



++

45697096

Componentes do React Native I

Mobile App Development

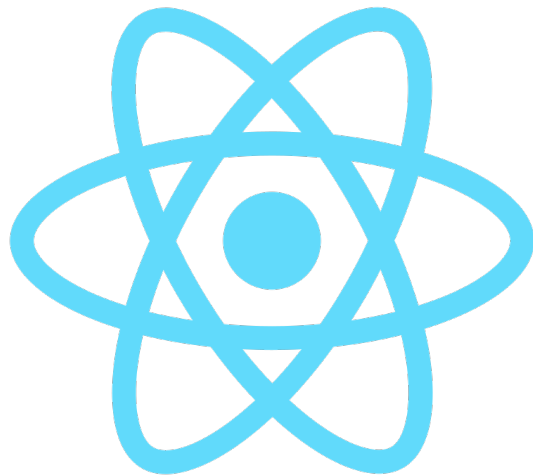
45697096

45697096

...



Prof. Vinny Albuquerque
profvinny.albuquerque@fiap.com.br



React Native - Componentes

- React Native é composto de React Components;
- Componentes são um paradigma poderoso para organizar visualizações e gerenciar dados dinâmicos.
- Quando usamos React Native, representamos diferentes partes da nossa aplicação como **componentes**.
 - Isso significa que podemos construir nosso aplicativo usando diferentes peças reutilizáveis de lógica, com cada peça exibindo uma parte específica de nossa IU.
- Podemos criar componentes no React Native utilizando classes JavaScript.
 - É necessário implementar o método **render ()**

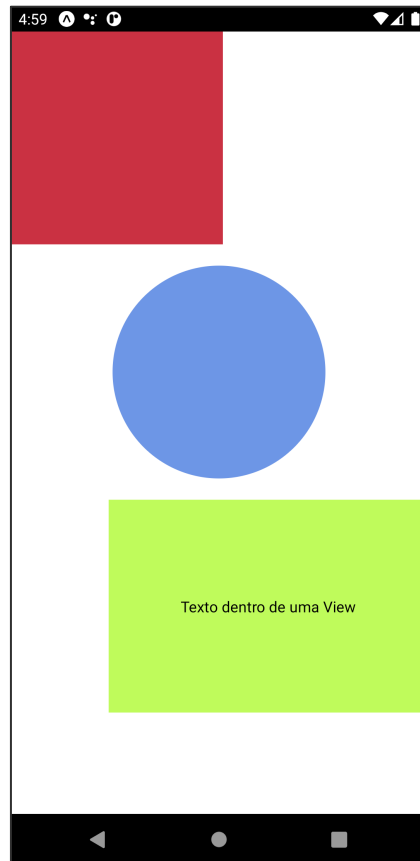
JS App.js > ...

```
1  import React from 'react';
2  import {View, Text} from 'react-native';
3
4  export default class App extends React.Component {
5    render() {
6      return (
7        <View>
8          <Text style={{color : 'red'}}>Olá Mundo!</Text>
9        </View>
10      );
11    }
12  }
```

- A **View** é um dos componentes mais básicos do React-Native, lembrando muito o uso de DIVs no HTML;
- A View pode ser usada como um **container** para outros componentes, facilitando assim a organização de forma horizontal ou vertical;
- Outro uso é a estilização do App, podendo usar a View para criar círculos, quadrados, retângulos, linhas, etc.
- Sua principal propriedade é o **style**.

Componente - View - Exemplo

```
js App.js > ...
1  import React from 'react';
2  import {StatusBar, Text, View} from 'react-native';
3
4  export default class App extends React.Component {
5    render() {
6      return (
7        <View>
8          <View style={{backgroundColor : 'crimson', height : 200, width : 200}}></View>
9          <View style={{alignSelf : 'center', backgroundColor : 'cornflowerblue',
10            borderRadius : 100, height : 200, marginVertical : 20, width : 200}}></View>
11          <View style={{alignItems : 'center', backgroundColor : 'greenyellow',
12            alignSelf : 'flex-end', height : 200, justifyContent : 'center', width : 300}}>
13            <Text>Texto dentro de uma View</Text>
14          </View>
15          <StatusBar style="light"/>
16        </View>
17      );
18    }
19  }
```



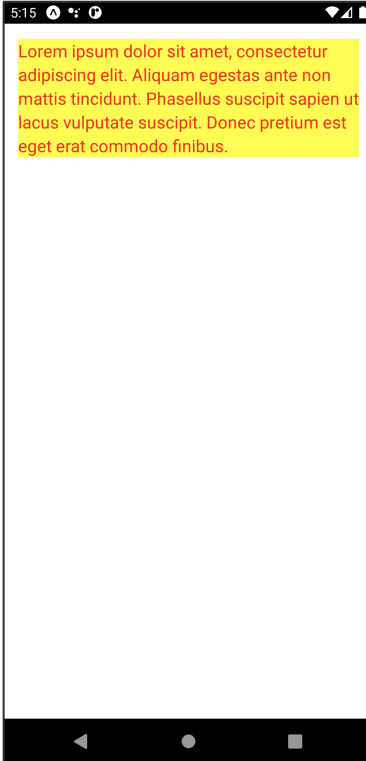
- React Native nos permite descrever a representação de um componente iOS e Android em JavaScript.
- JSX é uma extensão de JavaScript que nos permite usar uma sintaxe semelhante a XML para definir nossa estrutura de UI.
- No JSX, as chaves são um delimitador, sinalizando para JSX que o que reside entre as chaves é uma expressão JavaScript.

