

# 1. Introdução

Nos cursos anteriores aprendemos a criar aplicativos que nos permitem exibir algo para o usuário. Neste curso permitiremos ao usuário inserir dados no aplicativo através do componente nativo **TextInput**.

O aplicativo que vamos criar pode ser visto na Imagem 1



## O que vamos aprender

Neste curso você aprenderá a manipular o componente **TextInput** e atribuir a ele as seguintes propriedades:

- `textAlign`
- `keyboardType`
- `value`
- `autoFocus`
- `maxLength`
- `onChangeText`
- `onFocus`

- o onBlur

## Entendendo o componente TextInput

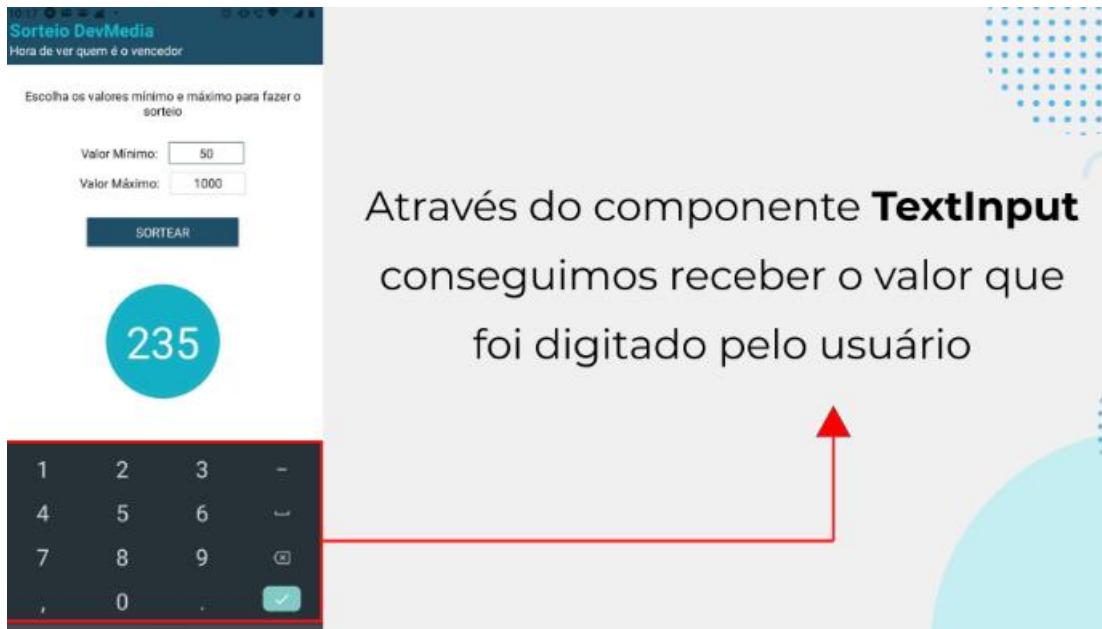




Por que esse componente é útil?

**TextInput**  
**Por que é útil?**

É através do componente **TextInput** que o usuário consegue enviar um texto para o aplicativo



Através do componente **TextInput** conseguimos receber o valor que foi digitado pelo usuário

## 2. Entendendo o exemplo

Já aprendemos que para receber um valor digitado pelo usuário precisamos utilizar o componente `TextInput`. Entenda como faremos isso no nosso aplicativo

**NOSSO  
EXEMPLO**

## 1º Passo

Vamos inserir na tela 2 componentes **TextInput**



## 2º Passo

Vamos passar algumas propriedades para alterar a forma como o **TextInput** vai ser exibido

The screenshot shows a user interface for a lottery draw. At the top, it says "Sorteio DevMedia" and "Hora de ver quem é o vencedor". Below that, it says "Escolha os valores mínimo e máximo para fazer o sorteio". There are two input fields: "Valor Mínimo:" with the value "50" and "Valor Máximo:" with the value "1000". A "SORTEAR" button is below them. In the center, there is a blue circle containing the number "235". At the bottom is a numeric keypad with digits 1-9, a decimal point ., and a checkmark button.

Annotations:

- A red arrow points from the text "Permitir digitar apenas 4 dígitos" to the numeric keypad.
- A red arrow points from the text "Texto centralizado" to the "235" circle.
- A red arrow points from the text "O tipo de teclado" to the numeric keypad.

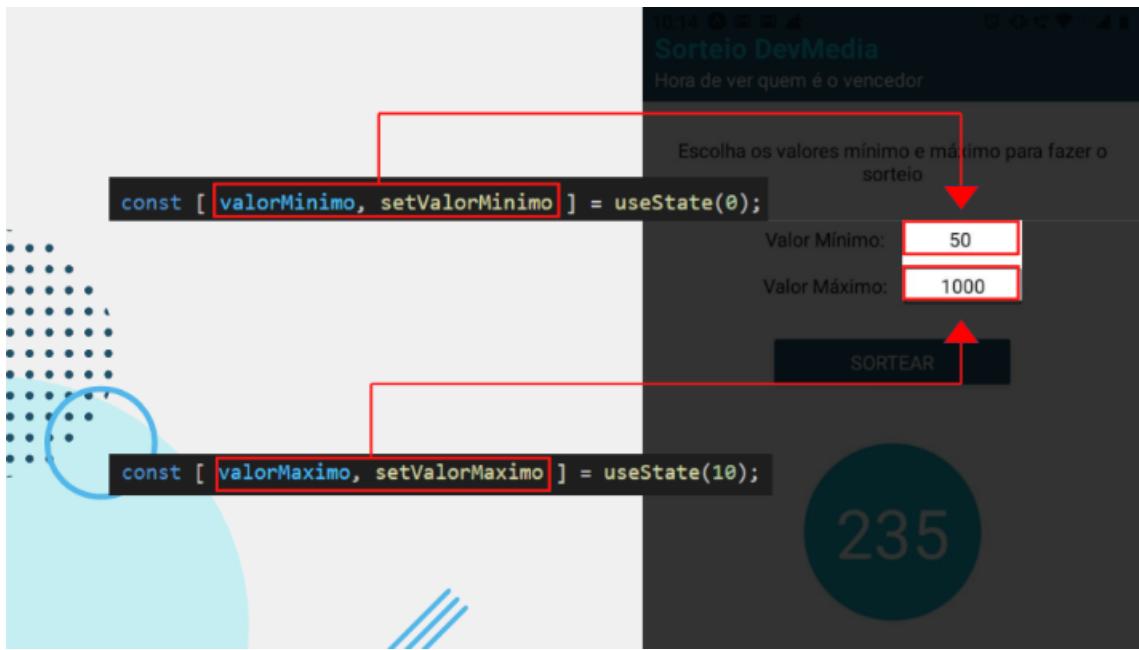
## 3º Passo

Vamos criar uma variável de estado para cada **TextInput**

The screenshot shows the same lottery draw interface. The "Valor Mínimo:" field now has a red border and contains the value "50". The "Valor Máximo:" field also has a red border and contains the value "1000". The "SORTEAR" button has a red border and contains the text "valorMaximo". In the center, there is a dark green circle containing the number "235".

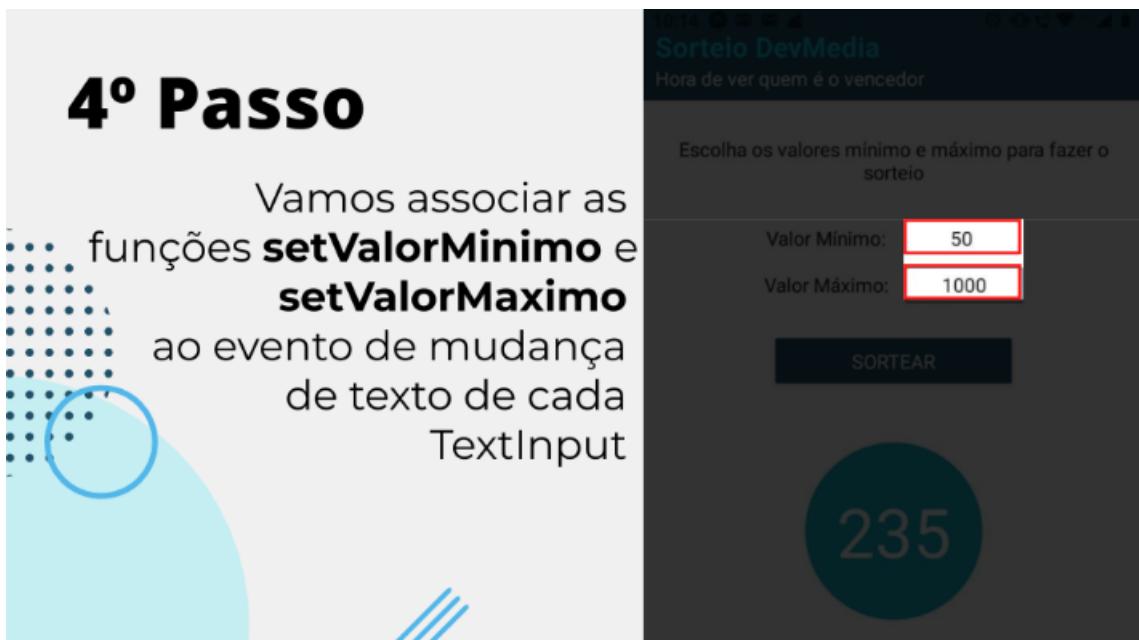
Annotations:

- A red arrow points from the text "valorMinimo" to the "50" input field.
- A red arrow points from the text "valorMaximo" to the "1000" input field.
- A red arrow points from the text "valorMaximo" to the "SORTEAR" button.



