



React-Native - Componentes básicos

Hybrid Mobile App Development

FIAP

Agenda

- View
- Text
- TextInput
- Image
- Button
- TouchableOpacity
- TouchableHighlight
- Switch

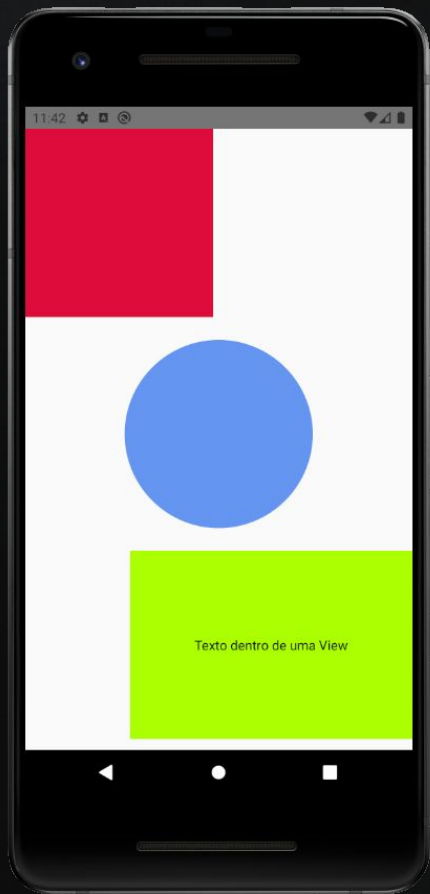
FIAP

Agenda

- View

- Text
- TextInput
- Image
- Button
- TouchableOpacity
- TouchableHighlight
- Switch

View



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**.

View - Código da Imagem anterior

```
JS App.js •
JS App.js > ...
1  import React from 'react'
2  import {
3    Text,
4    View
5  } from 'react-native'
6
7  export default class App extends React.Component {
8    render() {
9      return (
10        <View>
11          <View style={{backgroundColor : 'crimson', height : 200, width : 200}}></View>
12          <View style={{alignSelf : 'center', backgroundColor : 'cornflowerblue', borderRadius : 100, height : 200, marginVertical : 24, width : 200}}></View>
13          <View style={{alignItems : 'center', backgroundColor : 'greenyellow', alignSelf : 'flex-end', height : 200, justifyContent : 'center', width : 300}}>
14            <Text>Texto dentro de uma View</Text>
15          </View>
16        </View>
17      )
18    }
19  }
20
```

Principal propriedade da View

Para referências das cores utilizadas neste exemplo:

<https://reactnative.dev/docs/colors>

FIAP

Agenda

- View

- **Text**

- TextInput

- Image

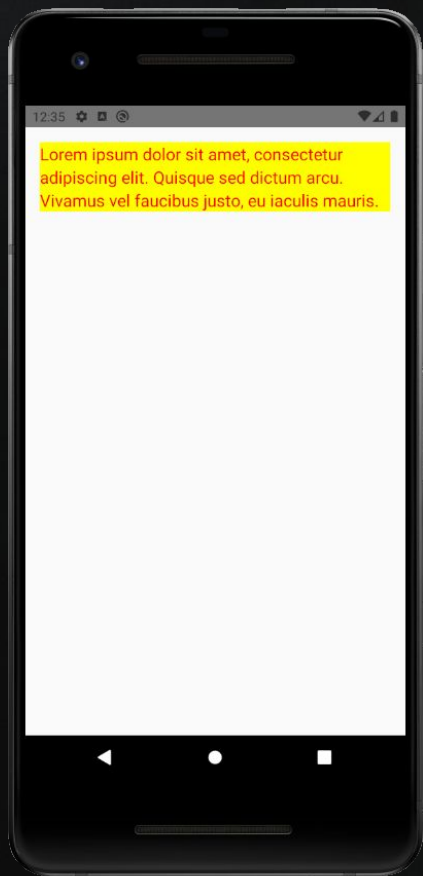
- Button

- TouchableOpacity

- TouchableHighlight

- Switch

Text



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.