- Props nos permite passar parâmetros aos componentes para personalizar suas características.
- Cada componente integrado fornecido pelo React Native tem seu próprio conjunto de props válidos que podemos usar para personalização.

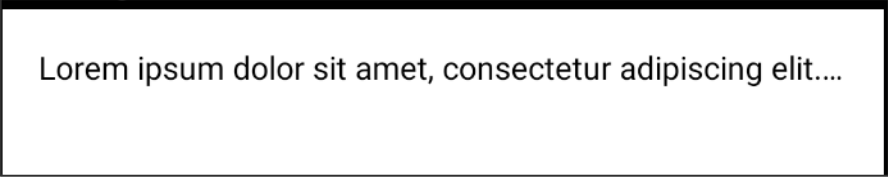
- O componente **Text** é utilizado sempre que necessário a renderização de textos na tela e permite a estilização das propriedade de um texto:
 - Tamanho;
 - Cor;
 - Background;
 - Entre outros.

Composante - Text - Exemple

```
JS App.js > ...
1 import React from 'react';
2 import {StatusBar, Text, View} from 'react-native';
3
4 export default class App extends React.Component {
5   render() {
6     return (
7       <View style={{ padding: 16 }}>
8         <Text style={{ backgroundColor: 'yellow', color: '#F00', fontSize: 18, lineHeight: 25 }}>
9           Lorem ipsum dolor sit amet, consectetur adipiscing elit.
10          Aliquam egestas ante non mattis tincidunt.
11          Phasellus suscipit sapien ut lacus vulputate suscipit.
12          Donec pretium est eget erat commodo finibus.
13        </Text>
14        <StatusBar style="light"/>
15      </View>
16    );
17  }
18 }
```



- É possível especificar o número de linhas que o componente Text pode exibir.



Lorem ipsum dolor sit amet, consectetur adipiscing elit....

```
<Text numberOfLines={1}>  
  Lorem ipsum dolor sit amet,  
  consectetur adipiscing elit.  
  Sed id malesuada neque.  
  Phasellus eget porta mi.  
</Text>
```

- Por padrão, o texto é truncado com "... " no final da linha, equivalente a propriedade: `ellipsizeMode="tail"`.

Componente - Text - Propriedades - ellipsizeMode

- Quando informado o número de linhas, é possível configurar o modo que a linha será truncada usando a propriedade `ellipsizeMode` com um dos seguintes valores: `head`, `middle`, `tail` ou `clip`.

...elit. Sed id malesuada neque. Phasellus eget porta mi.

```
<Text ellipsizeMode='head'  
  numberOfLines={1}>  
  Lorem ipsum dolor sit amet,  
  consectetur adipiscing elit.  
  Sed id malesuada neque.  
  Phasellus eget porta mi.  
</Text>
```

Lorem ipsum dolor sit amet...ue. Phasellus eget porta mi.

```
<Text ellipsizeMode='middle'  
  numberOfLines={1}>  
  Lorem ipsum dolor sit amet,  
  consectetur adipiscing elit.  
  Sed id malesuada neque.  
  Phasellus eget porta mi.  
</Text>
```

Componente - Text - Propriedades - ellipsizeMode

Lorem ipsum dolor sit amet, consectetur adipiscing elit. S

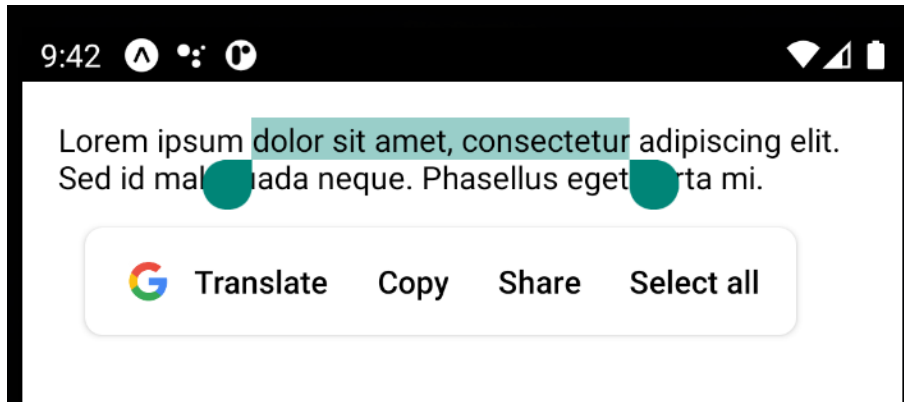
```
<Text ellipsizeMode='clip'  
  numberOfLines={1}>  
  Lorem ipsum dolor sit amet,  
  consectetur adipiscing elit.  
  Sed id malesuada neque.  
  Phasellus eget porta mi.  
</Text>
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit....