## 4º Passo

Vamos associar as funções **setValorMinimo** e **setValorMaximo** ao evento de mudança de texto de cada TextInput





Dessa forma para cada caractere digitado, a função (**setValorMinimo** ou **setValorMaximo**) será chamada

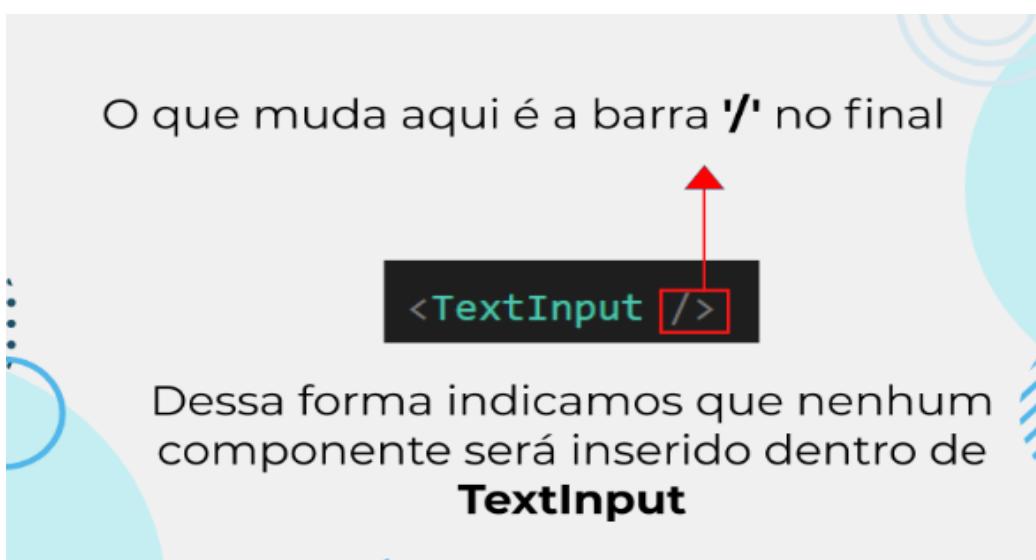
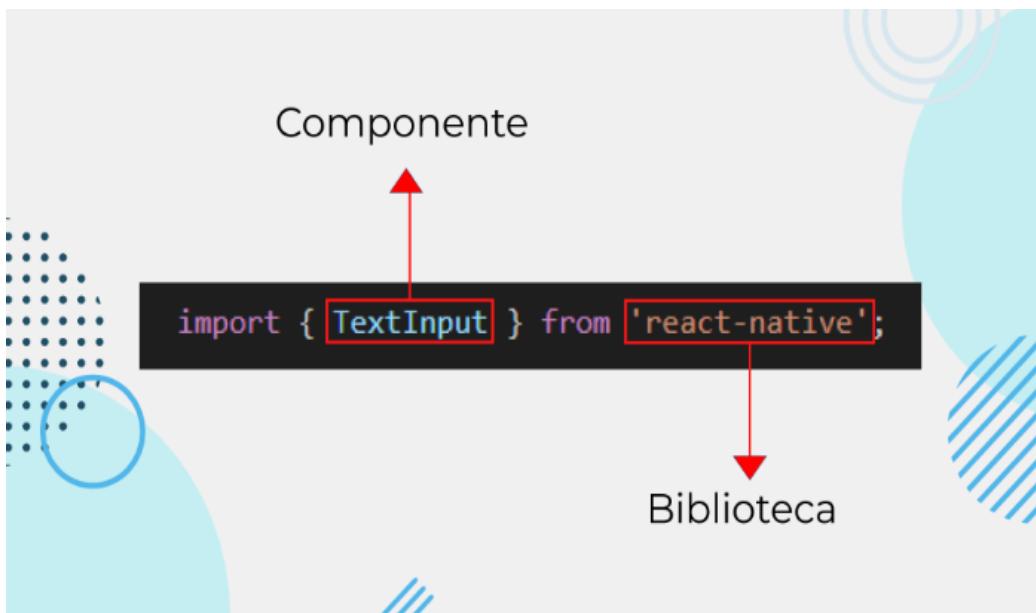
Sempre que o usuário digitar um valor em TextInput precisamos armazená-lo em uma variável de estado

### 3. Criando o TextInput

Agora que conhecemos a mecânica por trás de um componente TextInput, aprenderemos a exibi-lo na tela.

**TextInput**  
Importando e exibindo na tela

**TextInput** é um componente nativo e é importado da biblioteca **react-native**



# 4. Propriedades de TextInput

O componente TextInput possui algumas propriedades que permitem alterar suas características. Veja a seguir algumas delas e os seus possíveis valores:



## textAlign

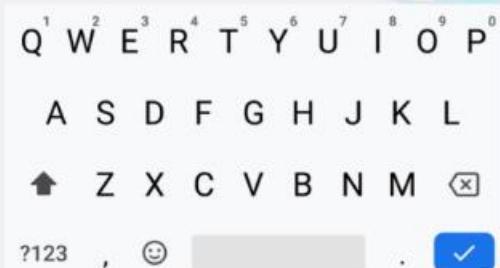
Valores

<code>0</code>	<code>→ left</code>
<code>50</code>	<code>→ center</code>
<code>100</code>	<code>→ right</code>

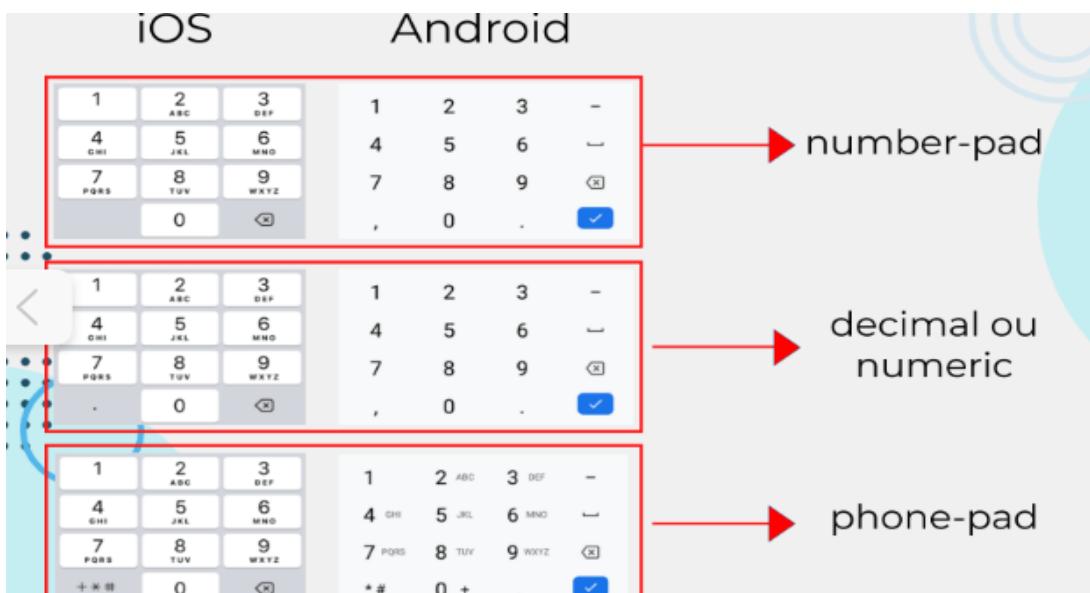
O valor será **left** quando não definido

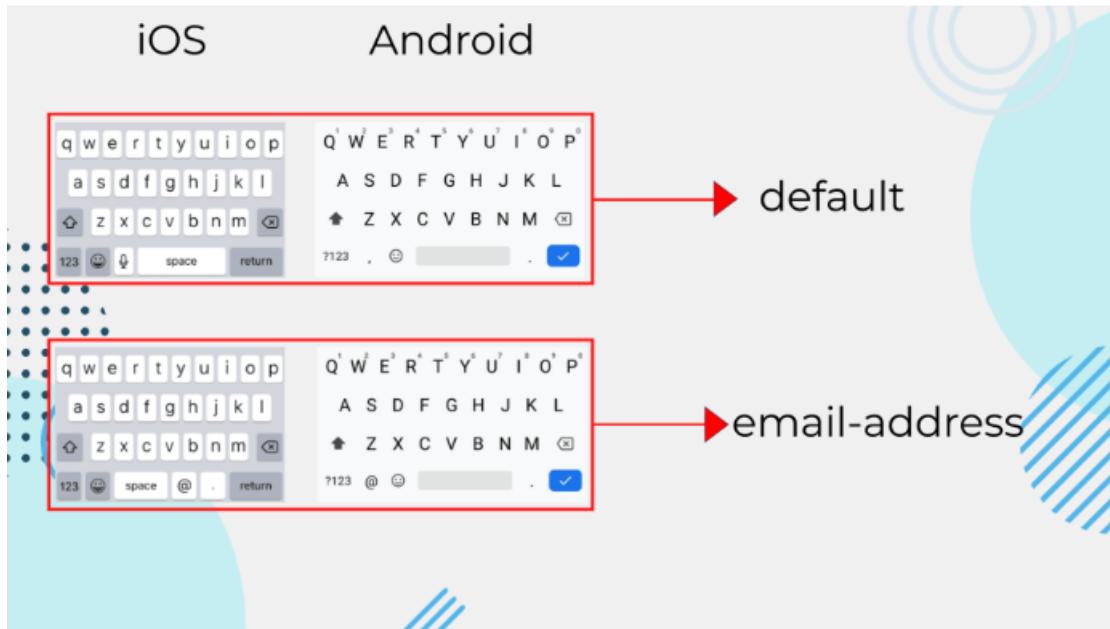
# keyboardType

```
<TextInput  
|  keyboardType="default"
```



O estilo de teclado que será exibido





## autoFocus

```
<TextInput  
  autoFocus={true}
```

Recebe um booleano  
**(true ou false)**

Ao carregar a tela o foco vai estar neste  
componente caso o valor seja **true**.  
O valor padrão é **false**

## maxLength

```
<TextInput  
  maxLength={5}>
```

Recebe um **number**

Indica a quantidade de caracteres que este campo pode receber

## value

```
<TextInput  
  value={texto}>
```

Recebe uma **string**

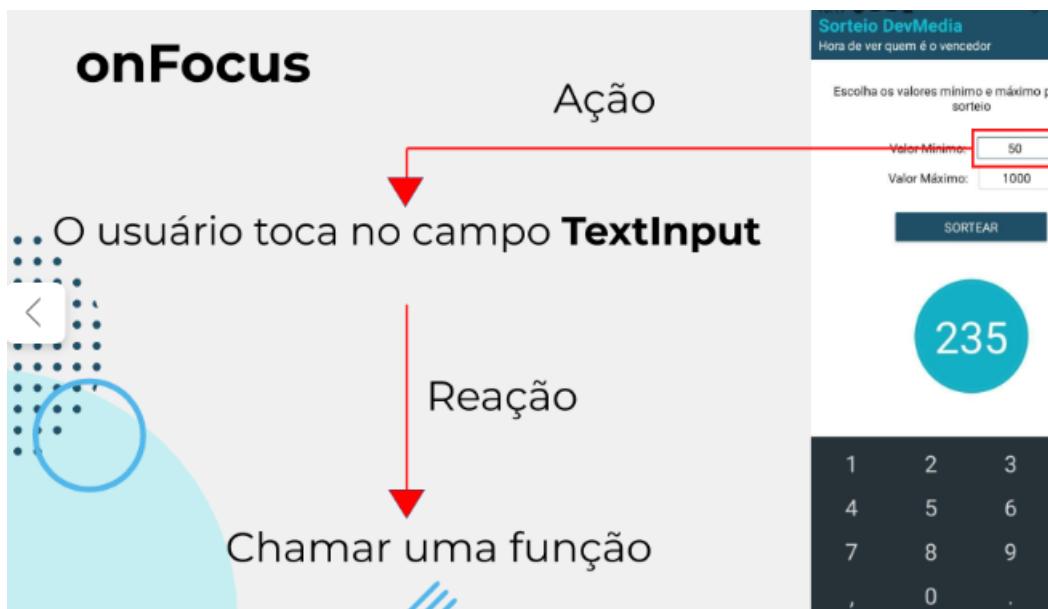
Caso atribuído um valor à ela deverá ser uma variável de estado. Dessa forma será possível alterar o seu valor quando o evento **onChangeText** for chamado  
(que veremos na aula à seguir)

A propriedade `keyboardType` possui outros valores, porém eles não servirão para todas as plataformas. Os valores apresentados nesta aula servirão tanto para Android quanto para iOS

# 5. Eventos de TextInput

Além das propriedades que aprendemos na aula anterior existem outras que representam eventos que serão disparados quando alguma coisa acontecer.