Text - Código da imagem anterior

```
JS App.js x
JS App.js > ...
1 import React from 'react'
2 import {
3   Text,
4   View
5 } from 'react-native'
6
7 export default class App extends React.Component {
8   render() {
9     return (
10      <View style={{ padding : 16 }}>
11        <Text style={{ backgroundColor : 'yellow', color: '#F00', fontSize : 18, lineHeight : 25 }}>
12          Lorem ipsum dolor sit amet,
13          consectetur adipiscing elit.
14          Quisque sed dictum arcu.
15          Vivamus vel faucibus justo,
16          eu iaculis mauris.
17        </Text>
18      </View>
19    )
20  }
21 }
22
```

Texto gerado por:

<https://lipsum.com/>

Text - Propriedades - numberOfLines

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



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

```
<Text numberOfLines={1}>
  Lorem ipsum dolor sit amet,
  consectetur adipiscing elit.
  Quisque sed dictum arcu.
  Vivamus vel faucibus justo,
  eu iaculis mauris.
</Text>
```

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

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**.

...dictum arcu. Vivamus vel faucibus justo, eu iaculis mauris.



```
<Text
  ellipsizeMode='head'
  numberOfLines={1}>
  Lorem ipsum dolor sit amet, consectetur
  Quisque sed dictum arcu. Vivamus vel fa
</Text>
```

Lorem ipsum dolor sit amet, c...cibus justo, eu iaculis mauris.



```
<Text
  ellipsizeMode='middle'
  numberOfLines={1}>
  Lorem ipsum dolor sit amet, consectetur
  Quisque sed dictum arcu. Vivamus vel fau
</Text>
```

Text - Propriedades - ellipsizeMode

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quis



```
<Text
  ellipsizeMode='clip'
  numberOfLines={1}>
    Lorem ipsum dolor sit amet, consectetur
    Quisque sed dictum arcu. Vivamus vel fau
  </Text>
```

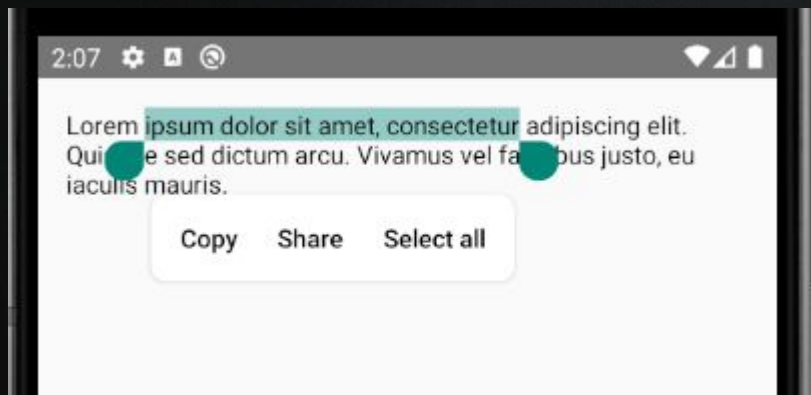
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Qu...



```
<Text
  ellipsizeMode='tail'
  numberOfLines={1}>
    Lorem ipsum dolor sit amet, consectetur a
    Quisque sed dictum arcu. Vivamus vel fauc
  </Text>
```

Text - Propriedade - selectable

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



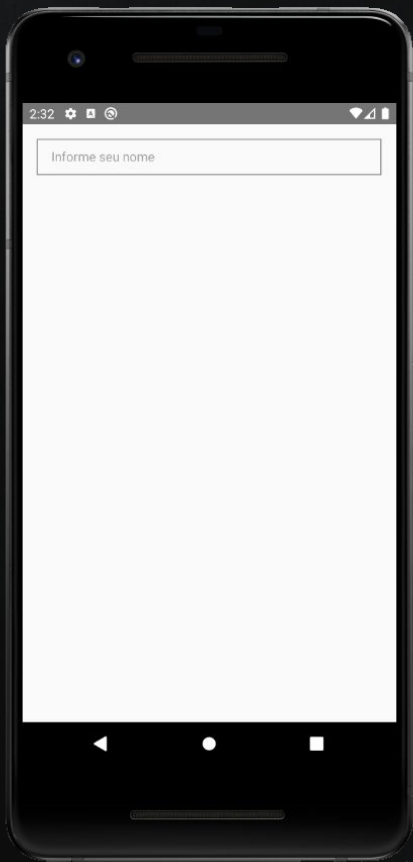
```
<View style={{ padding : 16 }}>  
  <Text selectable={true}>  
    Lorem ipsum dolor sit amet, consectetur  
    Quisque sed dictum arcu. Vivamus vel f  
  </Text>  
</View>
```