```
<Text ellipsizeMode='tail'  
  numberOfLines={1}>  
  Lorem ipsum dolor sit amet,  
  consectetur adipiscing elit.  
  Sed id malesuada neque.  
  Phasellus eget porta mi.  
</Text>
```

Componente - Text - Propriedades - selectable

- Para permitir que o texto possa ser selecionado e copiado, basta atribuir a propriedade **selectable** com o valor **true**. O padrão é **false**.



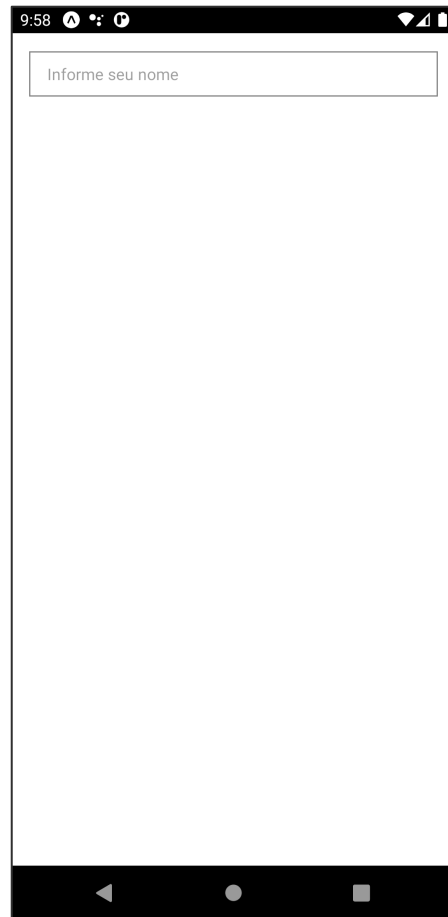
```
<Text selectable={true}>  
  Lorem ipsum dolor sit amet,  
  consectetur adipiscing elit.  
  Sed id malesuada neque.  
  Phasellus eget porta mi.  
</Text>
```

- **TextInput** é o principal componente para entrada de dados em formato texto pelo usuário.
- Por default, possui a estilização simples, sendo necessária a modificação através da propriedade `style`.

Componente - TextInput - Exemplo

JS App.js > ...

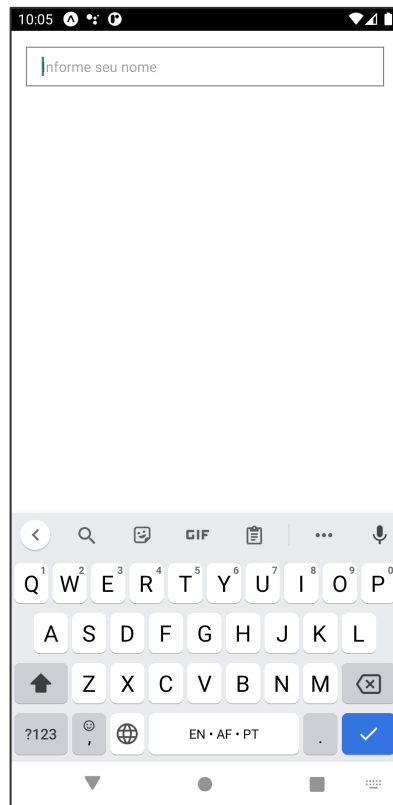
```
1  import React from 'react';
2  import { StatusBar, TextInput, View } from 'react-native';
3
4  export default class App extends React.Component {
5    render() {
6      return (
7        <View style={{ padding: 16 }}>
8          <TextInput
9            placeholder="Informe seu nome"
10             style={{
11               borderColor: 'gray',
12               borderWidth: 1,
13               height: 40,
14               paddingHorizontal: 16
15             }} />
16          <StatusBar style="light" />
17        </View>
18      );
19    }
20  }
```



Componente - TextInput - Propriedades - autoFocus

- Permite que o TextInput já inicie com o cursor dentro, pronto para digitação do texto.
- Os valores dessa propriedade pode ser **true** ou **false** (default).

```
<TextInput
  autoFocus={true}
  placeholder="Informe seu nome"
  style={{
    borderColor: 'gray',
    borderWidth: 1,
    height: 40,
    paddingHorizontal: 16
  }} />
```



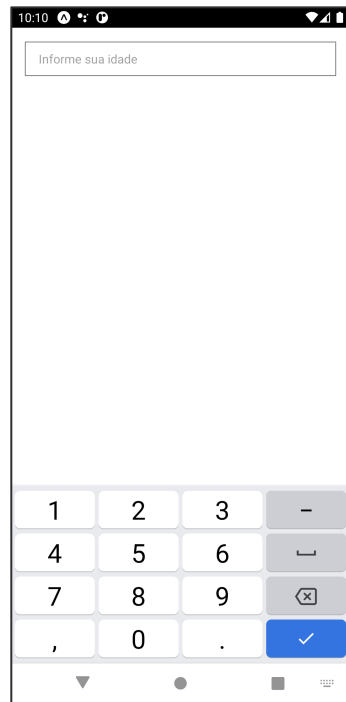
- Quando o valor **false** é informado para esta propriedade, o campo deixa de ser editável.

```
<TextInput
  editable={false}
  placeholder="Informe seu nome"
  style={{
    borderColor: 'gray',
    borderWidth: 1,
    height: 40,
    paddingHorizontal: 16
  }} />
```

Componente - TextInput - Propriedades - keyboardType

- Permite informar qual o tipo de teclado será utilizado para os dados do TextInput.

