folder. The code is a React Native component. Line 43 contains the line `onRemove={handleRemoveParticipant}`. The closing parenthesis ')' is on the same line as the closing tag of the `<View>` element, indicating that the function call does not have parentheses around it.

Altere a linha 43 para chamar a função handleRemoveParticipant passando como parâmetro o nome do participante.

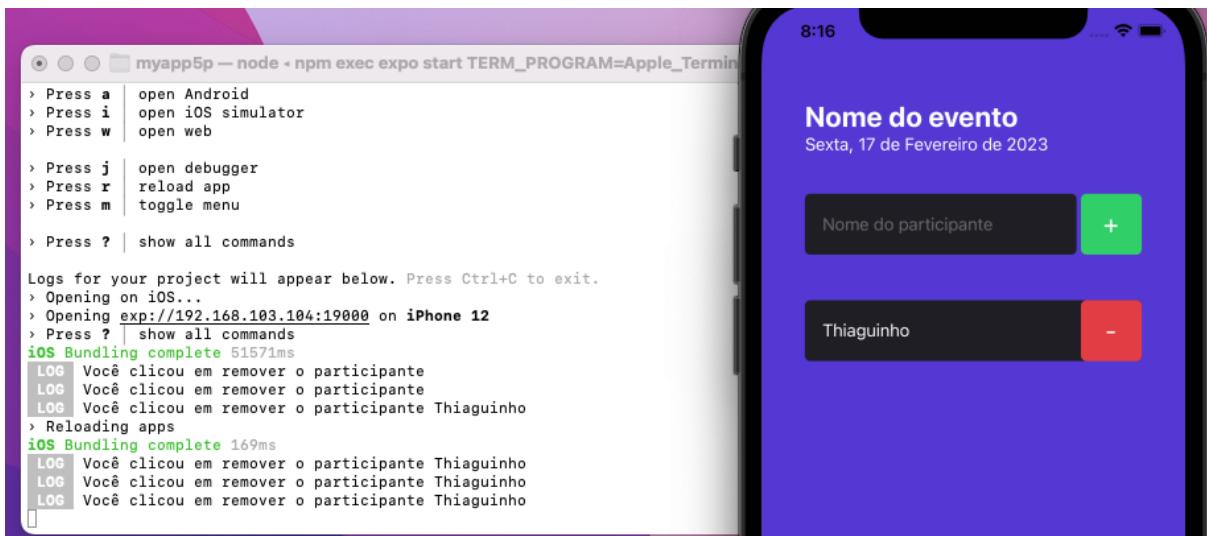
```
index.tsx Home 1, U ●
myapp5p > src > screens > Home > index.tsx > ...
36      +
37      </Text>
38      </TouchableOpacity>
39    </View>
40
41    <Participant
42      name='Thiaguinho'
43      onRemove={() => handleRemoveParticipant('Thiaguinho')} />
44  </View>
45)
46}
```

Altere a linha 1 para definir que a função handleRemoveParticipant recebe uma nome como string

Altere a linha 13 para exibir o conteúdo da variável name

```
index.tsx Home U ×
myapp5p > src > screens > Home > index.tsx > Home
6  export default function Home() {
7
8    function handleAddParticipant() {
9      console.log('Você clicou no botão Adicionar')
10     }
11
12    function handleRemoveParticipant(name: string) {
13      console.log(`Você clicou em remover o participante ${name}`)
14    }
15
16    return ()
```

Salve as alterações e veja o resultado. Ao clicar no botão remover será exibida a seguinte mensagem no console. Você clicou em remover o participante Thiaguinho



Ativando rolagem na tela

O próximo passo é listar os participantes utilizando um ScrollView

Edito o arquivo src/screens/Home/index.tsx

Insira a linha 7 até 19 para criar um array com os participants

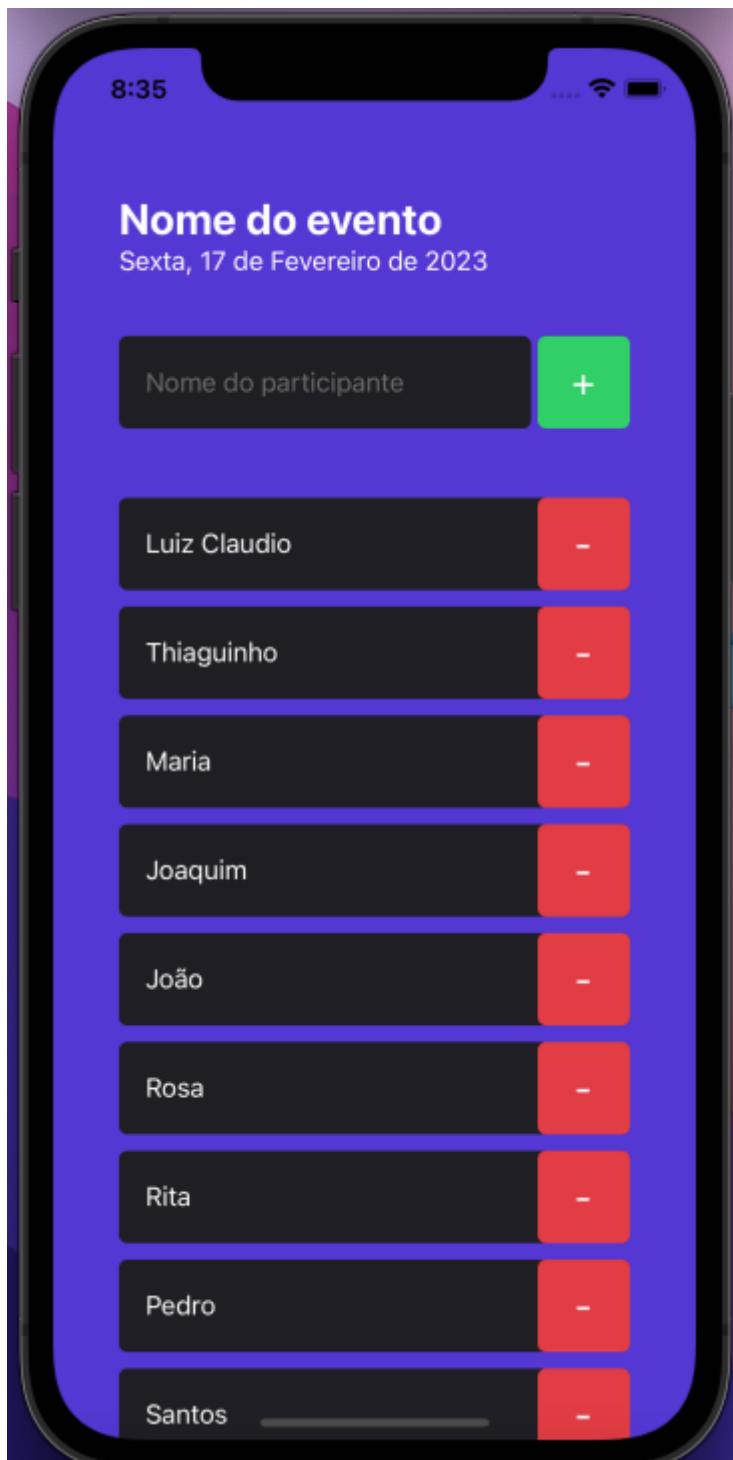
```
⚙️ index.tsx Home U X
● myapp5p > src > screens > Home > ⚙️ index.tsx > ⚡ Home > ⚡ handleRemoveParticip
6   export default function Home() {
7     const participants = [
8       { id: 1, name: 'Luiz Claudio' },
9       { id: 2, name: 'Thiaguinho' },
10      { id: 3, name: 'Maria' },
11      { id: 4, name: 'Joaquim' },
12      { id: 5, name: 'João' },
13      { id: 6, name: 'Rosa' },
14      { id: 7, name: 'Rita' },
15      { id: 8, name: 'Pedro' },
16      { id: 9, name: 'Santos' },
17      { id: 10, name: 'Carla' },
18      { id: 11, name: 'Marcia' }
19    ]
20
21    function handleAddParticipant() {
22      console.log('Você clicou no botão Adicionar')
23    }

```

Insira da linha 54 até 61 para exibir os participantes do array participant

```
index.tsx Home U X
myapp5p > src > screens > Home > index.tsx > ...
48     <Text style={styles.buttonText}>
49         +
50     </Text>
51     </TouchableOpacity>
52   </View>
53
54 {
55     participants.map(participant => (
56         <Participant
57             key={participant.id}
58             name={participant.name}
59             onRemove={() => handleRemoveParticipant('Thiaguino')} />
60     ))
61 }
62
63   </View>
64 )
65 }
```

Salve as alterações e veja o resultado.