FIAP

Agenda

- View
- Text
- **TextInput**
- Image
- Button
- TouchableOpacity
- TouchableHighlight
- Switch

TextInput



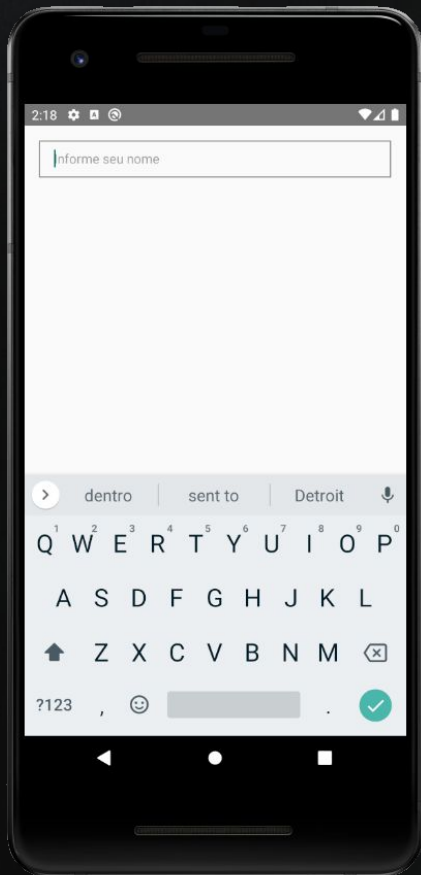
O principal componente para entrada de dados em formato texto pelo usuário.

Por default, possui a estilização simples, sendo necessária a modificação através da propriedade **style**.

TextInput - Código da imagem anterior

```
JS App.js
JS App.js > ...
1  import React from 'react'
2  import {
3    TextInput,
4    View
5  } from 'react-native'
6
7  export default class App extends React.Component {
8    render() {
9      return (
10        <View style={{ padding : 16 }}>
11          <TextInput
12            placeholder="Informe seu nome"
13            style={{
14              borderColor : 'gray',
15              borderWidth : 1,
16              height : 40,
17              paddingHorizontal : 16
18            }} />
19        </View>
20      )
21    }
22  }
23
```


TextInput - Propriedades - autoFocus



Permite que o TextInput já inicie com o cursor dentro, pronto para digitação do texto.

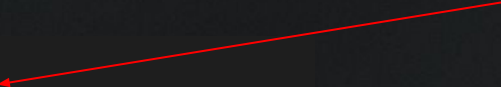
Os valores dessa propriedade pode ser **true** ou **false** (default).

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

TextInput - Propriedades - editable

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

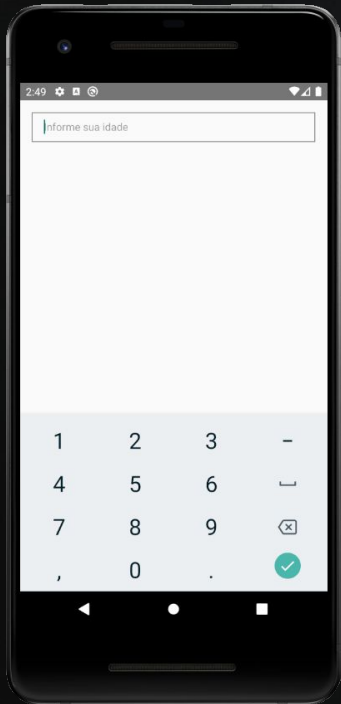
```
<TextInput
  editable={false}
  placeholder="Informe seu nome"
  style={{
    borderColor : 'gray',
    borderWidth : 1,
    height : 40,
    paddingHorizontal : 16
  }} />
```



TextInput - Propriedades - keyboardType

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

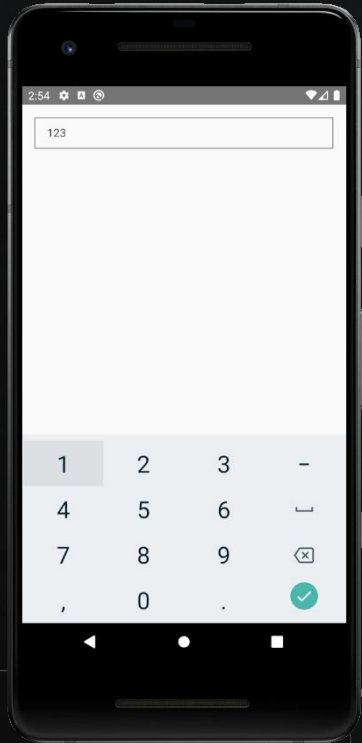
Veja todos as possibilidades em: <https://lefkowitz.me/visual-guide-to-react-native-textinput-keyboardtype-options/>