Veja no slide abaixo alguns desses eventos:



## onFocus

```
<TextInput  
  onFocus={() => setFlagInputFocus("txt_min")}
```

Função que será chamada quando **TextInput** entrar em foco

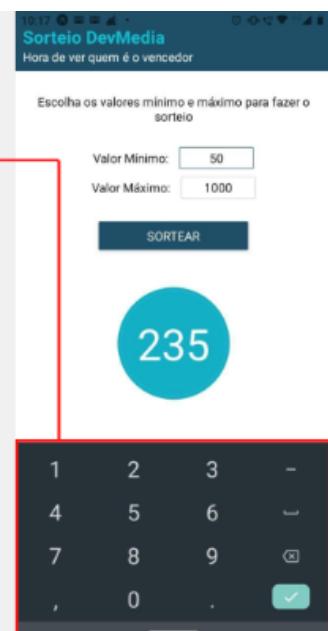
## onChangeText

Ação

O usuário digita no teclado e o valor de **TextInput** muda

Reação

Chamar uma função



## onChangeText

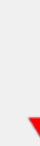
```
<TextInput  
|  onChangeText = { texto => setValorMinimo(texto) }
```



Função que será chamada quando o valor de **TextInput** mudar

## onChangeText

```
<TextInput  
|  onChangeText = { texto => setValorMinimo(texto) }
```



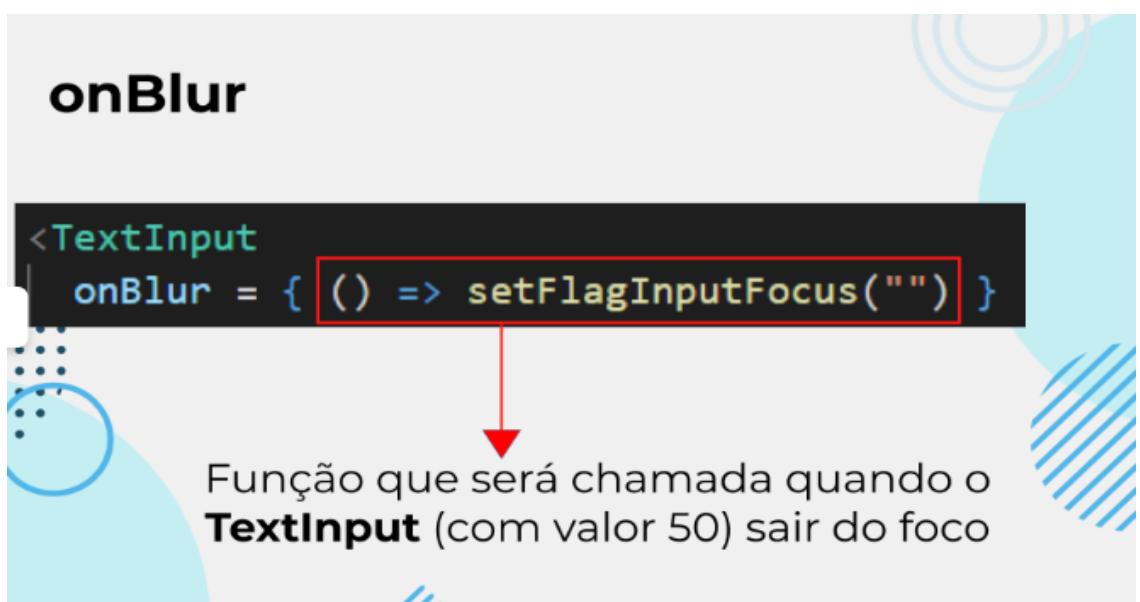
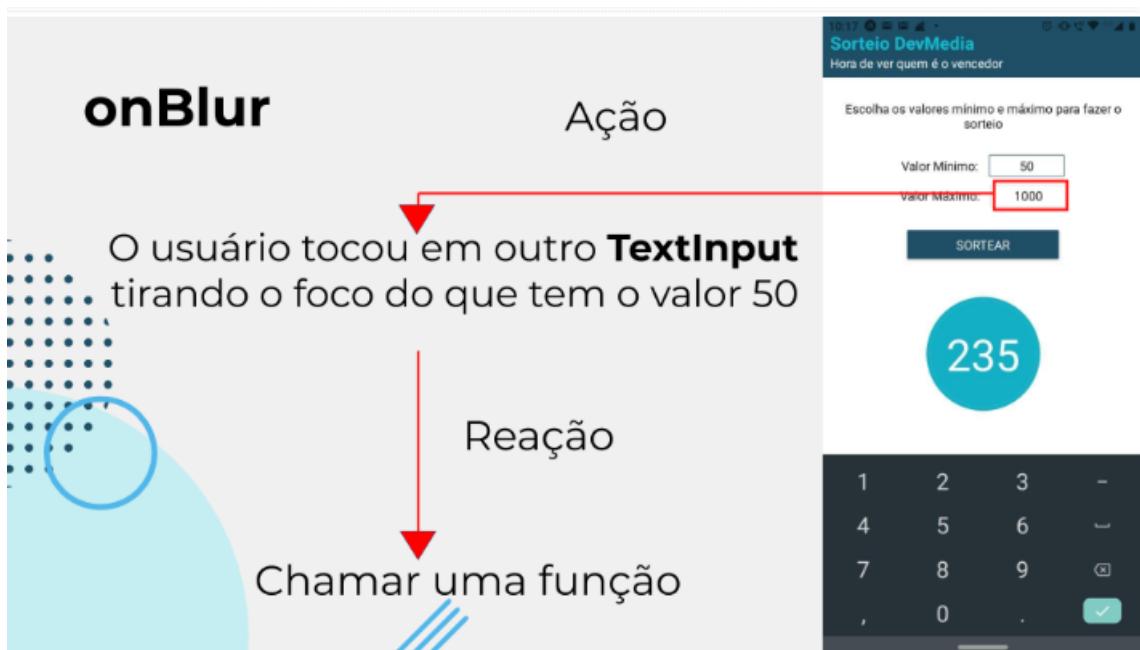
Função que será chamada quando o valor de **TextInput** mudar

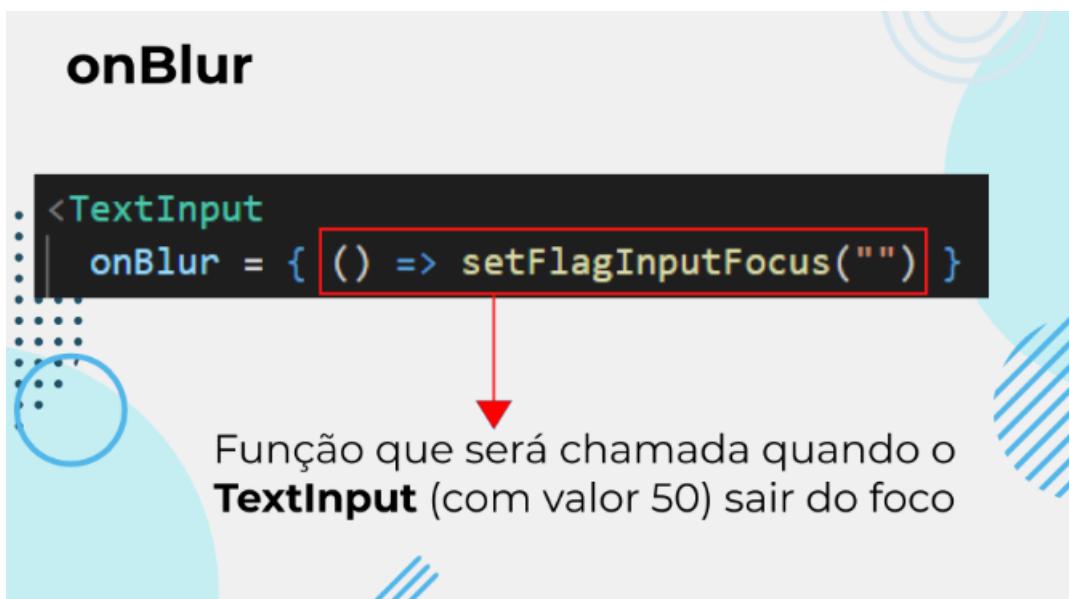
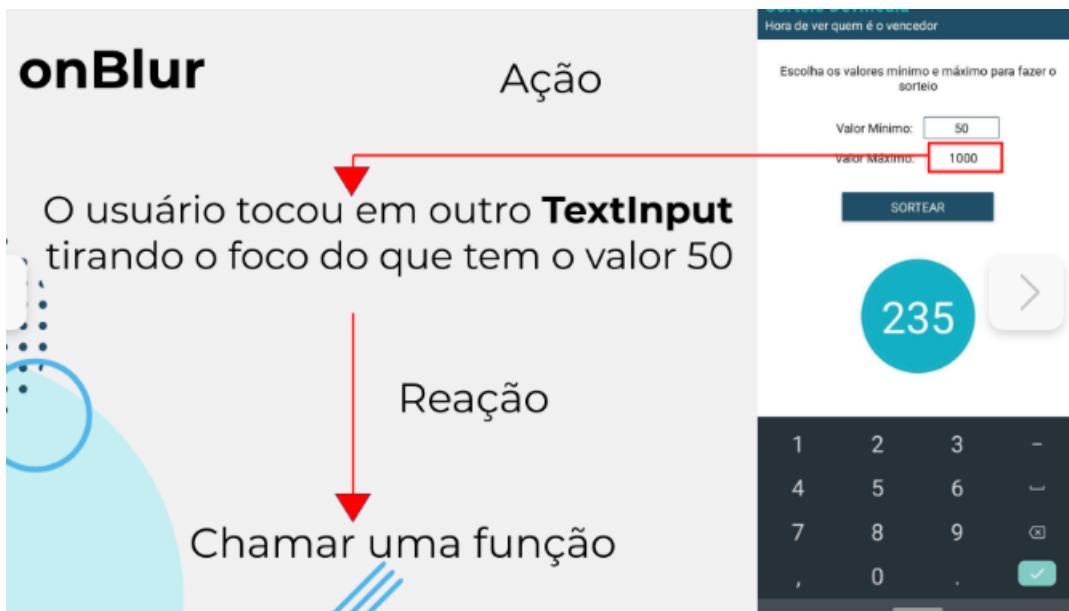
## onChangeText

```
<TextInput  
|  onChangeText = { valor => setValorMinimo(valor) }
```



Esse evento possui um parâmetro (com o novo valor do **TextInput**) que passaremos para a função que será chamada.  
Apelidamos esse parâmetro de **valor**





Quando utilizamos o componente `TextInput` significa que queremos receber um valor digitado pelo usuário. O evento `onChangeText` e uma variável de estado é de suma importância para que esse valor seja manipulado

O evento `onChangeText` possui uma forma resumida para ser declarada.

Veja no Código 1 a forma usada na explicação do slide:

```
<TextInput  
  onChangeText = { valor => setValorMinimo(valor)} />
```

Este código pode ser reduzido passando apenas o nome da função que deve ser executada quando o evento for invocado. Veja como ficaria:

```
<TextInput  
  onChangeText = { setValorMinimo } />
```

Dessa forma o evento entende que deve passar o parâmetro que ele recebe (o texto digitado) para a função definida entre chaves (no nosso caso setValorMinimo).