Observe que não foi exibido todos os nomes, para ajustar siga os passos

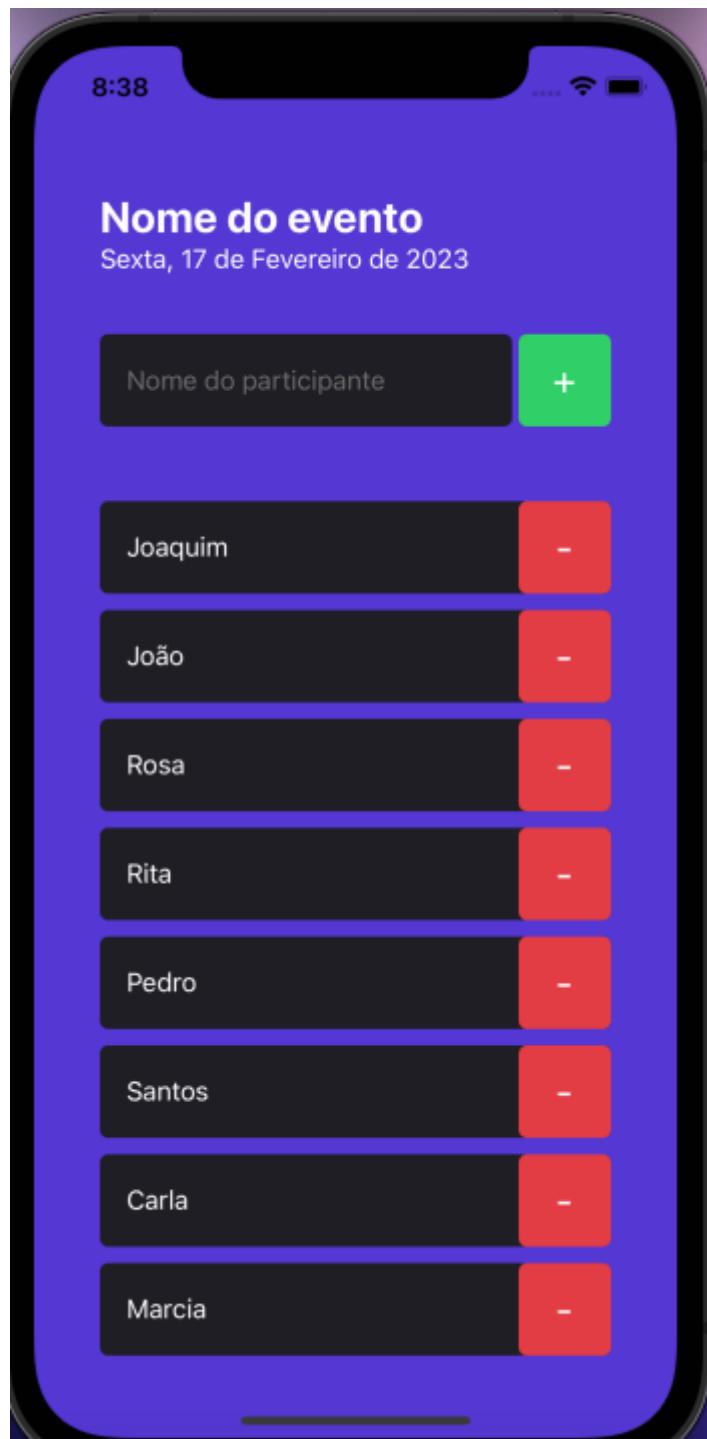
Altere a linha 1 para importar a ScrollView

```
index.tsx Home U X
① myapp5p > src > screens > Home > index.tsx > Home > participants
  1 import { Text, View, TextInput, TouchableOpacity, ScrollView } from "react-native";
  2 import { Participant } from "../../components/Participant";
  3
  4 import { styles } from './styles'
  5
  6 export default function Home() {
```

Insira as linhas 54 e 63 para ativar a rolagem

```
index.tsx Home U X
① myapp5p > src > screens > Home > index.tsx > ...
  47
  48
  49
  50      </Text>
  51      </TouchableOpacity>
  52    </View>
  53
  54    <ScrollView>
  55    {
  56      participants.map(participant => (
  57        <Participant
  58          key={participant.id}
  59          name={participant.name}
  60          onRemove={() => handleRemoveParticipant('Thiaguino')} />
  61      ))
  62    }
  63  </ScrollView>
  64
  65  </View>
  66)
  67}
```

Salve as alterações e veja o resultado. Observe que agora consegue fazer a rolagem dos nomes.



Observe que foi exibida uma barra de rolagem vertical e para desativá-la, altere a linha 54 para inserir a propriedade showsVerticalScrollIndicator conforme abaixo

```
index.tsx Home U X
myapp5p > src > screens > Home > index.tsx > ...
50     </Text>
51     </TouchableOpacity>
52   </View>
53
54   <ScrollView showsVerticalScrollIndicator={false}>
55     {
56       participants.map(participant => (
57         <Participant
58           key={participant.id}
59           name={participant.name}
60           onRemove={() => handleRemoveParticipant('Thiaguino')} />
61       ))
62     }
63   </ScrollView>
64
65 </View>
66
67 }
```

Salve as alterações e veja o resultado. Observe que a barra de rolagem vertical do lado direito ficou oculta.

Utilizando a FlatList para trabalhar com listas

Altere a linha 1 para importar a FlatList no lugar da ScrollView conforme abaixo

```
index.tsx Home 2, U ●
src > screens > Home > index.tsx > Home
1 import { Text, View, TextInput, TouchableOpacity, FlatList } from "react-native";
2 import { Participant } from "../components/Participant";
3
4 import { styles } from './styles'
```

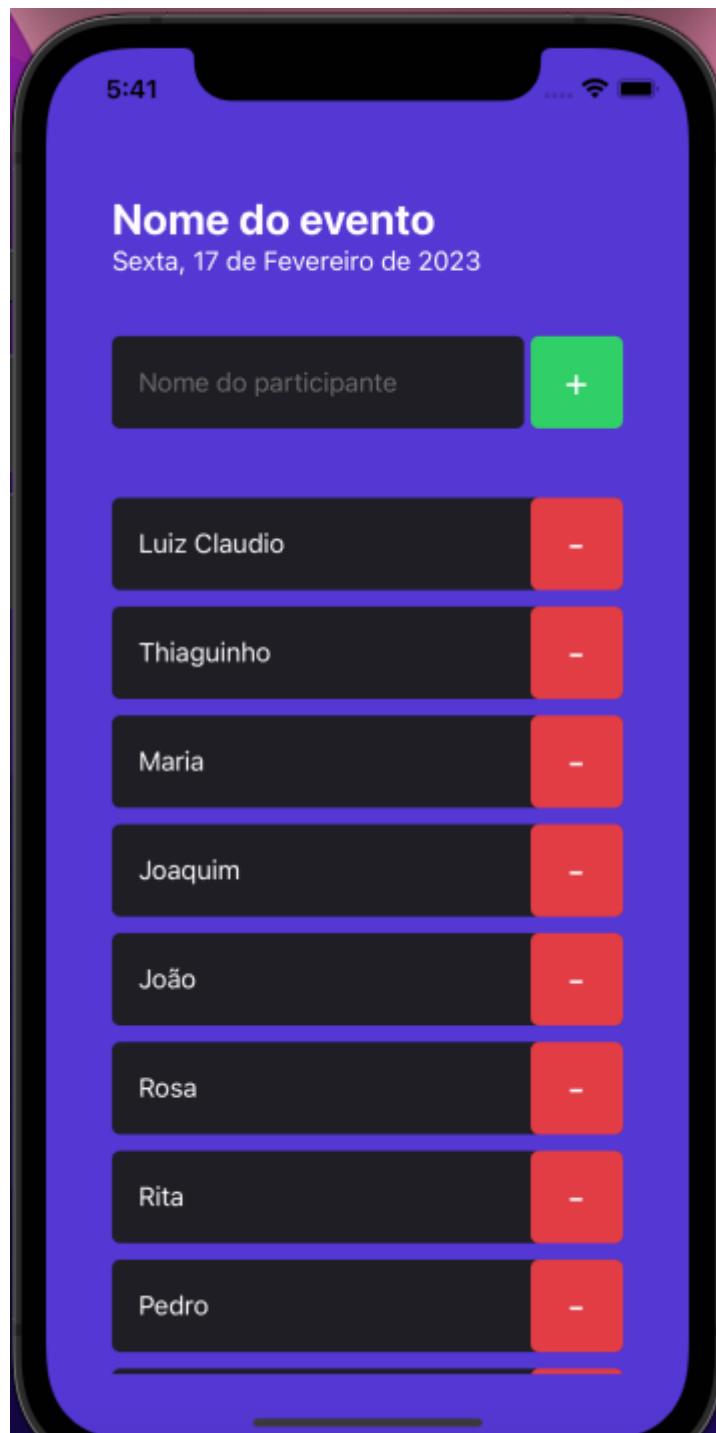
Apague as linhas 54 e 63 pois a importação da ScrollView foi apagada, salve as alterações para o App continuar funcionando, porém não tem a rolagem.

```
index.tsx Home 2, U ●
src > screens > Home > index.tsx > Home
50      </Text>
51      </TouchableOpacity>
52    </View>
53
54    <ScrollView showsVerticalScrollIndicator={false}>
55    {
56      participants.map(participant => (
57        <Participant
58          key={participant.id}
59          name={participant.name}
60          onRemove={() => handleRemoveParticipant('Thiaguino')} />
61      ))
62    }
63  </ScrollView>
64
65  </View>
66)
67}
```

Insira da linha 54 até 64.

```
index.tsx Home U X
src > screens > Home > index.tsx > ...
49      +
50      </Text>
51    </TouchableOpacity>
52  </View>
53
54  <FlatList
55    data={participants}
56    keyExtractor={item => item.id}
57    renderItem={({ item }) => (
58      <Participant
59        name={item.name}
60        onRemove={() => handleRemoveParticipant('Thiaguino')}
61      />
62    )}
63    showsVerticalScrollIndicator={false}
64  />
65  </View>
66)
67}
```

Salve as alterações e veja que o app continua funcionando e com rolagem.