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

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
  }} />
```

TextInput - Propriedades - multiline / textAlignVertical

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' />
```

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  
  }} />
```



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 function representa o valor dentro do TextInput.

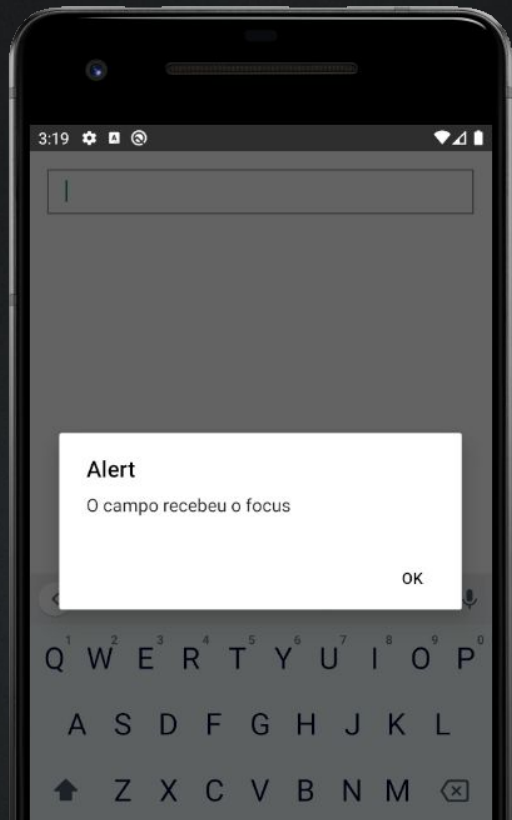
```
<TextInput  
  onChangeText={ value => alert(value) }  
  style={{  
    borderColor : 'gray',  
    borderWidth : 1,  
    height : 40,  
    paddingHorizontal : 16  
  }} />
```



TextInput - Propriedades - onFocus

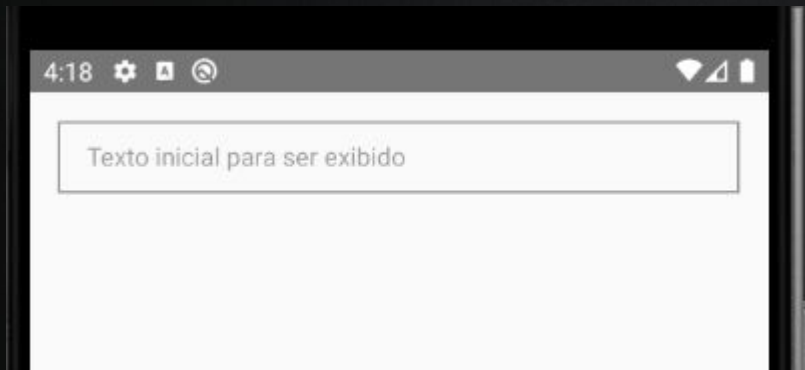
Evento disparado sempre que o TextInput receber o focus. Deve ser informado uma function que será executada quando o evento ocorrer.

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



TextInput - Propriedades - placeholder

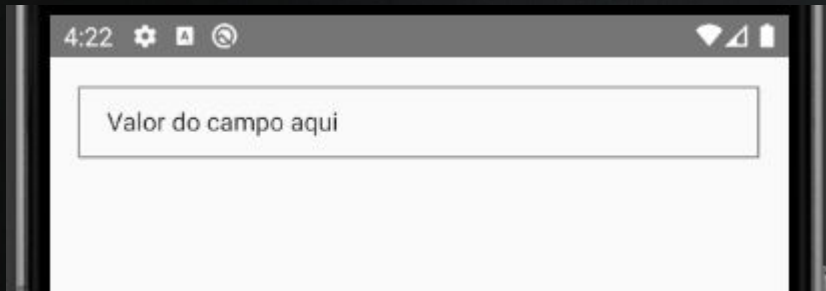
Esta propriedade permite informar a String que será renderizada antes da entrada de algum dado no TextInput.



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

TextInput - Propriedades - value

O valor do TextInput a ser exibido.



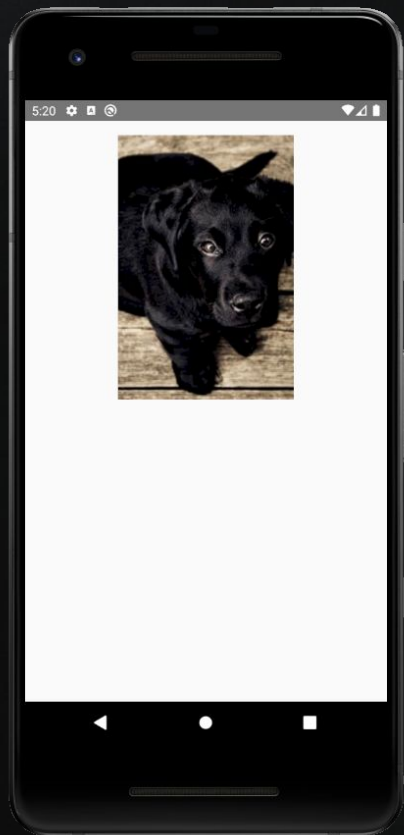
```
<TextInput
  value="Valor do campo aqui"
  style={{
    borderColor : 'gray',
    borderWidth : 1,
    height : 40,
    paddingHorizontal : 16
  }} />
```

FIAP

Agenda

- View
- Text
- TextInput
- **Image**
- Button
- TouchableOpacity
- TouchableHighlight
- Switch

Image



O componente **Image** permite o uso de imagens, sendo elas de uma URL ou de dentro dos arquivos do projeto.

Para testar com imagens aleatórias:

<https://picsum.photos/>

Image - Código da imagem anterior

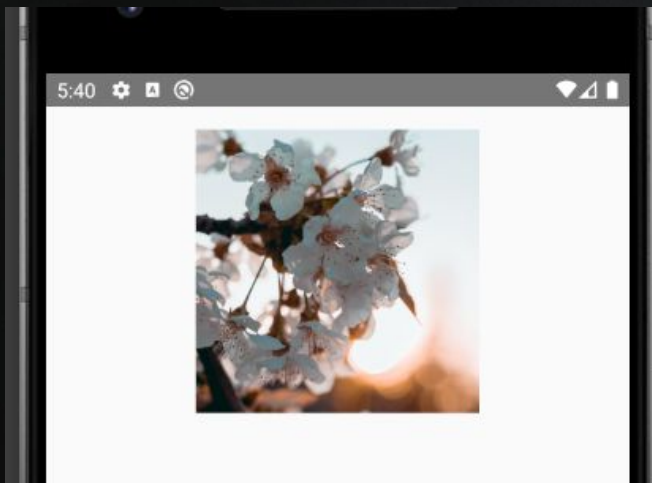
JS App.js

JS App.js > ...