```
<TextInput
  keyboardType='numeric'
  placeholder="Informe sua idade"
  style={{
    borderColor: 'gray',
    borderWidth: 1,
    height: 40,
    paddingHorizontal: 16
  }} />
```

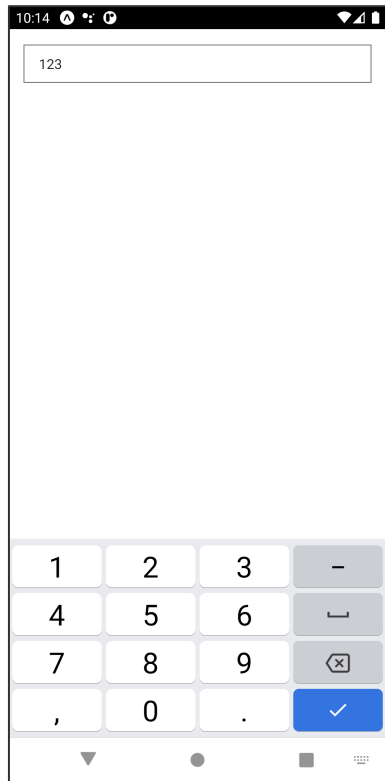


Veja todos as possibilidades em: <https://lefkowitz.me/visual-guide-to-react-native-textinput-keyboardtype-options/>

Componente - TextInput - Propriedades - maxLength

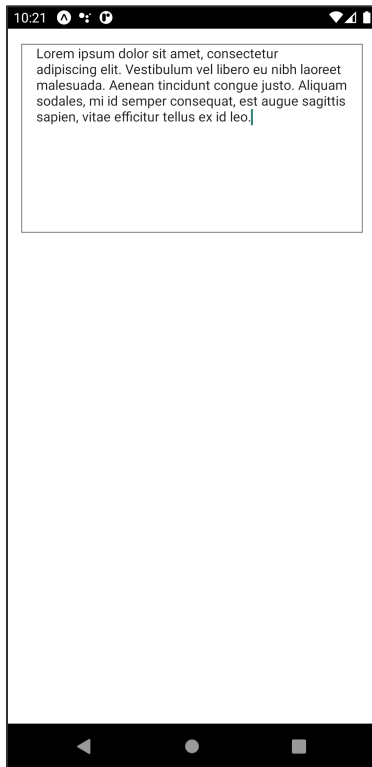
- Tamanho máximo de caracteres permitidos no TextInput.