Você pode adicionar uma propriedade na FlatList quando a lista estiver vazia.

Veja as alterações

Altere a linha 55 para deixar um array vazio

Insira da linha 64 até 68

```
index.tsx Home 3, U •
src > screens > Home > index.tsx > ...
54     <FlatList
55       data={[]}
56       keyExtractor={item => item.id}
57       renderItem={({ item }) => (
58         <Participant
59           name={item.name}
60           onRemove={() => handleRemoveParticipant('Thiaguino')}
61         />
62       )}
63       showsVerticalScrollIndicator={false}
64       ListEmptyComponent={() => (
65         <Text style={styles.listEmptyText}>
66           Adicione participante a sua lista de presença
67         </Text>
68       )}
69     />
70   </View>
71 )
72 }
```

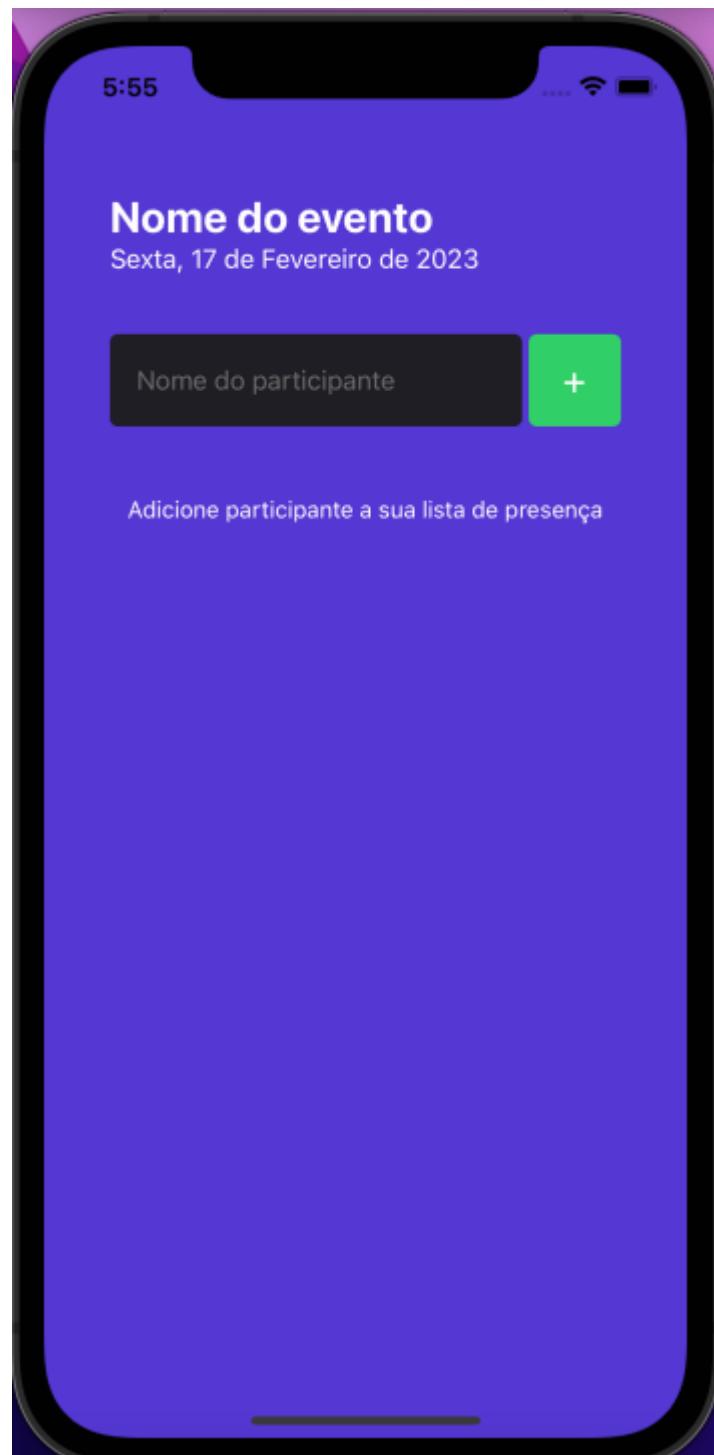
Criando o componente listEmptyText

Edite o arquivo src/screens/home/style.ts

Insira da linha 47 até 51

```
styles.ts Home U X
src > screens > Home > styles.ts > ...
41   form: {
42     width: '100%',
43     flexDirection: 'row',
44     marginTop: 36,
45     marginBottom: 42,
46   },
47   listEmptyText: {
48     color: '#FFF',
49     fontSize: 14,
50     textAlign: 'center',
51   }
52 })
```

Salve as alterações e veja o resultado. Como a lista está vazia, exibe a mensagem.

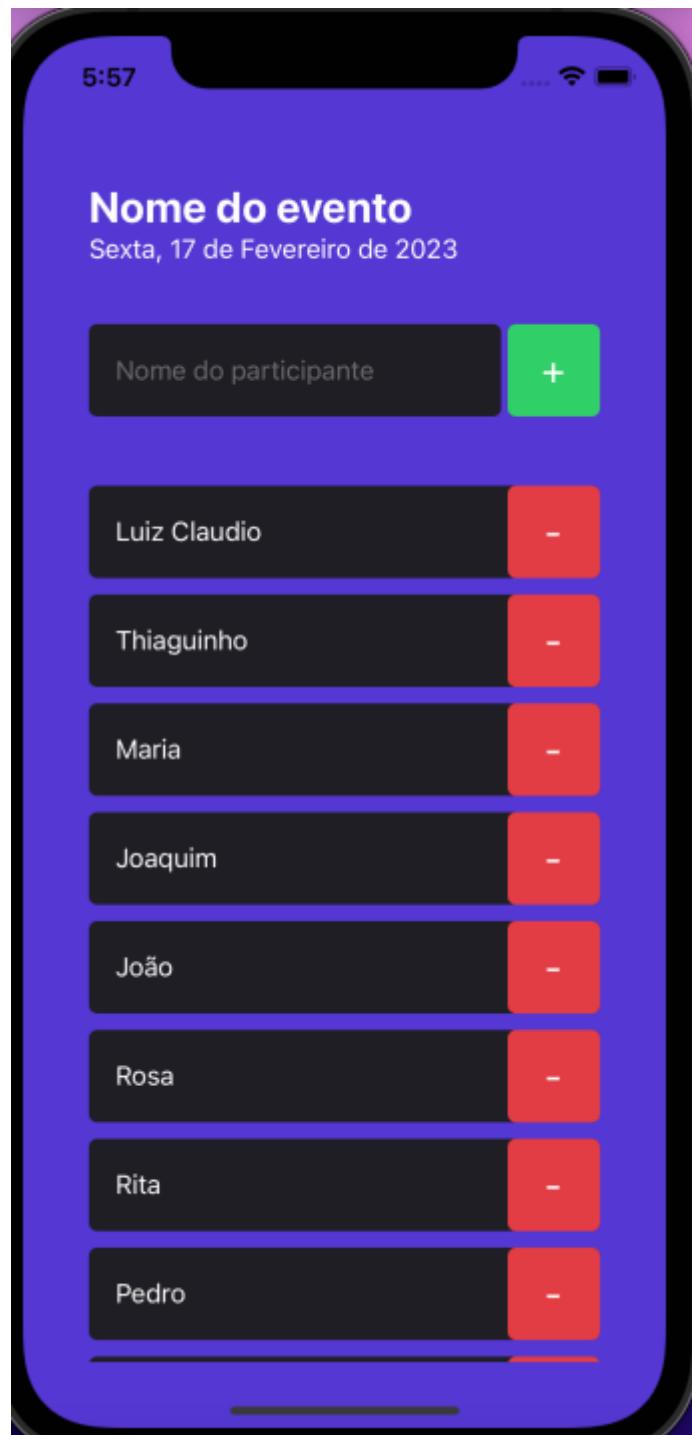


Se vc voltar com o array `participantes` a listagem volta a funcionar.