Os eventos nos permitem chamar uma função quando alguma coisa acontecer

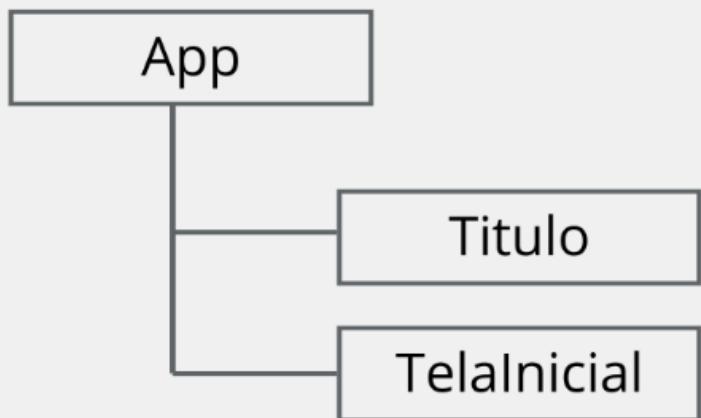
## 6. Arquitetura do projeto

A estrutura do nosso projeto é apresentada na Figura 1.



Figura 1. Estrutura base do projeto

## Em resumo:



Com exceção de `App`, todos os componentes são estilizados através do arquivo `estilo.js` nas suas respectivas pastas.

Veja a seguir o comportamento do nosso aplicativo:

**Nosso aplicativo**  
**Comportamento**  
**desejado**

The background features abstract graphic elements: a large teal circle on the right, a white circle with blue diagonal stripes above it, and a blue dotted pattern at the bottom right. The text "Nosso aplicativo" is in bold black, while "Comportamento" and "desejado" are in a matching teal color.



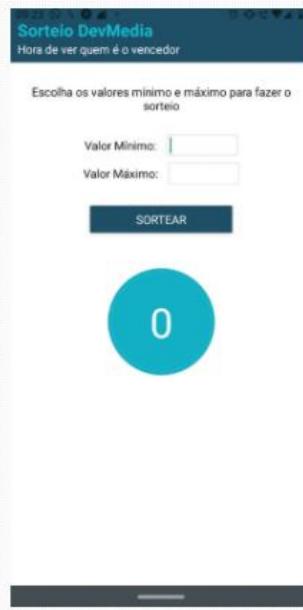
Quando o usuário digitar, o valor do **TextInput** e da variável de estado **valorMinimo** são alterados





## 8. Exibindo o componente TextInput

Nesta aula vamos exibir os campos para o que o usuário insira os valores mínimo e máximo. Para isso utilizaremos o componente `TextInput`. Além disso, vamos alterar o texto exibido para o usuário para ter a aparência da **Figura 1**.



**Figura 1.** Layout do aplicativo

Faremos as alterações no arquivo do componente `TelaInicial`

Faremos as alterações no arquivo do componente `TelaInicial` (`componentes/TelaInicial/index.js`), que tem o código apresentado no **Código 1**.

```
1 import React, { useState } from 'react';
2 import { View, Text, Button, TextInput } from 'react-native';
3 import estilos from './estilos';
4
5 const TelaInicial = () => {
6   const [ numeroSorteado, setNumeroSorteado ] = useState(0);
7
8   const gerarNumero = () => {
9     const novoNumero = Math.floor(Math.random() * 100 + 1);
10    setNumeroSorteado(novoNumero);
11  }
12
13  return (
14    <View style = { estilos.tela }>
15      <Text style = { estilos.titulo }>
16        Escolha os valores mínimo e máximo para fazer o sorteio
17      </Text>
18
19      <View style = { estilos.linhaInput }>
20        <Text>Valor Mínimo: </Text>
21
22        <TextInput style = { estilos.inputNormal }/>
23      </View>
24
25      <View style = { estilos.linhaInput }>
26        <Text>Valor Máximo: </Text>
27
28        <TextInput style = { estilos.inputNormal }/>
29      </View>
30
31      <View style = { estilos.boxBotao }>
32        <Button title="Sortear" onPress = { gerarNumero } color="#1f4f66"/>
33      </View>
34
35      <View style = { estilos.boxNumero }>
36        <Text style = { estilos.numero }>{ numeroSorteado }</Text>
37      </View>
38    );
39  );
40}
41
42 export default TelaInicial
```

**Código 1** Código do componente `TelaInicial` alterado

Entenda o que inserimos no slide abaixo:

# Entendendo o código

## componentes/TelaInicial/index.js

```
1 import React, { useState } from 'react';
2 import { View, Text, Button, TextInput } from 'react-native';
3 import estilos from './estilos';
4
5 const TelaInicial = () => {
6   const [ numeroSorteado, setNumeroSorteado ] = useState(0);
7   Importamos o componente TextInput
8   const gerarNumero = () => {
9     const novoNumero = Math.floor(Math.random() * 100 + 1);
10    setNumeroSorteado(novoNumero);
11  }
12
13  return () {
14    <View style = { estilos.tela }>
15      <Text style = { estilos.titulo }>
16        Escolha os valores mínimo e máximo para fazer o sorteio
17      </Text>
18    
```

```
1 import React, { useState } from 'react';
2 import { View, Text, Button, TextInput } from 'react-native';
3 import estilos from './estilos';
4
5 const TelaInicial = () => {
6   const [ numeroSorteado, setNumeroSorteado ] = useState(0);
7
8   const gerarNumero = () => {
9     const novoNumero = Math.floor(Math.random() * 100 + 1);
10    setNumeroSorteado(novoNumero);
11  }
12
13  return () {
14    <View style = { estilos.tela }>
15      <Text style = { estilos.titulo }>
16        Alteramos o texto exibido
17      </Text>
18      Escolha os valores mínimo e máximo para fazer o sorteio
19    
```

```

19   <View style = { estilo.linhaInput }>
20     <Text>Valor Mínimo: </Text>
21
22     <TextInput style = { estilo.inputNormal }/>
23   </View>
24
25   <View style = { estilo.linhaInput }>
26     <Text>Valor Máximo: </Text>
27
28     <TextInput style = { estilo.inputNormal }/>
29   </View>
30
31     Passamos para ele o estilo estilo.linhaInput
32     <View style = { estilo.boxBotao }>
33       <Button title="Sortear" onPress = { gerarNumero } color="#1f4f66"/>
34     </View>
35
36     <View style = { estilo.boxNumero }>
37       <Text style = { estilo.numero }>{ numeroSorteado }</Text>
38     </View>
39   );
40 }

```

```

19   <View style = { estilo.linhaInput }>
20     <Text>Valor Mínimo: </Text>
21
22     <TextInput style = { estilo.inputNormal }/>
23   </View>
24
25   <View style = { estilo.linhaInput }>
26     <Text>Valor Máximo: </Text>
27
28     Inserimos o componente TextInput e passamos para ele o estilo
29     inputNormal que vamos criar na próxima aula.
30   </View>
31
32     Através deste TextInput o usuário vai digitar o valor mínimo.
33     <Button title="Sortear" onPress = { gerarNumero } color="#1f4f66"/>
34   </View>
35
36   <View style = { estilo.boxNumero }>
37     <Text style = { estilo.numero }>{ numeroSorteado }</Text>
38   </View>
39 </View>
40

```

```

19   <View style = { estilo.linhaInput }>
20     <Text>Valor Mínimo: </Text>
21
22     <TextInput style = { estilo.inputNormal }/>
23   </View>
24
25   <View style = { estilo.linhaInput }>
26     <Text>Valor Máximo: </Text>
27
28     <TextInput style = { estilo.inputNormal }/>
29   </View>
30
31   <View style = { estilo.boxBotao }>
32     <Button title="Sortear" onPress = { gerarNumero } color="#1f4f66"/>
33   </View>
34
35   <View style = { estilo.boxNumero }>
36     <Text style = { estilo.numero }>{ numeroSorteado }</Text>
37   </View>
38 </View>
39

```

# 9. Estilizando o componente TelaInicial

Na aula anterior fizemos algumas alterações no layout da nossa tela (componente TelaInicial) e sua aparência pode ser vista na **Figura 1**.

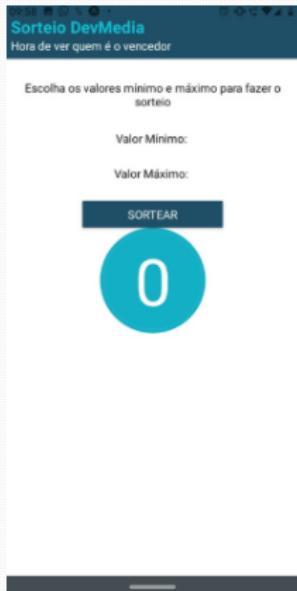
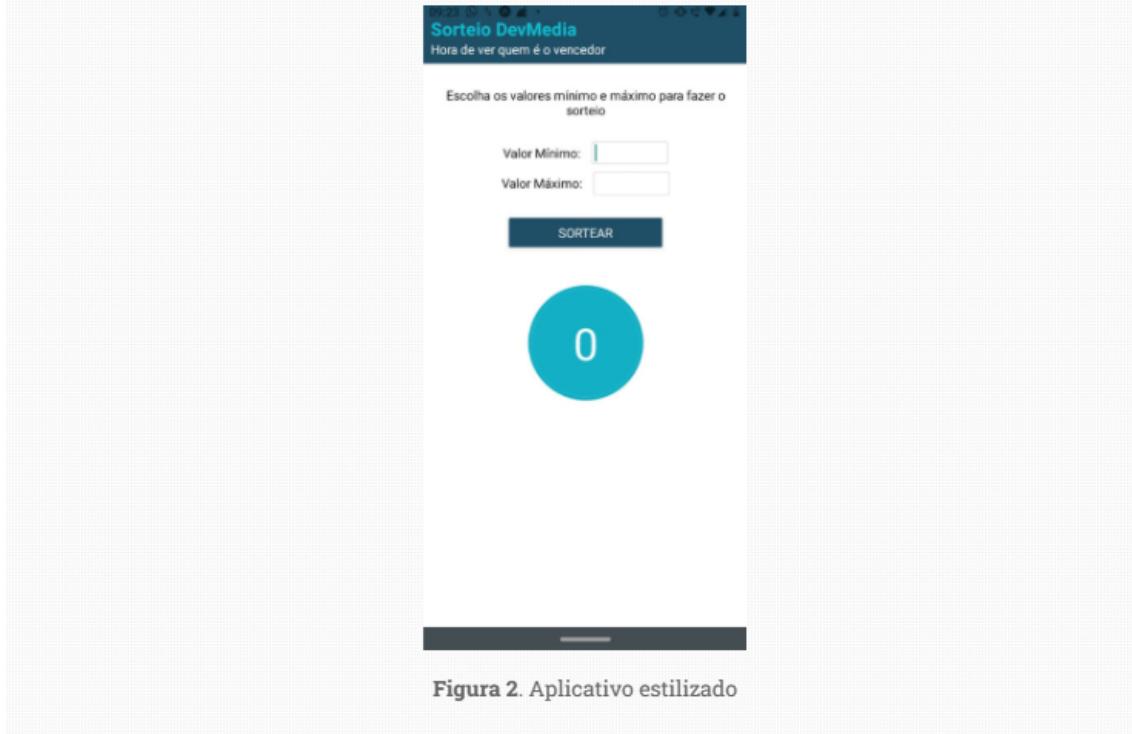


Figura 1. Aplicativo sem estilizar

Nesta aula vamos estilizar nosso componente para ter a aparência vista na **Figura 2**.