```
<TextInput
  keyboardType='numeric'
  maxLength={3}
  placeholder="Informe sua idade"
  style={{
    borderColor: 'gray',
    borderWidth: 1,
    height: 40,
    paddingHorizontal: 16
  }} />
```



- É possível permitir texto com múltiplas linhas usando a propriedade `multiline={true}` e o alinhamento deste campo com `textAlignVertical` com um dos seguintes valores: `auto`, `bottom`, `center` e `top`.

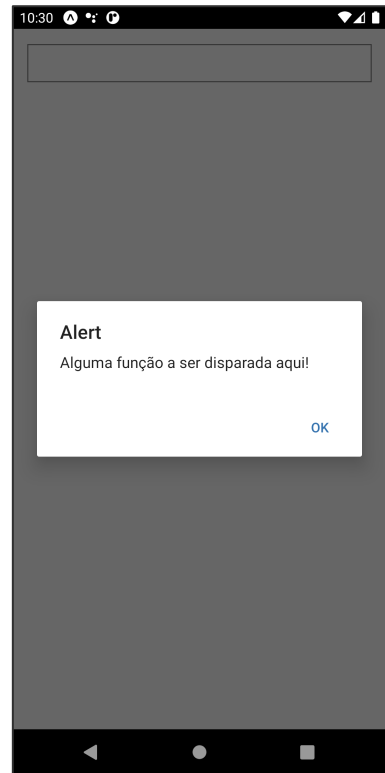
```
<TextInput
  multiline={true}
  style={{
    borderColor: 'gray',
    borderWidth: 1,
    height: 200,
    paddingHorizontal: 16
  }}
  textAlignVertical='top' />
```



Componente - TextInput - Propriedades - onBlur

- Evento chamado quando o foco do cursor deixa o TextInput. Deve ser passada uma **function** para esta propriedade.

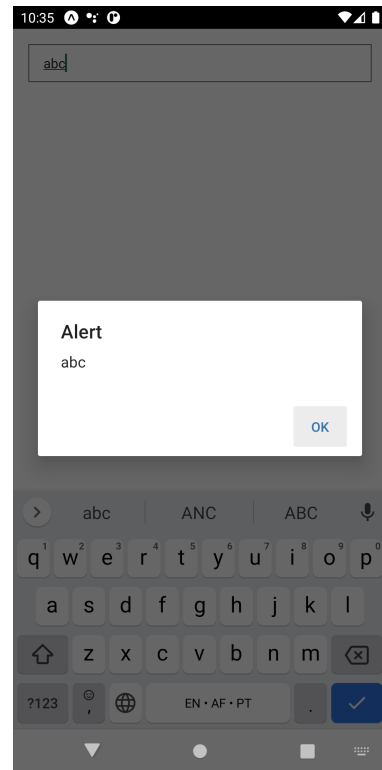
```
<TextInput
  onBlur={ _ => alert('Alguma função a ser disparada aqui!')}
  style={{
    borderColor: 'gray',
    borderWidth: 1,
    height: 40,
    paddingHorizontal: 16
  }} />
```



Componente - TextInput - Propriedades - onChangeText

- Evento disparado sempre que o conteúdo do TextInput é **alterado**. Deve ser informado uma **function** para esta propriedade. O primeiro parâmetro da fuction representa o valor dentro do TextInput.

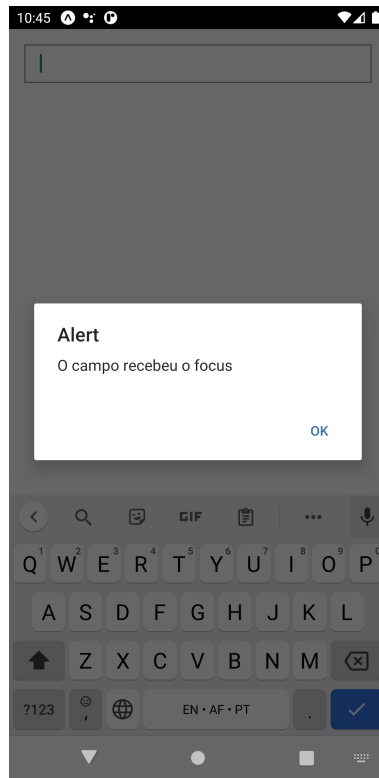
```
<TextInput
  onChangeText={value => alert(value)}
  style={{
    borderColor: 'gray',
    borderWidth: 1,
    height: 40,
    paddingHorizontal: 16
  }} />
```



Componente - TextInput - Propriedades - onFocus

- Evento disparado sempre que o TextInput receber o **focus**. Deve ser informado uma **function** que será executada quando o evento ocorrer.

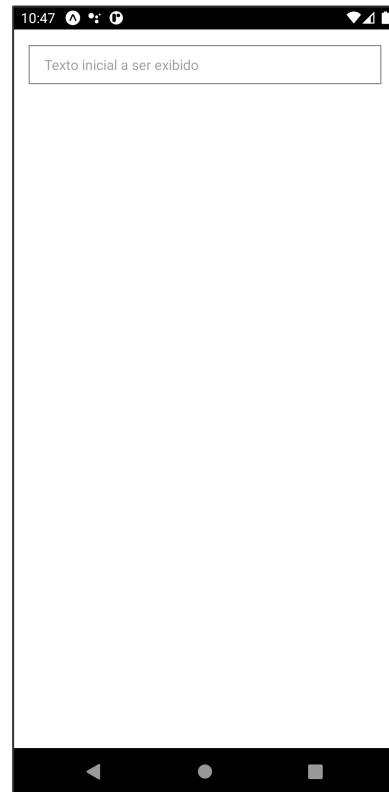
```
<TextInput
  onFocus={_ => alert('O campo recebeu o focus')}
  style={{
    borderColor: 'gray',
    borderWidth: 1,
    height: 40,
    paddingHorizontal: 16
  }} />
```



Componente - TextInput - Propriedades - placeholder

- Esta propriedade permite informar a String que será renderizada antes da entrada de algum dado no TextInput.

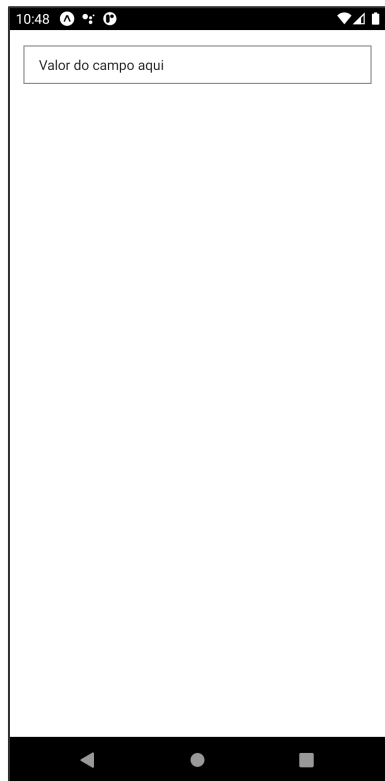
```
<TextInput
  placeholder="Texto inicial a ser exibido"
  style={{
    borderColor: 'gray',
    borderWidth: 1,
    height: 40,
    paddingHorizontal: 16
  }} />
```



Componente - TextInput - Propriedades - value

- O valor do TextInput a ser exibido.

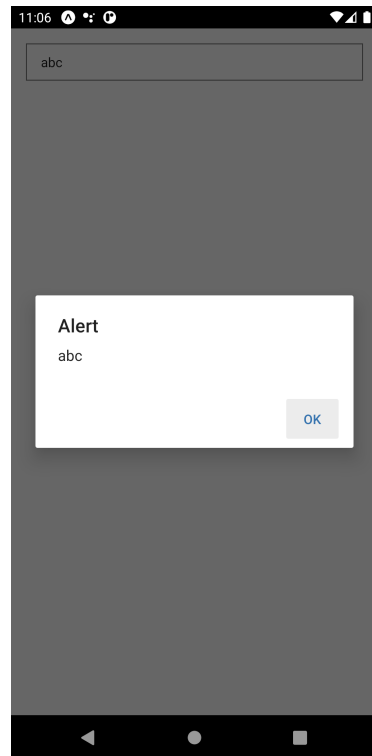
```
<TextInput
  value="Valor do campo aqui"
  style={{
    borderColor: 'gray',
    borderWidth: 1,
    height: 40,
    paddingHorizontal: 16
  }} />
```



Componente - TextInput - Propriedades - onSubmitEditing

- Evento disparado quando o botão **submit** do teclado virtual é pressionado. Deve ser informado uma **function** para esta propriedade.