Altere a linha 55 conforme abaixo.

```
index.tsx Home U X
src > screens > Home > index.tsx > ...
54     <FlatList
55       data={participants}
56       keyExtractor={item => item.id}
57       renderItem={({ item }) => (
58         <Participant
59           name={item.name}
60           onRemove={() => handleRemoveParticipant('Thiaguino')}
61         />
62       )}
63       showsVerticalScrollIndicator={false}
64       ListEmptyComponent={() => (
65         <Text style={styles.listEmptyText}>
66           Adicione participante a sua lista de presença
67         </Text>
68       )}
69     />
70   </View>
71 }
72 }
```

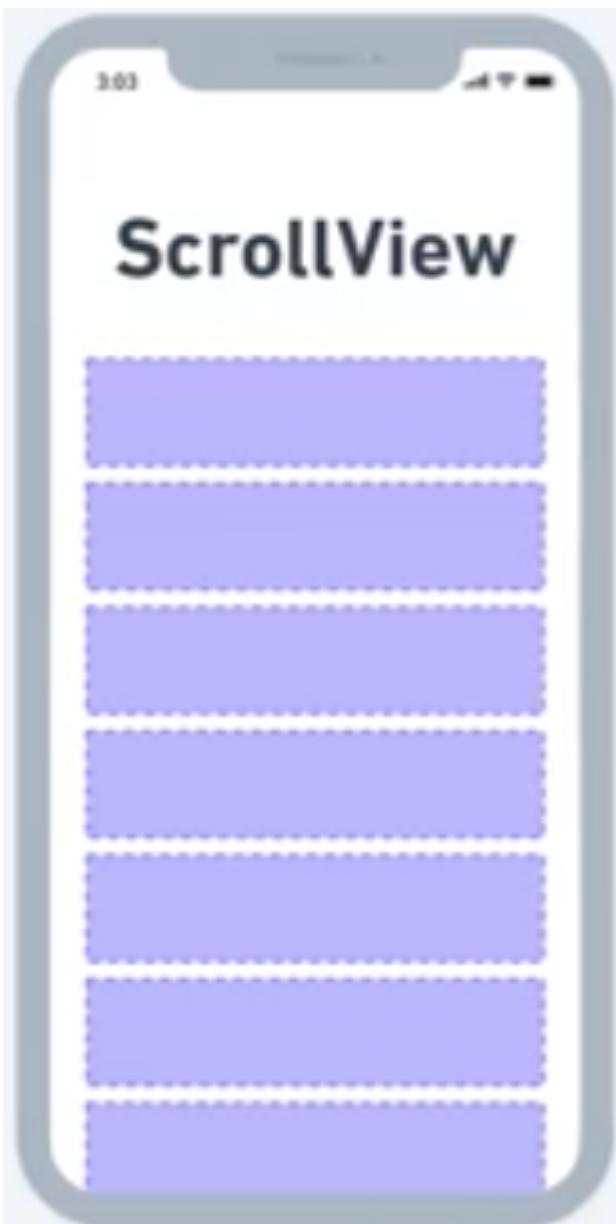
Salve as alterações e veja o resultado.



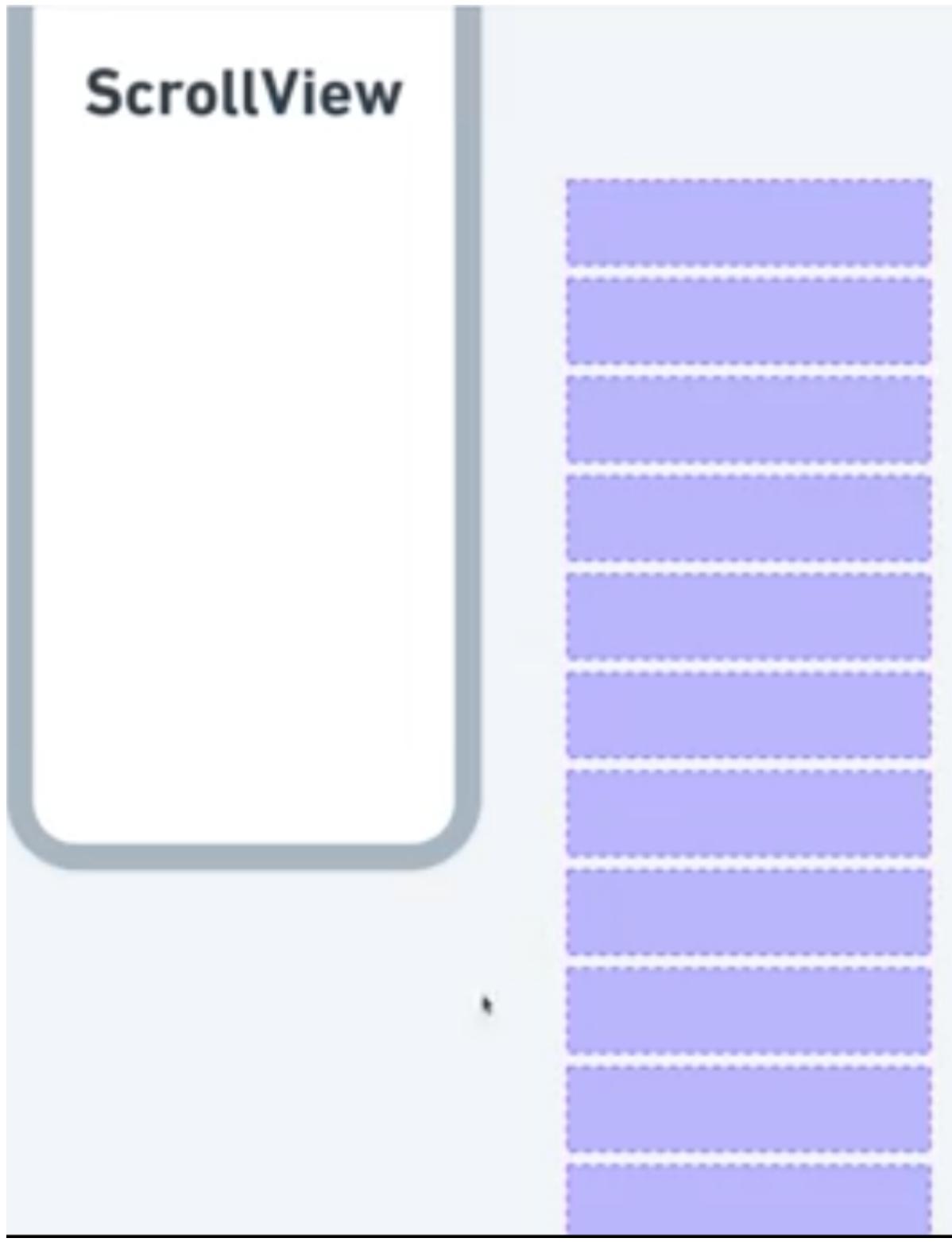
ScrollView x FlatList

ScrollView

Renderiza todos os seus componentes filhos de reação de uma só vez, mas isso tem uma desvantagem de desempenho. Ela é ideal para listas pequenas.



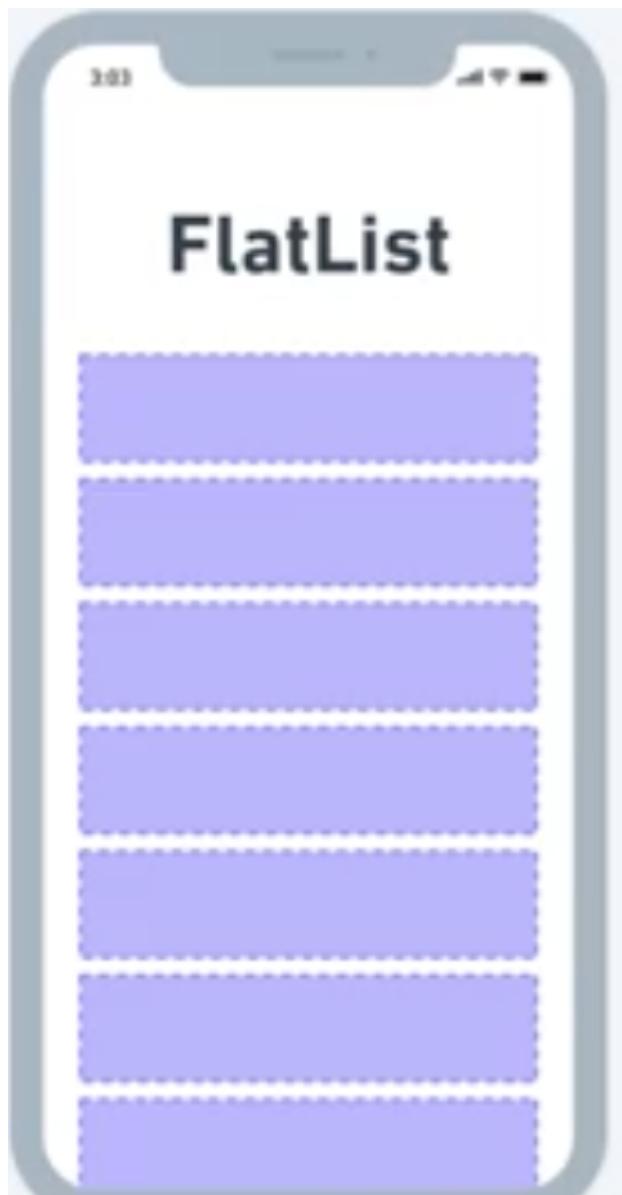
ScrollView



FlatList

Renderiza itens lentamente, quando ele estão prestes a aparecer, e remover itens que rolam para fora da tela para economizar memória e tempo de processamento. É

ideal para listas grandes.