```
1  import React from 'react'
2  import {
3    Image,
4    View
5  } from 'react-native'
6
7  export default class App extends React.Component {
8    render() {
9      return (
10        <View style={{ padding : 16 }}>
11          <Image
12            source={{ uri : 'https://i.picsum.photos/id/237/200/300.jpg' }}
13            style={{ alignSelf : 'center', height : 300, width : 200 }}
14          />
15        </View>
16      )
17    }
18  }
```


Image - Propriedades - source - imagem do projeto

A propriedade **source** representa a origem do arquivo de imagem a ser exibido pelo componente Image. Poderá ser uma URL remota conforme visto anteriormente ou um arquivo dentro do diretório do projeto:



https://unsplash.com/photos/iVB_P7pn2eY
Photo by [Michiel Leunens](#) on [Unsplash](#)

```
JS App.js  x
JS App.js > ...
1  import React from 'react'
2  import {
3    Image,
4    View
5  } from 'react-native'
6
7  import imgFlores from './assets/img/flores.jpg'
8
9  export default class App extends React.Component {
10    render() {
11      return (
12        <View style={{ padding : 16 }}>
13          <Image
14            source={ imgFlores }
15            style={{ alignSelf : 'center', height : 200, width : 200 }}
16          />
17        </View>
18      )
19    }
20  }
21
```


Image - Propriedades - resizeMode

Propriedade que configura a forma que a imagem será redimensionada quando o tamanho do componente **Image** não corresponder com as dimensões da imagem.

Os possíveis valores são: **cover**, **contain**, **stretch**, **repeat** e **center**.