**Figura 2.** Aplicativo estilizado

Para mudar o estilo do componente `TelaInicial` alterarmos o arquivo `estilo.js` localizado na mesma pasta do componente (componentes/TelaInicial/estilo.js), conforme visto no Código 1.

```
1 | import { StyleSheet } from "react-native";
2 |
3 | const estilo = StyleSheet.create({
4 |   tela: {
5 |     width: '100%',
6 |     justifyContent: "center",
7 |     alignItems: 'center',
8 |   },
9 |
10|   titulo: {
11|     fontSize: 14,
12|     marginVertical: 30,
13|     paddingHorizontal: 20,
14|     textAlign: "center"
15|   },
16|
17|   linhaInput: {
18|     flexDirection: 'row',
19|     alignItems: "center",
20|     marginBottom: 10
21|   },
22|
23|   inputFocus: {
24|     borderWidth: 1,
25|     borderColor: "#1f4f66",
26|     borderRadius: 3,
27|     width: 100,
28|     paddingHorizontal: 5,
29|     marginLeft: 10
30|   },
31|
32|   inputNormal: {
33|     borderWidth: 1,
34|     borderColor: '#d5d5d5',
35|     borderRadius: 3,
36|     width: 100,
37|     paddingHorizontal: 5,
38|     marginLeft: 10
39|   },
40|
41|   boxNumero:{ 
42|     borderColor: '#13b0c5',
43|     backgroundColor: '#13b0c5',
44|     borderWidth: 5,
45|     height: 150,
46|     width: 150,
47|     borderRadius: 75,
48|     justifyContent: "center",
49|     alignItems:"center",
50|     marginTop: 50,
51|     marginBottom: 50
52|   },
53|
54|   numero: {
55|     fontSize: 50,
56|     color: '#fff'
57|   },
58|   boxBotao: {
59|     width:200,
60|     marginTop: 20
61|   }
62| });
63|
64| export default estilo;
```

Código 1. Código do arquivo `estilo.js` que utiliza o componente `TelaInicial`.

# Entendendo o código

## componentes/TelaInicial/estilo.js

```
import { StyleSheet } from 'react-native';

const estilo = StyleSheet.create({
  tela: {
    width: '100%',
    justifyContent: "center",
    alignItems: 'center',
  },
  linhaInput: {
    flexDirection: 'row',
    alignItems: "center",
    marginBottom: 10
  },
})
```

Criamos um estilo para que os componentes **Text** e **TextInput** fiquem na mesma linha

Sorteio DevMedia  
Hora de ver quem é o vencedor

Escolha os valores mínimo e máximo para fazer o sorteio

Valor Mínimo:

Valor Máximo:

**SORTEAR**

0

```
22
23     inputFocus: {
24         borderWidth: 1,
25         borderColor: '#1f4f66',
26         borderRadius: 3,
27         width: 100,
28         paddingHorizontal: 5,
29         marginLeft: 10
30     },
31
32     inputNormal: {
33         borderWidth: 1,
34         borderColor: '#d5d5d5',
35         borderRadius: 3,
36         width: 100,
37         paddingHorizontal: 5,
38         marginLeft: 10
39     },
40
```

Criamos um estilo para quando o componente **TextInput** estiver em foco



```
22
23     inputFocus: {
24         borderWidth: 1,
25         borderColor: '#1f4f66',
26         borderRadius: 3,
27         width: 100,
28         paddingHorizontal: 5,
29         marginLeft: 10
30     },
31
32     inputNormal: {
33         borderWidth: 1,
34         borderColor: '#d5d5d5',
35         borderRadius: 3,
36         width: 100,
37         paddingHorizontal: 5,
38         marginLeft: 10
39     },
40
```

Criamos um estilo para quando o componente **TextInput** não estiver em foco



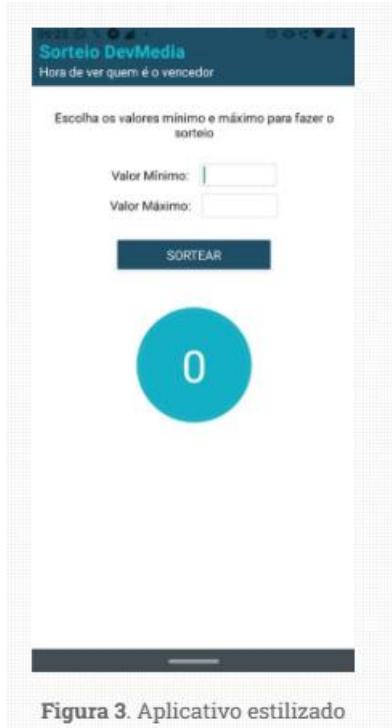


Figura 3. Aplicativo estilizado

O objetivo desse curso é focar no componente `TextInput`, por este motivo não detalhamos cada linha de código referente ao estilo do componente

Com nossa tela (componente `TelaInicial`) devidamente criada e estilizada vamos fazer alguns ajustes no componente `TextInput` inserindo as propriedades que aprendemos nas aulas anteriores.

As propriedades inseridas no componente `TextInput` podem ser vistas na **Figura 1**.

## 10. Personalizando o componente `TextInput`

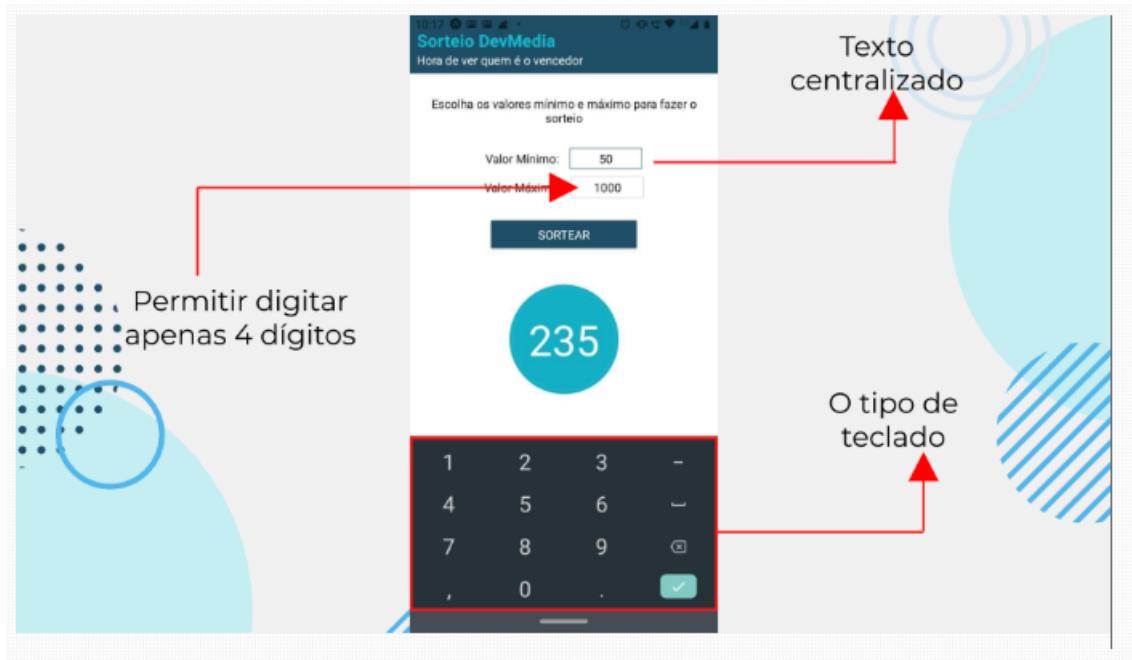


Figura 1. Propriedades do componente TextInput

```

1 import React, { useState } from 'react';
2 import { View, Text, Button, TextInput } from 'react-native';
3 import estilo from './estilo';
4
5 const TelaInicial = () => {
6   const [ numeroSorteado, setNumeroSorteado ] = useState(0);
7
8   const gerarNumero = () => {
9     const novoNumero = Math.floor(Math.random() * 100 + 1);
10    setNumeroSorteado(novoNumero);
11  }
12
13  return (
14    <View style = { estilo.tela }>
15      <Text style = { estilo.titulo }>
16        Escolha os valores mínimo e máximo para fazer o sorteio
17      </Text>
18
19      <View style = { estilo.linhaInput }>
20        <Text>Valor Mínimo: </Text>
21
22        <TextInput
23          style = { estilo.inputNormal }
24          textAlign = "center"
25          keyboardType = "number-pad"
26          maxLength = { 5 }
27          autoFocus = { true }/>
28      </View>
29
30      <View style = { estilo.linhaInput }>
31        <Text>Valor Máximo: </Text>
32
33        <TextInput
34          style = { estilo.inputNormal }
35          textAlign = "center"
36          keyboardType = "number-pad"
37          maxLength = { 5 }/>
38      </View>
39
40      <View style = { estilo.boxBotao }>
41        <Button title="Sortear" onPress = { gerarNumero } color="#1f4f66"/>
42      </View>
43
44      <View style = { estilo.boxNumero }>
45        <Text style = { estilo.numero }>{ numeroSorteado }</Text>
46      </View>
47    </View>
48  );
49}
50
51 export default TelaInicial;