Estilizando a StatusBar

Edite o arquivo src/App.tsx

Insira a linha 1 para importar a StatusBar

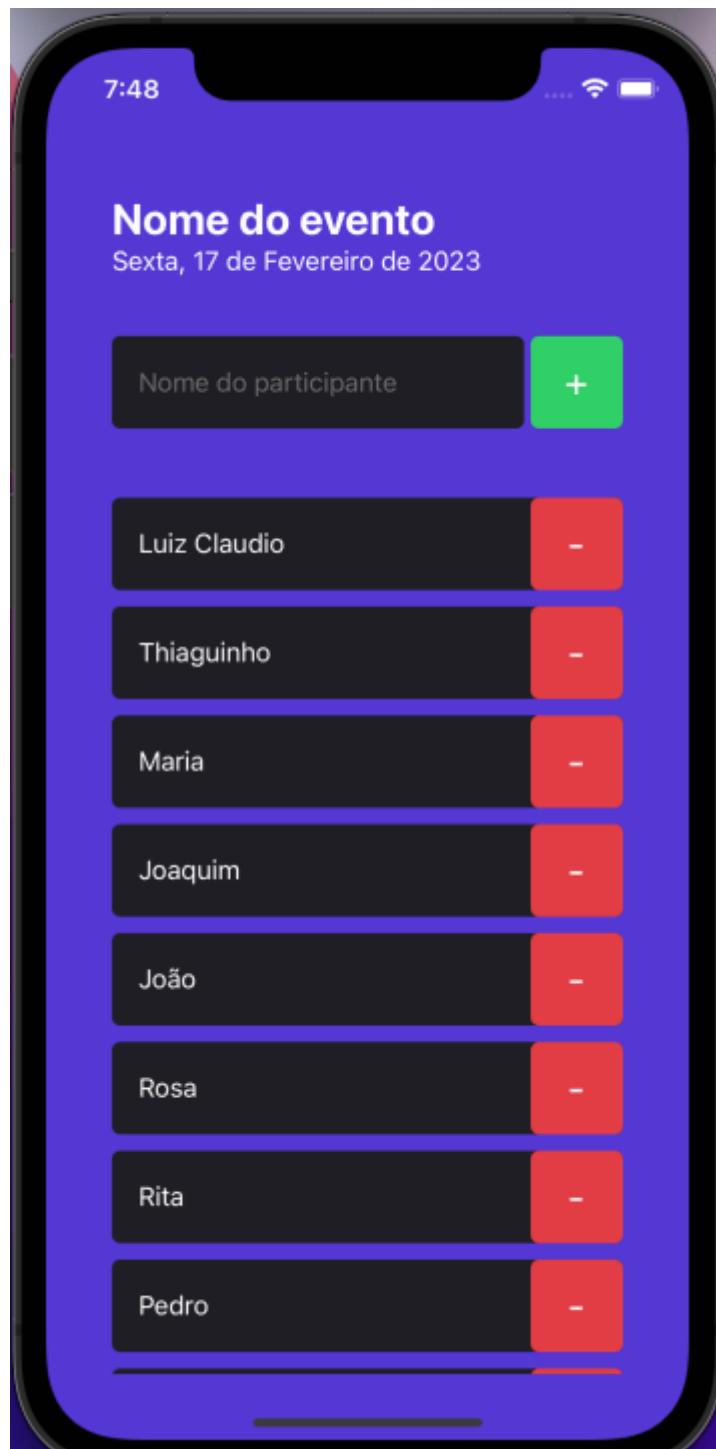
Insira as linhas 6, 7 e 9

```
⚙️ App.tsx myapp5p M X
● myapp5p > ⚙️ App.tsx > ...
1 import { StatusBar } from "react-native";
2 import Home from "./src/screens/Home";
3
4 export default function App() {
5   return (
6     <>
7       <StatusBar />
8       <Home />
9     </>
10    )
11  }
12 }
```

Altere a linha 7 conforme abaixo para definir um conteúdo light

```
⚙️ App.tsx myapp5p M X
● myapp5p > ⚙️ App.tsx > ...
1 import { StatusBar } from "react-native";
2 import Home from "./src/screens/Home";
3
4 export default function App() {
5   return (
6     <>
7       <StatusBar barStyle="light-content" />
8       <Home />
9     </>
10    )
11  }
```

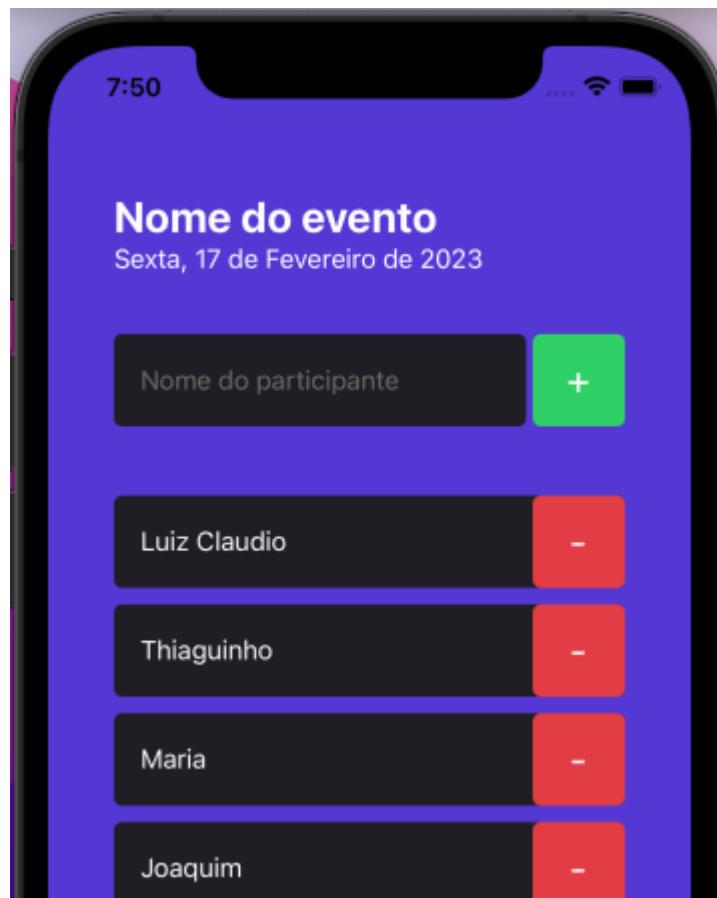
Salve as alterações e veja o resultado



Altere a linha 7 conforme abaixo para definir um conteúdo dark.

```
App.tsx myapp5p M X
● myapp5p > App.tsx > ...
1 import { StatusBar } from "react-native";
2 import Home from "./src/screens/Home";
3
4 export default function App() {
5   return (
6     <>
7       <StatusBar barStyle="dark-content" />
8       <Home />
9     </>
10   )
11 }
```

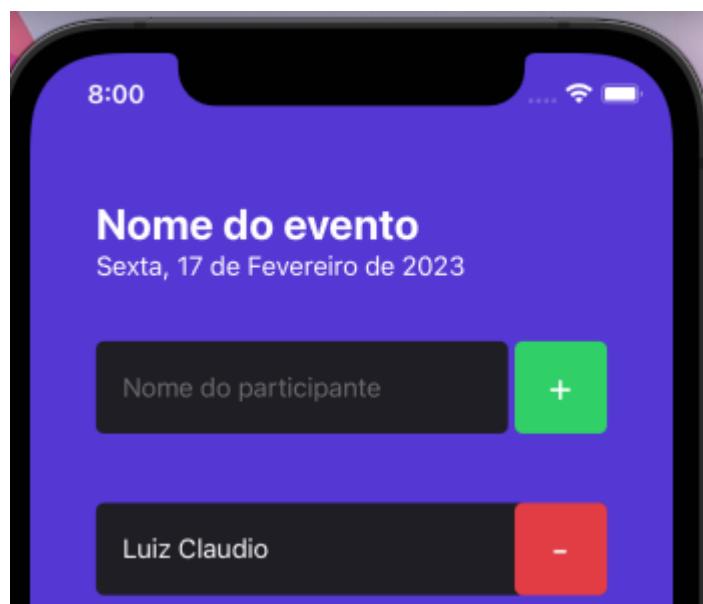
Salve as alterações e veja o resultado na barra de status



Insira a linha 9 para definir a cor do fundo como transparent.