```
<TextInput
  onSubmitEditing={event => alert(event.nativeEvent.text)}
  style={{
    borderColor: 'gray',
    borderWidth: 1,
    height: 40,
    paddingHorizontal: 16
  }} />
```



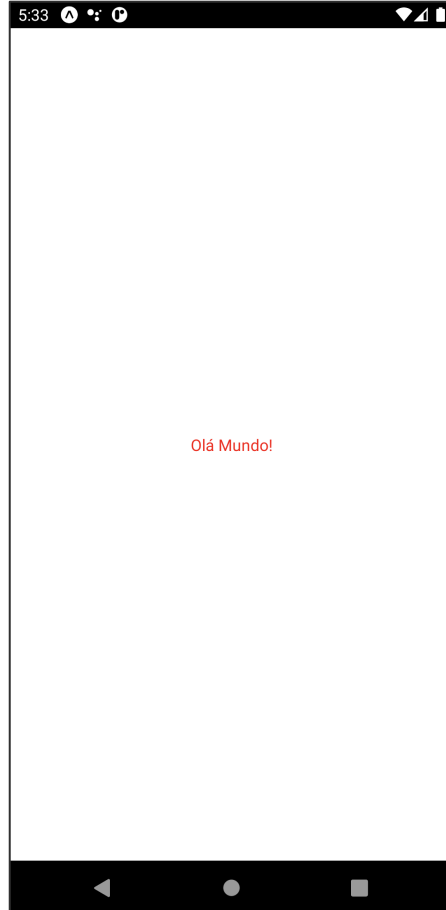
- Utilizamos a API `StyleSheet` do `React Native` para separar nossos estilos de nosso componente.
- Deve-se importar o `StyleSheet` no início do arquivo.

```
const styles = StyleSheet.create({
  container : {
    flex : 1,
    backgroundColor : '#fff',
    alignItems : 'center',
    justifyContent : 'center',
  },
  red : {
    color : 'red',
  },
})
```

Usando Estilos - Exemplo

```
JS App.js > ...
1  import React from 'react';
2  import { StatusBar, StyleSheet, Text, View } from 'react-native';
3
4  export default class App extends React.Component {
5    render() {
6      return (
7        <View style={styles.container}>
8          <Text style={styles.red}>Olá Mundo!</Text>
9          <StatusBar style="light" />
10        </View>
11      );
12    }
13  }
14
15  const styles = StyleSheet.create({
16    container: {
17      flex: 1,
18      backgroundColor: '#fff',
19      alignItems: 'center',
20      justifyContent: 'center',
21    },
22    red: {
23      color: 'red',
24    },
25  });
```

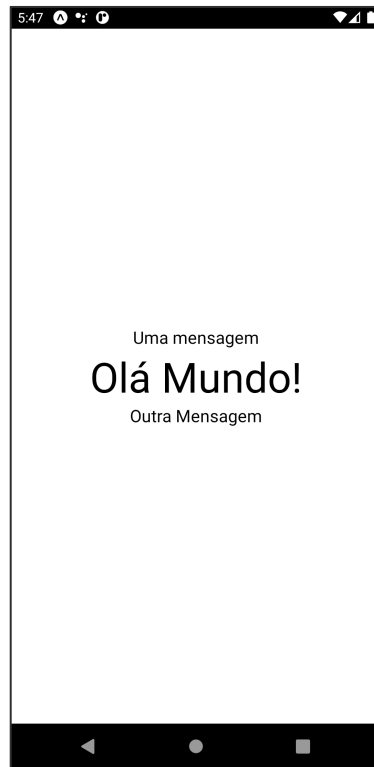
FIAP



- Para adicionar vários estilos a um único componente, podemos passar um array de estilos.
- Ao passar uma array, os estilos no final do array têm precedência sobre os estilos anteriores, no caso de quaisquer atributos repetidos.

Usando Estilos - Array de Estilos - Exemplo

```
JS App.js > [🔍] styles > [🔗] textStyle > [🔗] fontFamily
1  import React from 'react';
2  import { Platform, StatusBar, StyleSheet, Text, View } from 'react-native';
3
4  export default class App extends React.Component {
5    render() {
6      return (
7        <View style={styles.container}>
8          <Text style={[styles.smallText, styles.textStyle]}>Uma mensagem</Text>
9          <Text style={[styles.largeText, styles.textStyle]}>Olá Mundo!</Text>
10         <Text style={[styles.smallText, styles.textStyle]}>Outra Mensagem</Text>
11         <StatusBar style="light" />
12       </View>
13     );
14   }
15 }
16
17 const styles = StyleSheet.create({
18   container: {
19     flex: 1,
20     backgroundColor: '#fff',
21     alignItems: 'center',
22     justifyContent: 'center',
23   },
24   textStyle: {
25     textAlign: 'center',
26     fontFamily: Platform.OS === 'ios' ? 'AvenirNext-Regular' : 'Roboto',
27   },
28   largeText: {
29     fontSize: 44,
30   },
31   smallText: {
32     fontSize: 18,
33   }
34 });
```



Android



iOS

- A API `Platform` nos permite aplicar condicionalmente estilos ou propriedades diferentes em nosso componente com base no sistema operacional do dispositivo.
- O atributo `OS` do objeto retorna `iOS` ou `android`, dependendo do dispositivo do usuário.
- Em vez de aplicar verificações condicionais usando `Platform.OS` várias vezes em todo o arquivo do componente, também podemos aproveitar o uso de arquivos específicos da plataforma. Podemos criar dois arquivos separados para representar o mesmo componente, cada um com uma extensão diferente: **`.ios.js`** e **`.android.js`**.

Platform.select