```

**Código 1.** Código do componente Telainicial com as propriedades inseridas no componente TextInput

# Entendendo o código

## componentes/TelaInicial/estilo.js

```
import { StyleSheet } from 'react-native';

const estilo = StyleSheet.create({
  tela: {
    width: "100%",
    justifyContent: "center",
    alignItems: "center",
  },
  Criamos um estilo para que os componentes
  Text e TextInput fiquem na mesma linha
  linhaInput: {
    flexDirection: 'row',
    alignItems: "center",
    marginBottom: 10
  },
  7
  8
  9
  10
  11
  12
  13
  14
  15
  16
  17
  18
  19
  20
  21
  22
  23
  24
  25
  26
  27
  28
  29
  30
  31
  32
  33
  34
  35
  36
  37
  38
  39
  40
  41
  42
  43
  44
  45
  46
  47
  48
  49
  50
  51
  52
  53
  54
  55
  56
  57
  58
  59
  60
  61
  62
  63
  64
  65
  66
  67
  68
  69
  70
  71
  72
  73
  74
  75
  76
  77
  78
  79
  80
  81
  82
  83
  84
  85
  86
  87
  88
  89
  90
  91
  92
  93
  94
  95
  96
  97
  98
  99
  100
  101
  102
  103
  104
  105
  106
  107
  108
  109
  110
  111
  112
  113
  114
  115
  116
  117
  118
  119
  120
  121
  122
  123
  124
  125
  126
  127
  128
  129
  130
  131
  132
  133
  134
  135
  136
  137
  138
  139
  140
  141
  142
  143
  144
  145
  146
  147
  148
  149
  150
  151
  152
  153
  154
  155
  156
  157
  158
  159
  160
  161
  162
  163
  164
  165
  166
  167
  168
  169
  170
  171
  172
  173
  174
  175
  176
  177
  178
  179
  180
  181
  182
  183
  184
  185
  186
  187
  188
  189
  190
  191
  192
  193
  194
  195
  196
  197
  198
  199
  200
  201
  202
  203
  204
  205
  206
  207
  208
  209
  210
  211
  212
  213
  214
  215
  216
  217
  218
  219
  220
  221
  222
  223
  224
  225
  226
  227
  228
  229
  230
  231
  232
  233
  234
  235
  236
  237
  238
  239
  240
  241
  242
  243
  244
  245
  246
  247
  248
  249
  250
  251
  252
  253
  254
  255
  256
  257
  258
  259
  260
  261
  262
  263
  264
  265
  266
  267
  268
  269
  270
  271
  272
  273
  274
  275
  276
  277
  278
  279
  280
  281
  282
  283
  284
  285
  286
  287
  288
  289
  290
  291
  292
  293
  294
  295
  296
  297
  298
  299
  300
  301
  302
  303
  304
  305
  306
  307
  308
  309
  310
  311
  312
  313
  314
  315
  316
  317
  318
  319
  320
  321
  322
  323
  324
  325
  326
  327
  328
  329
  330
  331
  332
  333
  334
  335
  336
  337
  338
  339
  340
  341
  342
  343
  344
  345
  346
  347
  348
  349
  350
  351
  352
  353
  354
  355
  356
  357
  358
  359
  360
  361
  362
  363
  364
  365
  366
  367
  368
  369
  370
  371
  372
  373
  374
  375
  376
  377
  378
  379
  380
  381
  382
  383
  384
  385
  386
  387
  388
  389
  390
  391
  392
  393
  394
  395
  396
  397
  398
  399
  400
  401
  402
  403
  404
  405
  406
  407
  408
  409
  410
  411
  412
  413
  414
  415
  416
  417
  418
  419
  420
  421
  422
  423
  424
  425
  426
  427
  428
  429
  430
  431
  432
  433
  434
  435
  436
  437
  438
  439
  440
  441
  442
  443
  444
  445
  446
  447
  448
  449
  450
  451
  452
  453
  454
  455
  456
  457
  458
  459
  460
  461
  462
  463
  464
  465
  466
  467
  468
  469
  470
  471
  472
  473
  474
  475
  476
  477
  478
  479
  480
  481
  482
  483
  484
  485
  486
  487
  488
  489
  490
  491
  492
  493
  494
  495
  496
  497
  498
  499
  500
  501
  502
  503
  504
  505
  506
  507
  508
  509
  510
  511
  512
  513
  514
  515
  516
  517
  518
  519
  520
  521
  522
  523
  524
  525
  526
  527
  528
  529
  530
  531
  532
  533
  534
  535
  536
  537
  538
  539
  540
  541
  542
  543
  544
  545
  546
  547
  548
  549
  550
  551
  552
  553
  554
  555
  556
  557
  558
  559
  559
  560
  561
  562
  563
  564
  565
  566
  567
  568
  569
  569
  570
  571
  572
  573
  574
  575
  576
  577
  578
  579
  579
  580
  581
  582
  583
  584
  585
  586
  587
  588
  589
  589
  590
  591
  592
  593
  594
  595
  596
  597
  598
  599
  599
  600
  601
  602
  603
  604
  605
  606
  607
  608
  609
  609
  610
  611
  612
  613
  614
  615
  616
  617
  618
  619
  619
  620
  621
  622
  623
  624
  625
  626
  627
  628
  629
  629
  630
  631
  632
  633
  634
  635
  636
  637
  638
  639
  639
  640
  641
  642
  643
  644
  645
  646
  647
  648
  649
  649
  650
  651
  652
  653
  654
  655
  656
  657
  658
  659
  659
  660
  661
  662
  663
  664
  665
  666
  667
  668
  669
  669
  670
  671
  672
  673
  674
  675
  676
  677
  678
  679
  679
  680
  681
  682
  683
  684
  685
  686
  687
  688
  689
  689
  690
  691
  692
  693
  694
  695
  696
  697
  698
  699
  699
  700
  701
  702
  703
  704
  705
  706
  707
  708
  709
  709
  710
  711
  712
  713
  714
  715
  716
  717
  718
  719
  719
  720
  721
  722
  723
  724
  725
  726
  727
  728
  729
  729
  730
  731
  732
  733
  734
  735
  736
  737
  738
  739
  739
  740
  741
  742
  743
  744
  745
  746
  747
  748
  749
  749
  750
  751
  752
  753
  754
  755
  756
  757
  758
  759
  759
  760
  761
  762
  763
  764
  765
  766
  767
  768
  769
  769
  770
  771
  772
  773
  774
  775
  776
  777
  778
  779
  779
  780
  781
  782
  783
  784
  785
  786
  787
  788
  789
  789
  790
  791
  792
  793
  794
  795
  796
  797
  798
  799
  799
  800
  801
  802
  803
  804
  805
  806
  807
  808
  809
  809
  810
  811
  812
  813
  814
  815
  816
  817
  818
  819
  819
  820
  821
  822
  823
  824
  825
  826
  827
  828
  829
  829
  830
  831
  832
  833
  834
  835
  836
  837
  838
  839
  839
  840
  841
  842
  843
  844
  845
  846
  847
  848
  849
  849
  850
  851
  852
  853
  854
  855
  856
  857
  858
  859
  859
  860
  861
  862
  863
  864
  865
  866
  867
  868
  869
  869
  870
  871
  872
  873
  874
  875
  876
  877
  878
  879
  879
  880
  881
  882
  883
  884
  885
  886
  887
  888
  889
  889
  890
  891
  892
  893
  894
  895
  896
  897
  898
  899
  899
  900
  901
  902
  903
  904
  905
  906
  907
  908
  909
  909
  910
  911
  912
  913
  914
  915
  916
  917
  918
  919
  919
  920
  921
  922
  923
  924
  925
  926
  927
  928
  929
  929
  930
  931
  932
  933
  934
  935
  936
  937
  938
  939
  939
  940
  941
  942
  943
  944
  945
  946
  947
  948
  949
  949
  950
  951
  952
  953
  954
  955
  956
  957
  958
  959
  959
  960
  961
  962
  963
  964
  965
  966
  967
  968
  969
  969
  970
  971
  972
  973
  974
  975
  976
  977
  978
  979
  979
  980
  981
  982
  983
  984
  985
  986
  987
  988
  989
  989
  990
  991
  992
  993
  994
  995
  996
  997
  998
  999
  999
  1000
  1001
  1002
  1003
  1004
  1005
  1006
  1007
  1008
  1009
  1009
  1010
  1011
  1012
  1013
  1014
  1015
  1016
  1017
  1018
  1019
  1019
  1020
  1021
  1022
  1023
  1024
  1025
  1026
  1027
  1028
  1029
  1029
  1030
  1031
  1032
  1033
  1034
  1035
  1036
  1037
  1038
  1039
  1039
  1040
  1041
  1042
  1043
  1044
  1045
  1046
  1047
  1048
  1048
  1049
  1050
  1051
  1052
  1053
  1054
  1055
  1056
  1057
  1058
  1059
  1059
  1060
  1061
  1062
  1063
  1064
  1065
  1066
  1067
  1068
  1069
  1069
  1070
  1071
  1072
  1073
  1074
  1075
  1076
  1077
  1078
  1079
  1079
  1080
  1081
  1082
  1083
  1084
  1085
  1086
  1087
  1088
  1088
  1089
  1090
  1091
  1092
  1093
  1094
  1095
  1096
  1097
  1098
  1098
  1099
  1099
  1100
  1101
  1102
  1103
  1104
  1105
  1106
  1107
  1108
  1109
  1109
  1110
  1111
  1112
  1113
  1114
  1115
  1116
  1117
  1118
  1119
  1119
  1120
  1121
  1122
  1123
  1124
  1125
  1126
  1127
  1128
  1129
  1129
  1130
  1131
  1132
  1133
  1134
  1135
  1136
  1137
  1138
  1139
  1139
  1140
  1141
  1142
  1143
  1144
  1145
  1146
  1147
  1148
  1148
  1149
  1150
  1151
  1152
  1153
  1154
  1155
  1156
  1157
  1158
  1159
  1159
  1160
  1161
  1162
  1163
  1164
  1165
  1166
  1167
  1168
  1169
  1169
  1170
  1171
  1172
  1173
  1174
  1175
  1176
  1177
  1178
  1178
  1179
  1180
  1181
  1182
  1183
  1184
  1185
  1186
  1187
  1188
  1188
  1189
  1190
  1191
  1192
  1193
  1194
  1195
  1196
  1197
  1198
  1198
  1199
  1199
  1200
  1201
  1202
  1203
  1204
  1205
  1206
  1207
  1208
  1209
  1209
  1210
  1211
  1212
  1213
  1214
  1215
  1216
  1217
  1218
  1218
  1219
  1220
  1221
  1222
  1223
  1224
  1225
  1226
  1227
  1228
  1229
  1229
  1230
  1231
  1232
  1233
  1234
  1235
  1236
  1237
  1238
  1239
  1239
  1240
  1241
  1242
  1243
  1244
  1245
  1246
  1247
  1248
  1248
  1249
  1250
  1251
  1252
  1253
  1254
  1255
  1256
  1257
  1258
  1259
  1259
  1260
  1261
  1262
  1263
  1264
  1265
  1266
  1267
  1268
  1269
  1269
  1270
  1271
  1272
  1273
  1274
  1275
  1276
  1277
  1278
  1278
  1279
  1280
  1281
  1282
  1283
  1284
  1285
  1286
  1287
  1288
  1288
  1289
  1290
  1291
  1292
  1293
  1294
  1295
  1296
  1297
  1297
  1298
  1299
  1299
  1300
  1301
  1302
  1303
  1304
  1305
  1306
  1307
  1308
  1309
  1309
  1310
  1311
  1312
  1313
  1314
  1315
  1316
  1317
  1318
  1318
  1319
  1320
  1321
  1322
  1323
  1324
  1325
  1326
  1327
  1328
  1329
  1329
  1330
  1331
  1332
  1333
  1334
  1335
  1336
  1337
  1338
  1338
  1339
  1340
  1341
  1342
  1343
  1344
  1345
  1346
  1347
  1348
  1348
  1349
  1350
  1351
  1352
  1353
  1354
  1355
  1356
  1357
  1358
  1359
  1359
  1360
  1361
  1362
  1363
  1364
  1365
  1366
  1367
  1368
  1368
  1369
  1370
  1371
  1372
  1373
  1374
  1375
  1376
  1377
  1378
  1378
  1379
  1380
  1381
  1382
  1383
  1384
  1385
  1386
  1387
  1388
  1388
  1389
  1390
  1391
  1392
  1393
  1394
  1395
  1396
  1397
  1398
  1398
  1399
  1400
  1401
  1402
  1403
  1404
  1405
  1406
  1407
  1408
  1409
  1409
  1410
  1411
  1412
  1413
  1414
  1415
  1416
  1417
  1418
  1418
  1419
  1420
  1421
  1422
  1423
  1424
  1425
  1426
  1427
  1428
  1428
  1429
  1430
  1431
  1432
  1433
  1434
  1435
  1436
  1437
  1438
  1438
  1439
  1440
  1441
  1442
  1443
  1444
  1445
  1446
  1447
  1448
  1448
  1449
  1450
  1451
  1452
  1453
  1454
  1455
  1456
  1457
  1458
  1458
  1459
  1460
  1461
  1462
  1463
  1464
  1465
  1466
  1467
  1468
  1468
  1469
  1470
  1471
  1472
  1473
  1474
  1475
  1476
  1477
  1477
  1478
  1479
  1480
  1481
  1482
  1483
  1484
  1485
  1486
  1487
  1488
  1488
  1489
  1490
  1491
  1492
  1493
  1494
  1495
  1496
  1497
  1498
  1498
  1499
  1500
  1501
  1502
  1503
  1504
  1505
  1506
  1507
  1508
  1509
  1509
  1510
  1511
  1512
  1513
  1514
  1515
  1516
  1517
  1518
  1518
  1519
  1520
  1521
  1522
  1523
  1524
  1525
  1526
  1527
  1528
  1528
  1529
  1530
  1531
  1532
  1533
  1534
  1535
  1536
  1537
  1538
  1538
  1539
  1540
  1541
  1542
  1543
  1544
  1545
  1546
  1547
  1548
  1548
  1549
  1550
  1551
  1552
  1553
  1554
  1555
  1556
  1557
  1558
  1559
  1559
  1560
  1561
  1562
  1563
  1564
  1565
  1566
  1567
  1568
  1568
  1569
  1570
  1571
  1572
  1573
  1574
  1575
  1576
  1577
  1578
  1578
  1579
  1580
  1581
  1582
  1583
  1584
  1585
  1586
  1587
  1588
  1588
  1589
  1590
  1591
  1592
  1593
  1594
  1595
  1596
  1597
  1598
  1598
  1599
  1600
  1601
  1602
  1603
  1604
  1605
  1606
  1607
  1608
  1609
  1609
  1610
  1611
  1612
  1613
  1614
  1615
  1616
  1617
  1618
  1618
  1619
  1620
  1621
  1622
  1623
  1624
  1625
  1626
  1627
  1628
  1628
  1629
  1
```

```
3 InputFocus: {  
4   Criamos um estilo para quando o  
5   componente TextInput não estiver em foco  
6   width: 100,  
7   paddingHorizontal: 5,  
8   marginLeft: 10  
9 },  
10  
11 inputNormal: {  
12   borderWidth: 1,  
13   borderColor: '#d5d5d5',  
14   borderRadius: 3,  
15   width: 100,  
16   paddingHorizontal: 5,  
17   marginLeft: 10  
18 },  
19  
20 }
```