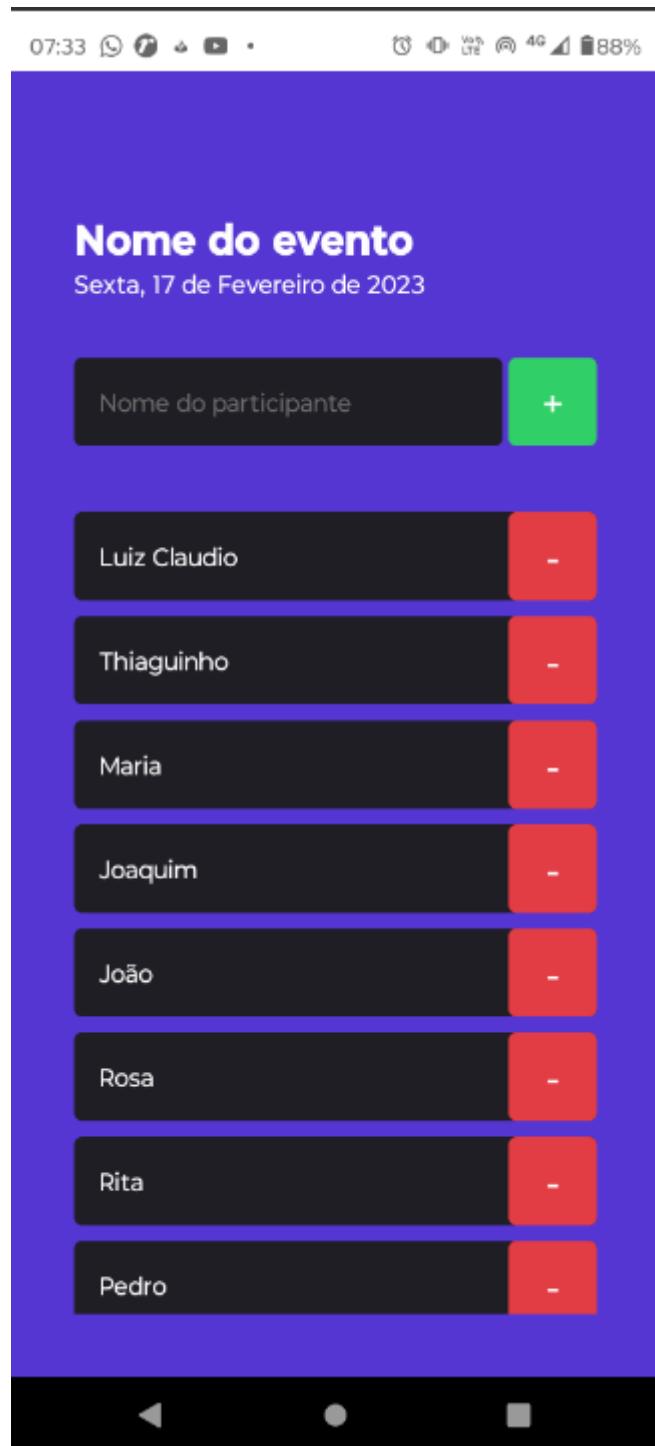
```
App.tsx myapp5p M X
myapp5p > App.tsx > App
1 import { StatusBar } from "react-native";
2 import Home from "./src/screens/Home";
3
4 export default function App() {
5   return (
6     <>
7       <StatusBar
8         barStyle="light-content"
9         backgroundColor="transparent"
10      />
11      <Home />
12    </>
13  )
14}
```

Salve as alterações e veja o resultado, observe que no iphone não teve alteração.



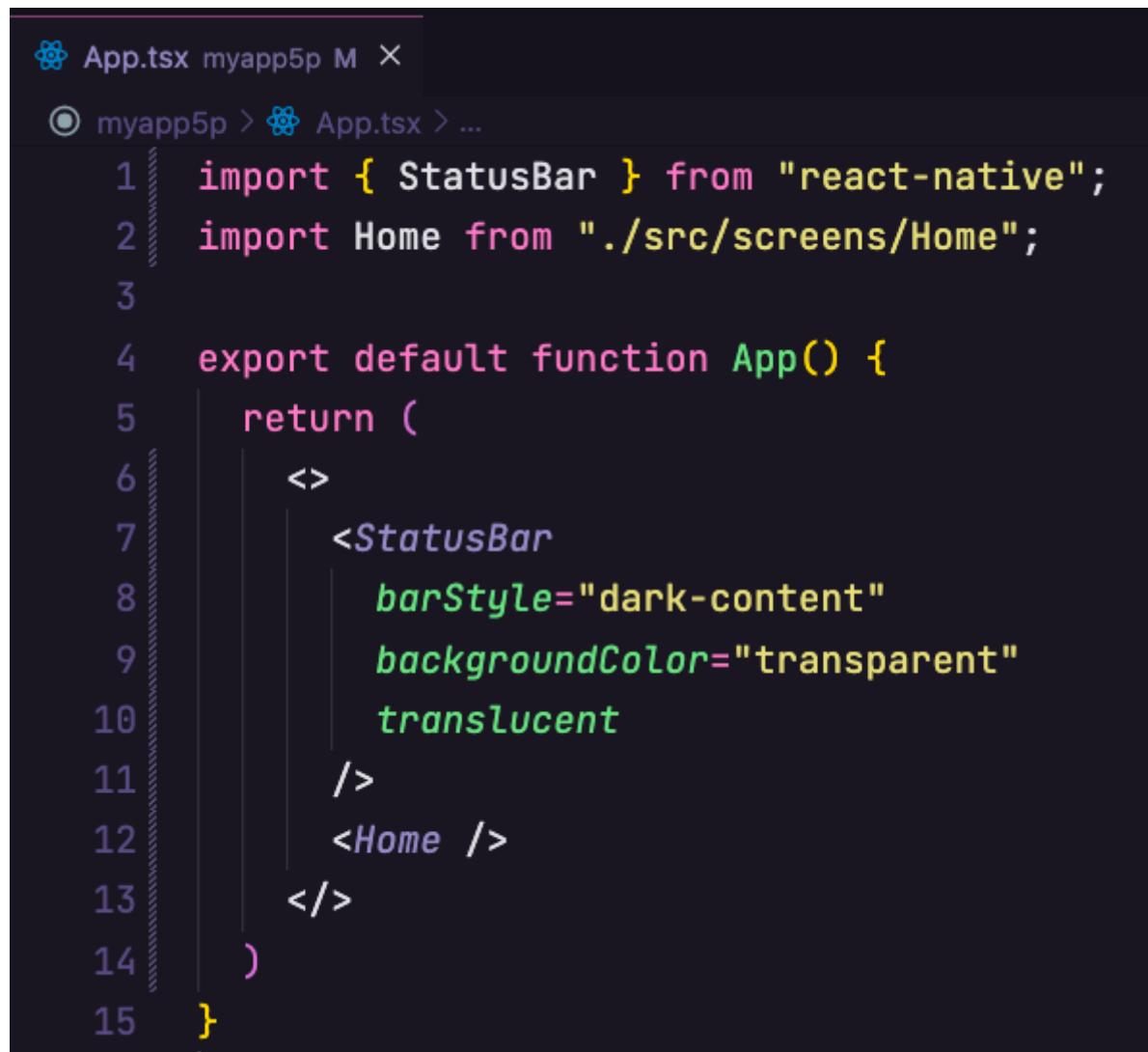
O print abaixo mostra a cor de fundo da barra de status no android, no android fica um fundo branco.

print está no email do android.



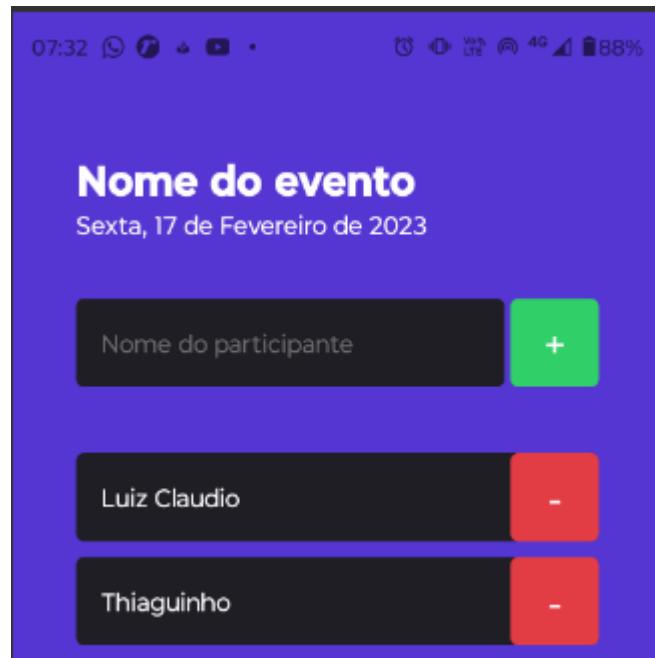
A aplicação começa a partir da barra de status, mas vc pode sobrepor esta funcionalidade, pedir para a Status Bar sobrepor a aplicação inserindo a propriedade translucent.

Insira a linha 10 conforme print abaixo.



```
App.tsx myapp5p M X
myapp5p > App.tsx > ...
1 import { StatusBar } from "react-native";
2 import Home from "./src/screens/Home";
3
4 export default function App() {
5   return (
6     <>
7       <StatusBar
8         barStyle="dark-content"
9         backgroundColor="transparent"
10        translucent
11      />
12      <Home />
13    </>
14  )
15}
```

Salve as alterações e veja o resultado.



Adicionando alertas no Aplicativo

Por exemplo, o app vai aceitar nomes participantes com nomes únicos, não irá aceitar nomes repetidos.