```
Js App.js > ...
1  import React from 'react';
2  import { StatusBar, StyleSheet, Text, View } from 'react-native';
3
4  export default class App extends React.Component {
5    render() {
6      return (
7        <View style={styles.container}>
8          <Text style={styles.textStyle}>Olá Mundo!</Text>
9          <StatusBar style="light" />
10        </View>
11      );
12    }
13  }
14
15  const styles = StyleSheet.create({
16    container: {
17      flex: 1,
18      backgroundColor: "#fff",
19      alignItems: 'center',
20      justifyContent: 'center',
21    },
22    textStyle: {
23      textAlign: 'center',
24      fontSize: 20,
25      ...Platform.select({
26        ios: {
27          fontFamily: 'AvenirNext-Regular',
28        }, android: {
29          fontFamily: 'Roboto',
30        },
31      }),
32    },
33  });
```

- Os componentes podem conter opcionalmente um estado, isto é, um conjunto mutável e privado de dados.
- Estado é uma ótima maneira de controlar a entrada do usuário, solicitações assíncronas e eventos.
- Utiliza-se o método construtor para inicializar os dados específicos do componente (**estado**).
 - Acessamos as propriedades de estado usando **this.state**
 - Modificamos as propriedades de estado usando o método **this.setState()**

State - Exemplo

```
JS App.js > styles > container
1  import React from 'react';
2  import { StatusBar, StyleSheet, Text, TextInput, View } from 'react-native';
3
4  export default class App extends React.Component {
5    // construtor
6    constructor(props) {
7      super(props);
8      // definição de uma variável de estado
9      this.state = {
10        mensagem: "",
11      };
12    }
13
14    mudouMensagem = msg => {
15      // alteração do valor da variável de estado
16      this.setState({ mensagem: msg });
17    }
18
19    render() {
20      return (
21        <View style={styles.container}>
22          <TextInput style={styles.inputStyle}
23            | onChangeText={this.mudouMensagem}/>
24          /* Uso da variável de estado */
25          <Text style={styles.textStyle}>{this.state.mensagem}</Text>
26          <StatusBar style="light" />
27        </View>
28      );
29    }
30  }
```

```
32  const styles = StyleSheet.create({
33    container: {
34      flex: 1,
35      backgroundColor: "#fff",
36      justifyContent: 'center',
37      padding: 16,
38    },
39    inputStyle: {
40      borderColor: 'gray',
41      borderWidth: 1,
42      height: 32,
43      padding: 5,
44      marginBottom: 32,
45    },
46    textStyle: {
47      textAlign: 'center',
48      fontSize: 20,
49    },
50  });
```

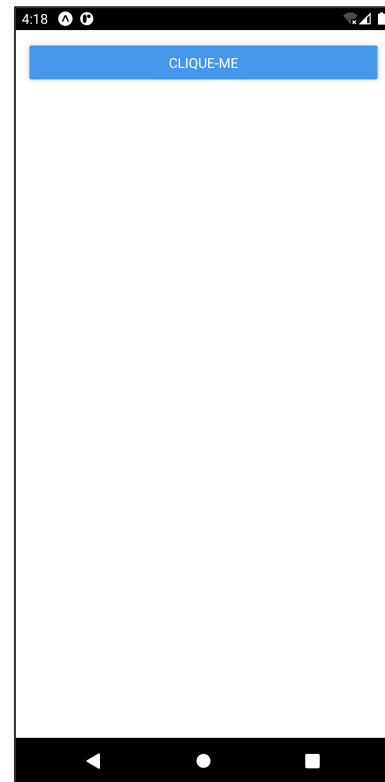
State - Exemplo usando Destructuring de state

```
JS App.js > ...
1  import React from 'react';
2  import { StatusBar, StyleSheet, Text, TextInput, View } from 'react-native';
3
4  export default class App extends React.Component {
5    // construtor
6    constructor(props) {
7      super(props);
8      // definição de uma variável de estado
9      this.state = {
10        mensagem: '',
11      };
12    }
13
14    mudouMensagem = msg => {
15      // alteração do valor da variável de estado
16      this.setState({ mensagem: msg });
17    }
18
19    render() {
20      const { mensagem } = this.state
21      return (
22        <View style={styles.container}>
23          <TextInput style={styles.inputStyle}
24            onChangeText={this.mudouMensagem}/>
25          /* Uso da variável de estado */
26          <Text style={styles.textStyle}>{mensagem}</Text>
27          <StatusBar style="light" />
28        </View>
29      );
30    }
31  }
```

Componente - Button

- Um elemento básico e clicável com poucas opções de personalização.