```
<Image
  resizeMode='repeat'
  source={{ uri : 'https://i.picsum.photos/id/237/200/300.jpg' }}
  style={{ alignSelf : 'center', height : 200, width : 200 }}
/>
```

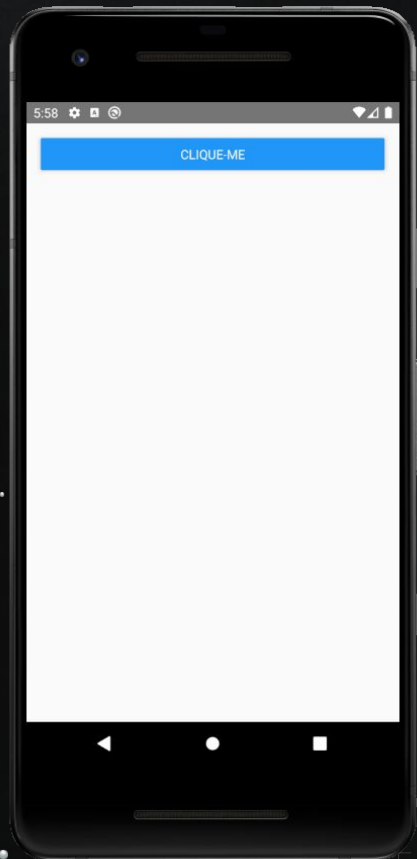


FIAP

Agenda

- View
- Text
- TextInput
- Image
- **Button**
- TouchableOpacity
- TouchableHighlight
- Switch

Button



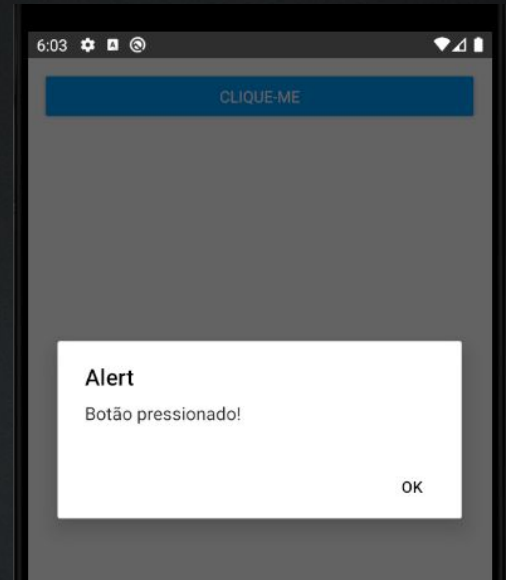
Um elemento básico e clicável com poucas opções de personalização.

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

Button - Propriedades - onPress

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

```
<Button  
  onPress={ _ => alert('Botão pressionado!') }  
  title="Clique-me" />
```



Button - Propriedades - color

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

```
<Button  
  color="#000"  
  title="Clique-me" />
```



FIAP

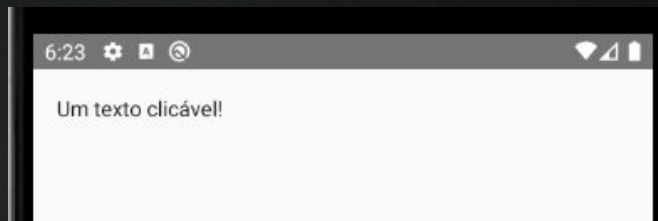
Agenda

- View
- Text
- TextInput
- Image
- Button
- **TouchableOpacity**
- TouchableHighlight
- Switch

TouchableOpacity

```
JS App.js > ...
1  import React from 'react'
2  import {
3    Text,
4    TouchableOpacity,
5    View
6  } from 'react-native'
7
8  export default class App extends React.Component {
9    render() {
10     return (
11       <View style={{ padding : 16 }}>
12         <TouchableOpacity>
13           <Text>Um texto clicável!</Text>
14         </TouchableOpacity>
15       </View>
16     )
17   }
18 }
```

O **TouchableOpacity** é um container clicável para outros componentes e quando pressionado, a opacidade dos componentes filhos tem o valor diminuído.



TouchableOpacity - Propriedades - onPress

Esta propriedade permite acionar uma function ao pressionar o componente:

```
<TouchableOpacity  
  onPress={ _ => alert('Pressionado!') }>  
  <Text>Um texto clicável!</Text>  
</TouchableOpacity>
```

TouchableOpacity - Propriedades - activeOpacity

A propriedade **activeOpacity** permite o nível de opacidade ao pressionar.

A variação pode ser de 0 à 1.

O valor default é: 0.2

```
<TouchableOpacity  
  activeOpacity={ 0.05 }>  
  <Text>Um texto clicável!</Text>  
</TouchableOpacity>
```

FIAP

Agenda

- View
- Text
- TextInput
- Image
- Button
- TouchableOpacity
- **TouchableHighlight**
- Switch

TouchableHighlight

Container parecido com o TouchableOpacity, com a diferença que permite escolher a cor de fundo para destaque ao ser pressionado.

```
import React from 'react'
import {
  Text,
  TouchableHighlight,
  View
} from 'react-native'

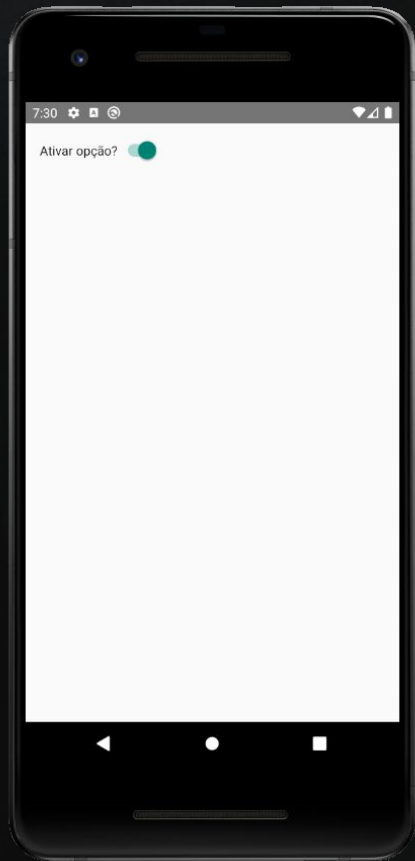
export default class App extends React.Component {
  render() {
    return (
      <View style={{ padding : 16 }}>
        <TouchableHighlight
          activeOpacity={ 0.6 }
          onPress={ _ => alert('Pressionado!') }
          underlayColor="#333">
          <Text>Um texto clicável!</Text>
        </TouchableHighlight>
      </View>
    )
  }
}
```

FIAP

Agenda

- View
- Text
- TextInput
- Image
- Button
- TouchableOpacity
- TouchableHighlight
- **Switch**

Switch



O Switch renderiza um componente booleano equivalente ao Checkbox que estamos acostumados.

Obrigatoriamente necessita da implementação da propriedade **onValueChange** para atualização do **value** para que seja refletida a alteração do usuário.

Neste caso, é necessário o uso do **state**.

Switch - Código do exemplo anterior

```
JS App.js x
JS App.js > ...
1  import React from 'react'
2  import {
3    Switch,
4    Text,
5    View
6  } from 'react-native'
7
8  export default class App extends React.Component {
9
10   state = {
11     isEnabled : false
12   }
13
14   render() {
15     return (
16       <View style={{ padding : 16, flexDirection : "row", alignItems : 'center' }}>
17         <Text>Ativar opção?</Text>
18         <Switch
19           value={this.state.isEnabled}
20           onChange={ _ => this.setState( { isEnabled : !this.state.isEnabled } }
21         />
22       </View>
23     )
24   }
25 }
```


Switch - Propriedades - thumbColor

Permite colorizar o controle do Switch:

```
<Switch  
  value={this.state.isEnabled}  
  onValueChange={ _ => this.setState( { isEnabled : !this.state.isEnabled } }  
  thumbColor='#F00'  
/>
```

Ativar opção?



Switch - Propriedades - trackColor

Permite colorizar o trilho do Switch para cada estado possível do **value**:

```
<Switch
  value={this.state.isEnabled}
  onValueChange={ _ => this.setState( { isEnabled : !this.state.isEnabled } }
  trackColor={{ false : '#333', true : '#F00' }}
/>
```

Ativar opção? 

Ativar opção? 



Copyright © 2020 Prof. Douglas Cabral <douglas.cabral@fiap.com.br> <https://www.linkedin.com/in/douglascabral/>

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).