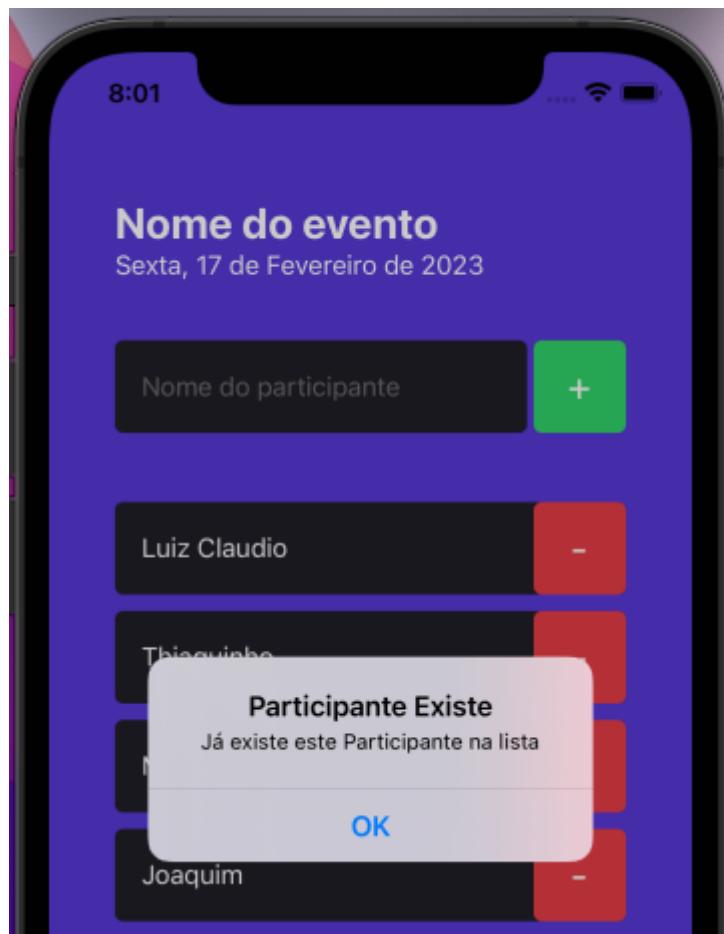
Altere a linha 1 para inserir o Alert

```
index.tsx Home U X
① myapp5p > src > screens > Home > index.tsx > Home > participants
1 import { Text, View, TextInput, TouchableOpacity, FlatList, Alert } from "react-native";
2 import { Participant } from "../../components/Participant";
3
4 import { styles } from './styles'
```

Insira da linha 18 até 22 para fazer a validação se o participante já existe

```
index.tsx Home U X
myapp5p > src > screens > Home > index.tsx > Home > handleAddParticipant
  7 | const participants = [
  8 |   { id: '1', name: 'Luiz Claudio' },
  9 |   { id: '2', name: 'Thiaguinho' },
 10 |   { id: '3', name: 'Maria' },
 11 |   { id: '4', name: 'Joaquim' },
 12 |   { id: '5', name: 'João' },
 13 |   { id: '6', name: 'Rosa' },
 14 |   { id: '7', name: 'Rita' },
 15 |
 16 |
 17 | function handleAddParticipant() {
 18 |   const result = participants.filter(participant => participant.name === 'Luiz Claudio')
 19 |
 20 |   if (result.length > 0) {
 21 |     return Alert.alert('Participante Existe', 'Já existe este Participante na lista')
 22 |   }
 23 |   console.log('Você clicou no botão Adicionar')
 24 | }
```

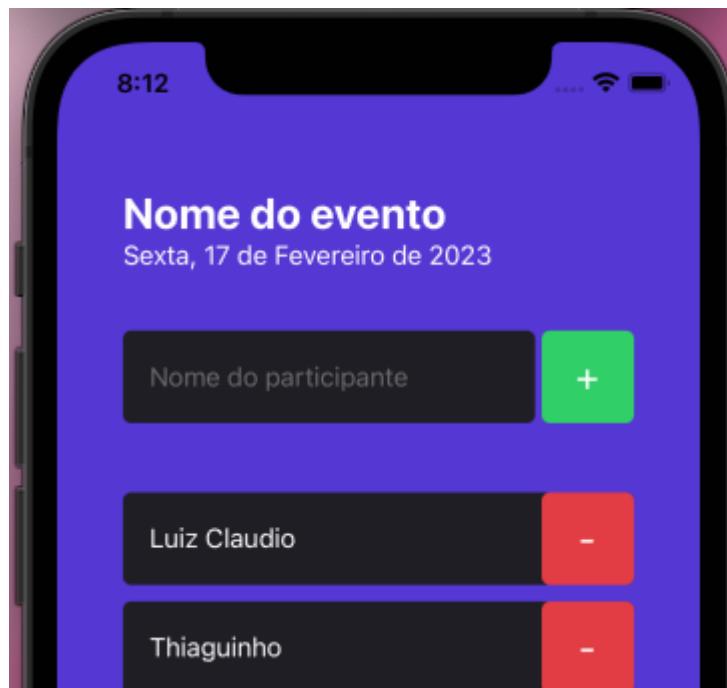
Salve as alterações, abra o app, clique no botão Adicionar (+) e veja o resultado.



Altere a linha 18 para um nome que não existe no array de participantes

```
index.tsx Home U X
myapp5p > src > screens > Home > index.tsx > Home
 7 | const participants = [
 8 |   { id: '1', name: 'Luiz Claudio' },
 9 |   { id: '2', name: 'Thiaguinho' },
10 |   { id: '3', name: 'Maria' },
11 |   { id: '4', name: 'Joaquim' },
12 |   { id: '5', name: 'João' },
13 |   { id: '6', name: 'Rosa' },
14 |   { id: '7', name: 'Rita' },
15 |
16 |
17 function handleAddParticipant() {
18   const result = participants.filter(participant => participant.name === 'Beatriz')
19
20   if (result.length > 0) {
21     return Alert.alert('Participante Existe', 'Já existe este Participante na lista')
22   }
23   console.log('Você clicou no botão Adicionar')
24 }
```

Salve as alterações, clique no botão Adicionar (+) e veja que a mensagem não será exibida.



Fazendo uma pergunta

Insira da linha 27 até 36 conforme abaixo

```
index.tsx Home U X
myapp5p > src > screens > Home > index.tsx > Home
25
26     function handleRemoveParticipant(name: string) {
27         Alert.alert('Remover', `Remover o participante ${name}`, [
28             {
29                 text: 'Sim',
30                 onPress: () => Alert.alert('Deletado')
31             },
32             {
33                 text: 'Não',
34                 style: 'cancel'
35             }
36         ])
37         console.log(`Você clicou em remover o participante ${name}`)
38     }
39
40     return (

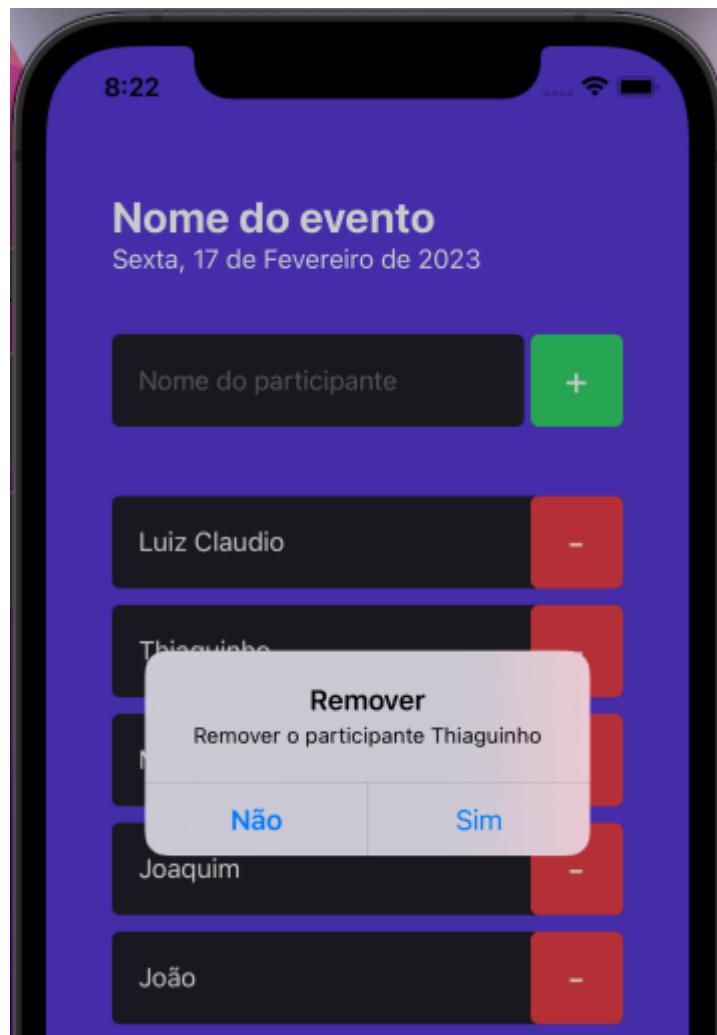
```

Altere a linha 71 para enviar a função o nome do participante da sua lista que será deletado

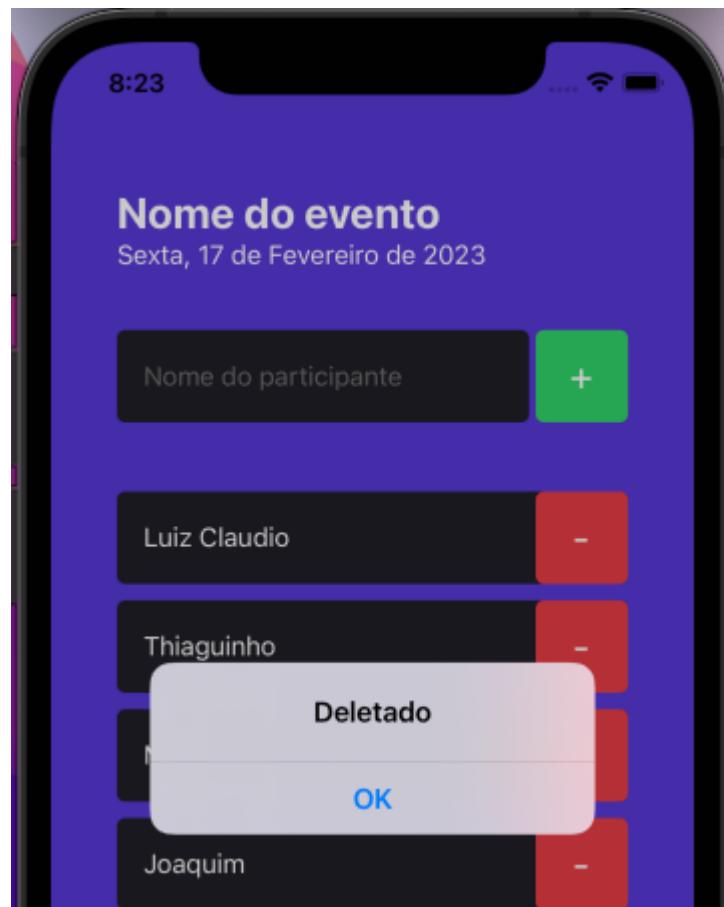
```
index.tsx Home U X
myapp5p > src > screens > Home > index.tsx > Home
04
05     <FlatList
06         data={participants}
07         keyExtractor={item => item.id}
08         renderItem={({ item }) =>
09             <Participant
10                 name={item.name}
11                 onRemove={() => handleRemoveParticipant(item.name)}
12             />
13         }
14         showsVerticalScrollIndicator={false}