## 11. Definindo os eventos onFocus e onBlur

O último passo para que a aparência do nosso aplicativo seja concluída é mudar o estilo do componente **TextInput** quando ele estiver em foco.



```
import React, { useState } from 'react';
import { View, Text, Button, TextInput } from 'react-native';
import estilos from './estilos';

const TelaInicial = () => {
  const [ numeroSorteado, setNumeroSorteado ] = useState(0);

  const [ flagInputFocus, setFlagInputFocus ] = useState("");

  const gerarNumero = () => {
    Criamos uma variável de estado onde vamos armazenar
    seNúmeroSorteado é o número que
    qual componente está em foco

    return (
      
        
          Escolha os valores mínimo e máximo para fazer o sorteio
        
        

```

```
</View>
<View style={ estilos.tela }>
  <Text style={ estilos.titulo }>
    Escolha os valores mínimo e máximo para fazer o sorteio
  </Text>
  A propriedade style vai receber o estilo baseado no valor
  da variável flagInputFocus.
  <View style={ estilos.linhaInput }>
    <Text>Valor Mínimo:</Text>
    <TextInput
      textAlign="center"
      keyboardType="number-pad"
      maxLength={5}
      autoFocus={true}
      style={ flagInputFocus === "txt_min" ? estilos.inputFocus : estilos.inputNormal }
      onFocus={ () => setFlagInputFocus("txt_min") }
      onBlur={ () => setFlagInputFocus("") } />
  </View>
  Este valor será verificado através de um if ternário
  <View style={ estilos.linhaInput }>
    <Text>Valor Máximo:</Text>

```

```
</View>
<View style={ estilos.tela }>
  <Text style={ estilos.titulo }>
    Escolha os valores mínimo e máximo para fazer o sorteio
  </Text>

  <View style={ estilos.linhaInput }>
    Se o valor da variável flagInputFocus for igual a "txt_min"
    o estilo recebido será estilos.inputFocus
    <TextInput
      textAlign="center"
      keyboardType="number-pad"
      maxLength={5}
      autoFocus={true}
      style={ flagInputFocus === "txt_min" ? estilos.inputFocus : estilos.inputNormal }
      onFocus={ () => setFlagInputFocus("txt_min") }
      onBlur={ () => setFlagInputFocus("") } />
  </View>
  Senão o estilo será estilos.inputNormal
  <View style={ estilos.linhaInput }>
    <Text>Valor Máximo:</Text>

```

```
</View>
<View style={ estilos.tela }>
  <Text style={ estilos.titulo }>
    Escolha os valores mínimo e máximo para fazer o sorteio
  </Text>

  <View style={ estilos.linhaInput }>
    Definimos a função que será chamada quando o
    componente estiver em foco
    <TextInput
      textAlign="center"
      keyboardType="number-pad"
      maxLength={5}
      autoFocus={true}
      style={ flagInputFocus === "txt_min" ? estilos.inputFocus : estilos.inputNormal }
      onFocus={ () => setFlagInputFocus("txt_min") }
      onBlur={ () => setFlagInputFocus("") } />
  </View>

  <View style={ estilos.linhaInput }>
    <Text>Valor Mínimo:</Text>

```

```
1 <View style = { estilo.tela }>
2   <text style = { estilo.titulo }>
3     Escolha os valores mínimo e máximo para fazer o sorteio
4   </Text>
5
6   <View style = { estilo.linhaInput }>
7     Note que quando o componente estiver em foco, o valor
8     da variável flagInputFocus será alterado para "txt_min"
9     <TextInput
10       textAlign="center"
11       keyboardType="number-pad"
12       maxLength=5
13       autoFocus={true}
14       style = { flagInputFocus === "txt_min" ? estilo.inputFocus : estilo.inputNormal }
15       onFocus={ () => setFlagInputFocus("txt_min") }
16       onBlur={ () => setFlagInputFocus("") } />
17   </View>
18
19   <View style = { estilo.linhaInput }>
20     <Text>Valor Máximo</Text>
21
22 </View>
```

```
1 <View style = { estilo.tela }>
2   <text style = { estilo.titulo }>
3     Escolha os valores mínimo e máximo para fazer o sorteio
4   </Text>
5
6   <View style = { estilo.linhaInput }>
7     Definimos a função que será chamada quando o componente sair
8       de foco
9     <TextInput
10       textAlign="center"
11       keyboardType="number-pad"
12       maxLength=5
13       autoFocus={true}
14       style = { flagInputFocus === "txt_min" ? estilo.inputFocus : estilo.inputNormal }
15       onFocus={ () => setFlagInputFocus("txt_min") }
16       onBlur = { () => setFlagInputFocus("") } />
17   </View>
18
19   <View style = { estilo.linhaInput }>
20     <Text>Valor Mínimo</Text>
21
22 </View>
```

```
1 <View style = { estilo.tela }>
2   <text style = { estilo.titulo }>
3     Escolha os valores mínimo e máximo para fazer o sorteio
4   </Text>
5
6   <View style = { estilo.linhaInput }>
7     Quando o componente sair de foco, o valor da variável
8       flagInputFocus será alterado para "" (vazio)
9     <TextInput
10       textAlign="center"
11       keyboardType="number-pad"
12       maxLength=5
13       autoFocus={true}
14       style = { flagInputFocus === "txt_min" ? estilo.inputFocus : estilo.inputNormal }
15       onFocus={ () => setFlagInputFocus("txt_min") }
16       onBlur = { () => setFlagInputFocus("") } />
17   </View>
18
19   <View style = { estilo.linhaInput }>
20     <Text>Valor Máximo</Text>
21
22 </View>
```

O mesmo foi feito para o  
TextInput que receberá o valor  
máximo

```

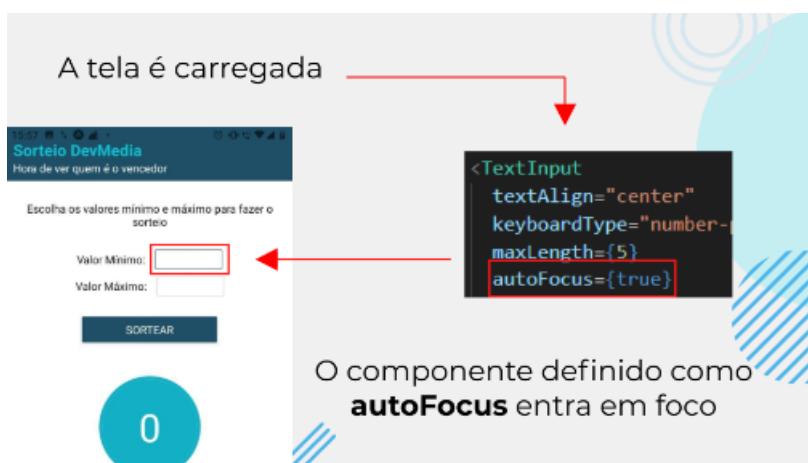
44     View style={ estilo.linhaInput }
45     textValue={Maximo} //Text
46
47     <input
48       textAlign="center"
49       keyboardType="number-pad"
50       maxLength={5}
51     style={ flagInputFocus === "txt_max" ? estilo.inputFocus : estilo.inputNormal }
52     onFocus={()>setFlagInputFocus("txt_max")}
53     onBlur={()>setFlagInputFocus("")}/>
54   </View>
55
56   <View style={ estilo.boxBotao }>
57     <button style={ estilo.botao } onClick={()>gerarNumero()} color="#747678">
58       Gerar
59     </button>
60   </View>
61
62 </View>
63
64 ]
65

```

A única diferença aqui é que o valor a ser comparado é setado será "txt\_max"

Vamos resumir o que vai acontecer com essas alterações feitas.

## onFocus



O evento **onFocus** é disparado

```
<TextInput  
    textAlign="center"  
    keyboardType="number-pad"  
    maxLength={5}  
    autoFocus={true}  
    style = { flagInputFocus === "txt_min" ? estilos.inputFocus : estilos.inputNormal }  
    onFocus={ () => setFlagInputFocus("txt_min") }  
    onBlur = { () => setFlagInputFocus("") }/>
```

O valor da variável **flagInputFocus**  
é alterado para **txt\_min**

**flagInputFocus** agora possui o valor **txt\_min**  
com isso a condição é verdadeira

```
<TextInput  
    textAlign="center"  
    keyboardType="number-pad"  
    maxLength={5}  
    autoFocus={true}  
    style = { flagInputFocus === "txt_min" ? estilos.inputFocus : estilos.inputNormal }  
    onFocus={ () => setFlagInputFocus("txt_min") }  
    onBlur = { () => setFlagInputFocus("") }/>
```

O estilo atribuído ao componente será  
**estilo.inputFocus**



O resultado desse  
processo é o **TextInput**  
ficar com a borda azul

## onBlur comportamento atual

DevMedia

er quem é o vencedor

i os valores mínimo e máximo para f.  
sorteio

Valor Mínimo:

Valor Máximo:

SORTEAR

O usuário toca no  
segundo **TextInput**

DevMedia

er quem é o vencedor

i os valores mínimo e máximo para f.  
sorteio

Valor Mínimo:

Valor Máximo:

SORTEAR

O primeiro **TextInput** sai  
de foco

O evento **onBlur** (do componente que saiu de foco) é disparado

```
<TextInput  
    textAlign="center"  
    keyboardType="number-pad"  
    maxLength={5}  
    autoFocus={true}  
    style = { flagInputFocus === "txt_min" ? estilo.inputFocus : estilo.inputNormal }  
    onFocus={ () => setFlagInputFocus("txt_min") }  
    onBlur = { () => setFlagInputFocus("") }/>
```

O valor da variável **flagInputFocus** é alterado para "" (vazio)

**flagInputFocus** não é igual a **txt\_min** e com isso a condição é falsa

```
<TextInput  
    textAlign="center"  
    keyboardType="number-pad"  
    maxLength={5}  
    autoFocus={true}  
    style = { flagInputFocus === "txt_min" ? estilo.inputFocus : estilo.inputNormal }  
    onFocus={ () => setFlagInputFocus("txt_min") }  
    onBlur = { () => setFlagInputFocus("") }/>
```

O estilo atribuído ao componente será **estilo.inputNormal**



O resultado desse processo é o **TextInput** não ter a borda azul

Finalizamos toda a parte visual do nosso aplicativo. O último passo é programar para que ele sorteie o número baseado nos valores digitados pelo usuário.