```
JS App.js > ...
1  import React from 'react';
2  import { Button, StatusBar, View } from 'react-native';
3
4  export default class App extends React.Component {
5
6    render() {
7      return (
8        <View style={{ padding: 16 }}>
9          <Button title={"Clique-me"} />
10         <StatusBar style="light" />
11        </View>
12      );
13    }
14  }
```

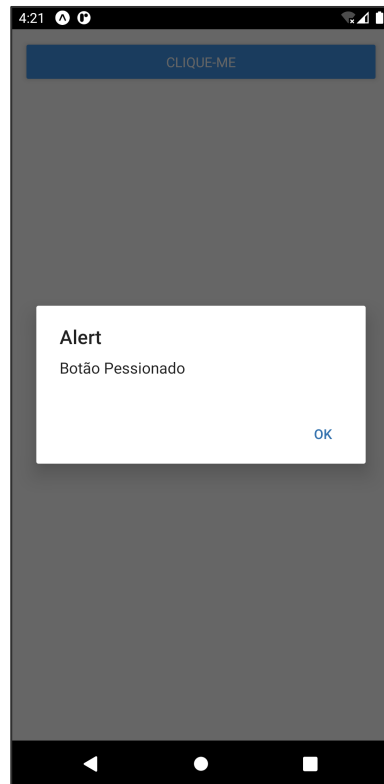


Componente - Button - Propriedades - onPress

- A propriedade **onPress** permite passar uma **function** que será executada assim que o **Button** for pressionado.

JS App.js > ...

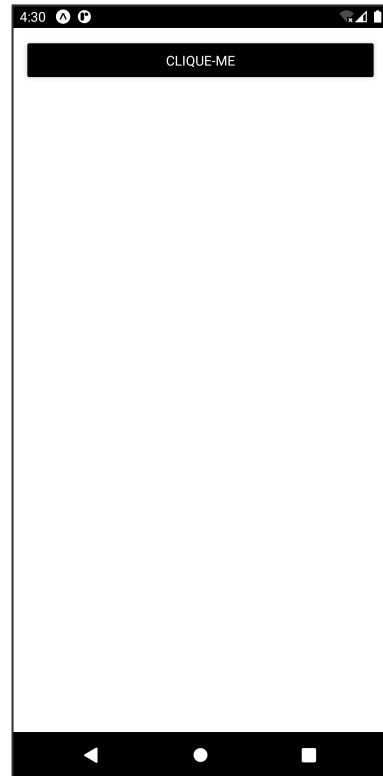
```
1  import React from 'react';
2  import { Button, StatusBar, View } from 'react-native';
3
4  export default class App extends React.Component {
5
6    render() {
7      return (
8        <View style={{ padding: 16 }}>
9          <Button
10            onPress={_ => alert('Botão Pessionado')}
11            title={"Clique-me"} />
12          <StatusBar style="light" />
13        </View>
14      );
15    }
16  }
```



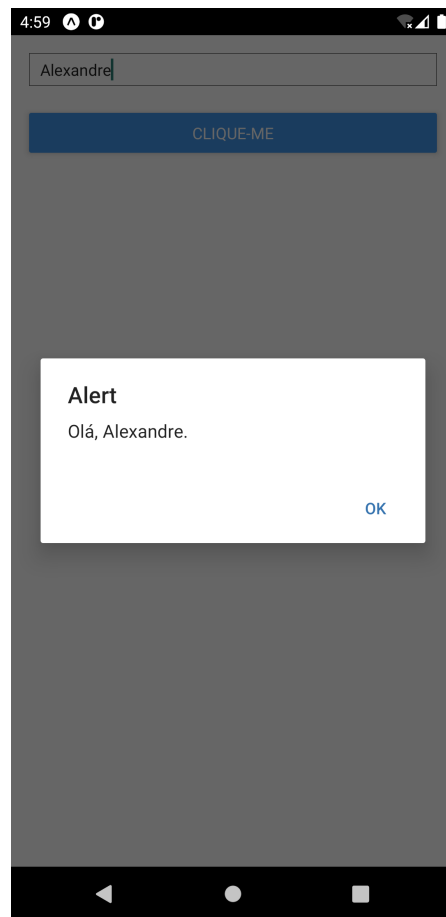
Componente - Button - Propriedades - color

- A propriedade **color** permite escolher a cor do texto (iOS) ou a cor de fundo (Android) do Button.

```
<Button  
  color='#000'  
  onPress={_ => alert('Botão Pessionado')}  
  title={"Clique-me"} />
```



Exemplo



Exemplo

```
JS App.js > ...
1  import React from 'react';
2  import { Button, StatusBar, TextInput, View } from 'react-native';
3
4  export default class App extends React.Component {
5    constructor(props) {
6      super(props);
7      this.state = {
8        nome: ''
9      };
10   }
11
12   cliqueBotao = _ => {
13     const { nome } = this.state;
14     if (!nome) {
15       alert('Informe um nome.');
```

```
22   render() {
23     return (
24       <View style={{ padding: 16 }}>
25         <TextInput
26           placeholder="Informe seu nome."
27           style={{ marginBottom: 24,
28             borderWidth: 0.5, paddingHorizontal: 10 }}
29           onChangeText={nome => this.setState({ nome })} />
30         <Button
31           onPress={this.cliqueBotao}
32           title="Clique-me" />
33         <StatusBar style="light" />
34       </View>
35     );
36   }
37 }
```

+

+

45697096

Dúvidas?



45697096

45697096