```

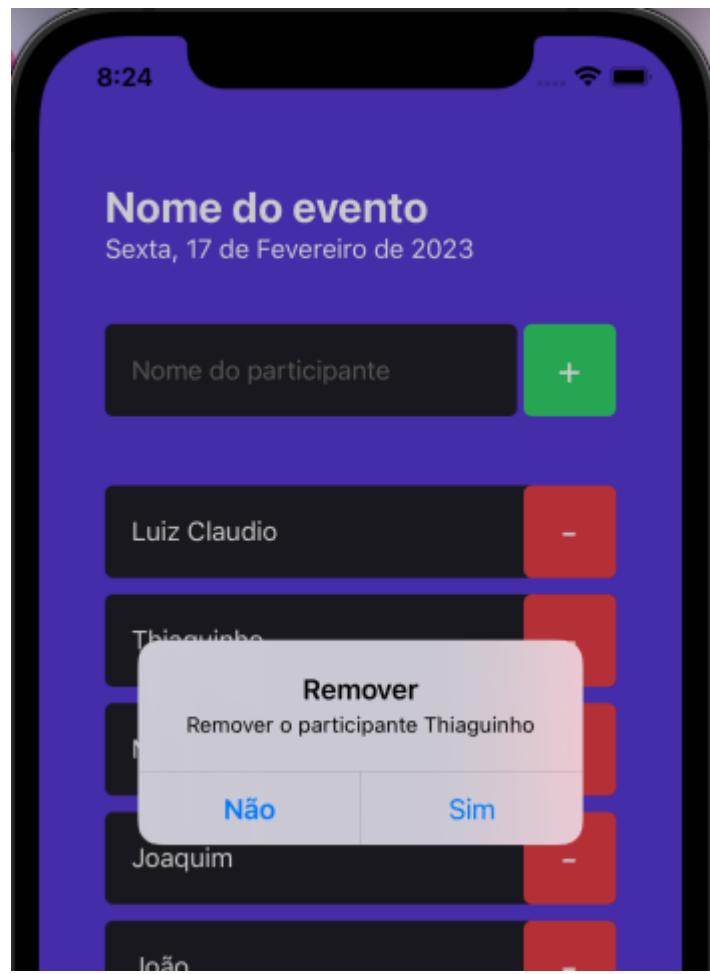
Salve as alterações, clique no botão Excluir (-), por exemplo, ao lado do nome Thiaguinho e veja o resultado.



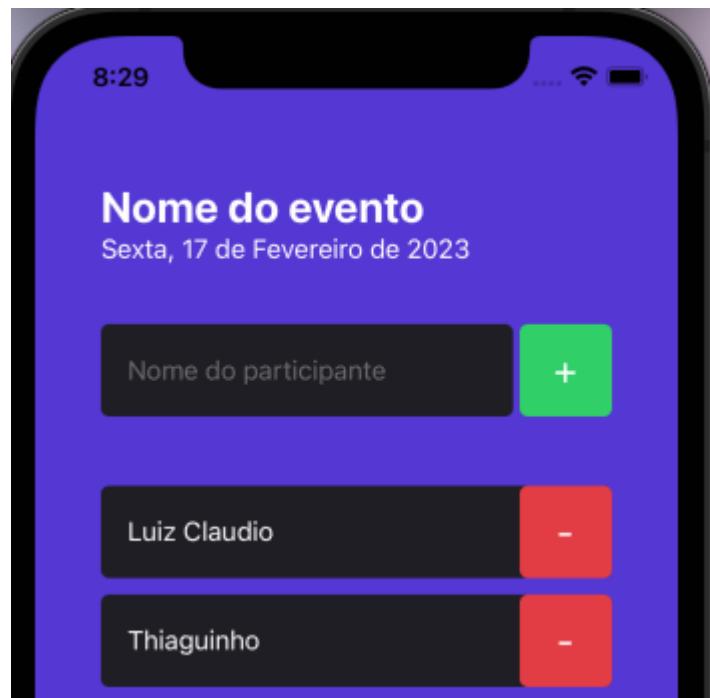
Clique na opção Sim e será exibida a mensagem abaixo



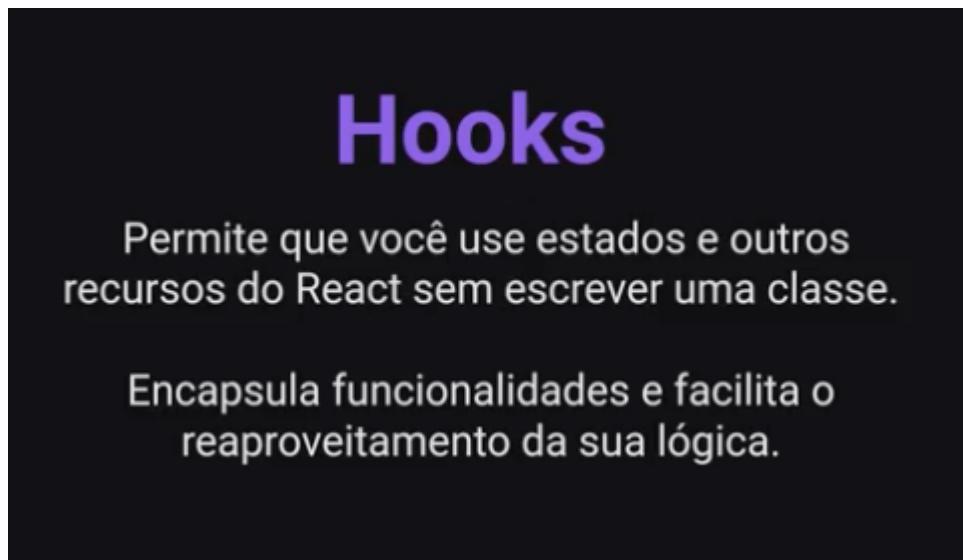
Faça outro teste para excluir e clique no botão Não, observe que nenhuma mensagem será exibida



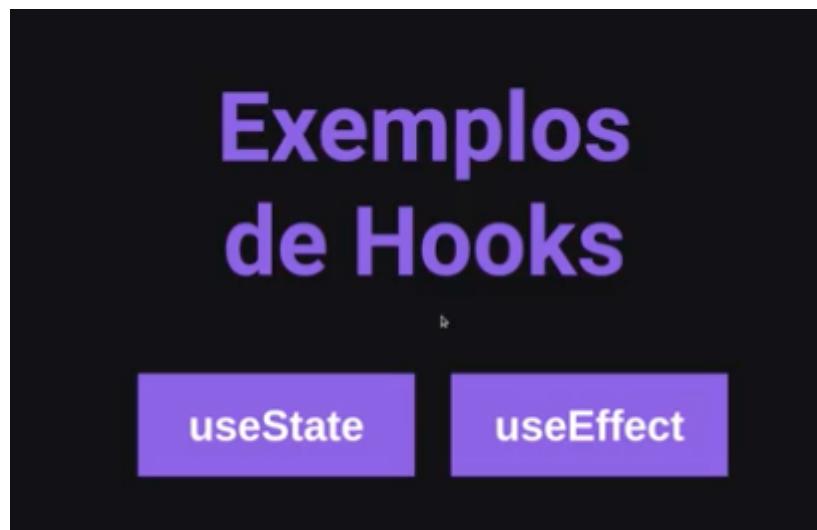
Tela depois que clicar no botão Não



Conceitos de Hooks



Exemplos de Hooks



Nome do Hook

NomeDoHook

camelCase

antes do nome do Hook vc acrescenta o use

useNomeDoHook

camelCase

Outros exemplos de padrões de nome



snake_case

Pros: Concise when it consists of a few words.
Cons: Redundant as hell when it gets longer.
`push_something_to_first_queue`, `pop_what`, `get_whatever...`



PascalCase

Pros: Seems neat.
`GetItem`, `SetItem`, `Convert`, ...
Cons: Barely used. (why?)



camelCase

Pros: Widely used in the programmer community.
Cons: Looks ugly when a few methods are ~~n-worded~~.
`push`, `reserve`, `beginBuilding`, ...