# 12. Definindo o evento onchangeText

Já codificamos toda parte visual do nosso aplicativo. O que precisamos agora é armazenar o que foi digitado pelo usuário para sortearmos o número, como pode ser visto na



```
import { View, Text, Button, TextInput } from 'react-native';
import estilos from './estilos';

Criamos duas variáveis de estado.

const TelaInicial = () => {
  const [ numeroSorteado, setNumeroSorteado ] = useState(0);
  const [ valorMinimo, setValorMinimo ] = useState(0);
  const [ valorMaximo, setValorMaximo ] = useState(10);

  const [ flagInputFocus, setFlagInputFocus ] = useState("");

  Uma vai armazenar o valor mínimo (inicializada com o valor
  0) e a outra o valor máximo (inicializada com o valor 10).
  setNumeroSorteado(novoNúmero);
}

return (
  <View style={ estilos.tela }>
    <Text style={ estilos.título }>
      Escolha os valores mínimo e máximo para fazer o sorteio
    </Text>

    <View style={ estilos.linhaInput }>
      <Text>Valor Mínimo:</Text>
      Passamos para a propriedade value o valor da variável valorMinimo.
      Dessa forma ao iniciar o aplicativo o valor desse TextInput iniciará com o
      valor da variável valorMinimo, ou seja, 0.
      <TextInput
        keyboardType="number"
        maxLength={ 5 }
        autoFocus={ true }
        style={ flagInputFocus === "txt_min" ? estilos.inputFocus : estilos.input }
        onFocus={ () => setFlagInputFocus("txt_min") }
        onBlur={ () => setFlagInputFocus("") }
        value={ valorMinimo.toString() }
        onChangeText={ ( valor ) => setValorMinimo(valor) }
      />
    </View>
  </View>
)
```

```

    <View style={estilo.tela}>
      <Text style={estilo.titulo}>
        Escolha os valores mínimo e máximo para fazer o sorteio
      </Text>

      <View style={estilo.linhaInput}>
        <Text>Valor Mínimo:</Text>
        Por se tratar de um campo de texto o valor da propriedade value deve
        ser uma string. Ao utilizar a função toString convertemos o valor da
        variável para uma string.
        <Text>Valor Mínimo:</Text>
        <TextInput
          textAlign="center"
          keyboardType="number-pad"
          maxLength={5}
          autoFocus={true}
          style={flagInputFocus === "txt_min" ? estilo.inputFocus : estilo.input}
          onFocus={() => setFlagInputFocus("txt_min")}
          onBlur={() => setFlagInputFocus("")}
          value={valorMinimo.toString()} // Aqui
          onChangeText={valor => setValorMinimo(valor)} // Aqui
        </TextInput>
      </View>
    </View>
  
```

```

    <View style={estilo.tela}>
      <Text style={estilo.titulo}>
        Escolha os valores mínimo e máximo para fazer o sorteio
      </Text>

      <View style={estilo.linhaInput}>
        <Text>Valor Mínimo:</Text>
        Definimos a função que será chamada quando o texto
        do TextInput mudar.
        <Text>Valor Mínimo:</Text>
        <TextInput
          textAlign="center"
          keyboardType="number-pad"
          maxLength={5}
          autoFocus={true}
          style={flagInputFocus === "txt_min" ? estilo.inputFocus : estilo.input}
          onFocus={() => setFlagInputFocus("txt_min")}
          onBlur={() => setFlagInputFocus("")}
          value={valorMinimo.toString()} // Aqui
          onChangeText={valor => setValorMinimo(valor)} // Aqui
        </TextInput>
      </View>
    </View>
  
```

```

    <View style={estilo.tela}>
      <Text style={estilo.titulo}>
        Escolha os valores mínimo e máximo para fazer o sorteio
      </Text>

      <View style={estilo.linhaInput}>
        <Text>Valor Mínimo:</Text>
        Alteramos o valor da variável ValorMinimo utilizando o parâmetro
        recebido (valor).
        <Text>Valor Mínimo:</Text>
        <TextInput
          textAlign="center"
          keyboardType="number-pad"
          maxLength={5}
          autoFocus={true}
          style={flagInputFocus === "txt_min" ? estilo.inputFocus : estilo.input}
          onFocus={() => setFlagInputFocus("txt_min")}
          onBlur={() => setFlagInputFocus("")}
          value={valorMinimo.toString()} // Aqui
          onChangeText={valor => setValorMinimo(valor)} // Aqui
        </TextInput>
      </View>
    </View>
  
```

```

    <View style={estilo.tela}>
      <Text style={estilo.titulo}>
        Escolha os valores mínimo e máximo para fazer o sorteio
      </Text>

      <View style={estilo.linhaInput}>
        <Text>Valor Mínimo:</Text>
        Lembrando que este parâmetro é o novo valor do TextInput.
        <Text>Valor Mínimo:</Text>
        <TextInput
          textAlign="center"
          keyboardType="number-pad"
          maxLength={5}
          autoFocus={true}
          style={flagInputFocus === "txt_min" ? estilo.inputFocus : estilo.input}
          onFocus={() => setFlagInputFocus("txt_min")}
          onBlur={() => setFlagInputFocus("")}
          value={valorMinimo.toString()} // Aqui
          onChangeText={valor => setValorMinimo(valor)} // Aqui
        </TextInput>
      </View>
    </View>
  
```

Entenda o que acontece quando o usuário digitar um valor



O evento **onChangeText** é disparado

```
<TextInput  
    textAlign = "center"  
    keyboardType = "number-pad"  
    maxLength = { 5 }  
    autoFocus = { true }  
    style = { flagInputFocus === "txt_min" ? estilo.inputF  
    onFocus={ () => setFlagInputFocus("txt_min") }  
    onBlur = { () => setFlagInputFocus("") }  
    value = { valorMinimo.toString() }  
    onChangeText = { valor => setValorMinimo(valor) } />
```

O evento recebe o novo valor e passa como parâmetro para a função atrelada ao evento

```
<TextInput  
    textAlign = "center"  
    keyboardType = "number-pad"  
    maxLength = { 5 }  
    autoFocus = { true }  
    style = { flagInputFocus === "txt_min" ? estilo.inputF  
    onFocus={ () => setFlagInputFocus("txt_min") }  
    onBlur = { () => setFlagInputFocus("") }  
    value = { valorMinimo.toString() }  
    onChangeText = { valor => setValorMinimo(valor) } />
```

O valor da variável **valorMinimo** é alterado para o novo valor que foi recebido

O novo valor da variável **valorMinimo** é exibido no componente **TextInput**

```
<TextInput  
    textAlign = "center"  
    keyboardType = "number-pad"  
    maxLength = { 5 }  
    autoFocus = { true }  
    style = { flagInputFocus === "txt_min" ? estilo.inputF  
    onFocus={ () => setFlagInputFocus("txt_min") }  
    onBlur = { () => setFlagInputFocus("") }  
    value = { valorMinimo.toString() }  
    onChangeText = { valor => setValorMinimo(valor) } />
```

## 13. Sorteando um número

Agora só precisamos de uma última alteração para que nosso aplicativo funcione como desejado, ou seja, sortear um número baseado no que o usuário digitou, como visto na

Para isso vamos alterar mais uma vez o arquivo do componente `TelaInicial`, conforme mostra o [Código 1](#).

```
1 import React, { useState } from 'react';
2 import { View, Text, Button, TextInput } from 'react-native';
3 import estilos from './estilos';
4
5 const TelaInicial = () => {
6   const [ numeroSorteado, setNumeroSorteado ] = useState(0);
7   const [ valorMinimo, setValorMinimo ] = useState(0);
8   const [ valorMaximo, setValorMaximo ] = useState(10);
9
10  const [ flagInputFocus, setFlagInputFocus ] = useState("");
11
12  const validarCampos = (minimo, maximo) => {
13    if ( isNaN(minimo) || isNaN(maximo) ) {
14      alert('Digite os valores');
15      return false;
16    }
17
18    if ( minimo > maximo ) {
19      alert('O valor mínimo deve ser menor que o valor máximo');
20      return false;
21    }
22
23    return true;
24  }
25
26  const gerarNumero = () => {
27    const min = parseInt(valorMinimo);
28    const max = parseInt(valorMaximo);
29
30    if ( !validarCampos(min, max) ) {
31      return;
32    }
33  }
```

```

33
34     const novoNumero = Math.floor(Math.random() * (max + 1 - min) + min);
35     setNumeroSorteado(novoNumero);
36 }
37
38 return (
39     <View style = { estilo.tela }>
40         <Text style = { estilo.titulo }>
41             Escolha os valores mínimo e máximo para fazer o sorteio
42         </Text>
43
44     <View style = { estilo.linhaInput }>
45         <Text>Valor Mínimo: </Text>
46
47         <TextInput
48             textAlign = "center"
49             keyboardType = "number-pad"
50             maxLength = { 5 }
51             autoFocus = { true }
52             style = { flagInputFocus === "txt_min" ?
53                 estilo.inputFocus : estilo.inputNormal }
54             onFocus={ () => setFlagInputFocus("txt_min") }
55             onBlur = { () => setFlagInputFocus("") }
56             value = { valorMinimo.toString() }
57             onChangeText = { valor => setValorMinimo(valor) } />
58     </View>
59
60     <View style = { estilo.linhaInput }>
61         <Text>Valor Máximo: </Text>
62
63         <TextInput
64             textAlign="center"
65             keyboardType="number-pad"
66             maxLength = { 5 }
67             style = { flagInputFocus === "txt_max" ?
68                 estilo.inputFocus : estilo.inputNormal }
69             onFocus = { ()=>setFlagInputFocus("txt_max") }
70             onBlur = { ()=>setFlagInputFocus("") }
71             value = { valorMaximo.toString() }
72             onChangeText = { setValorMaximo } />
73     </View>
74
75     <View style = { estilo.boxBotao }>
76         <Button title="Sortear" onPress = { gerarNumero } color="#1f4f66"/>
77     </View>
78
79     <View style = { estilo.boxNumero }>
80         <Text style = { estilo.numero }>{ numeroSorteado }</Text>
81     </View>
82     </View>
83 );
84 }
85
86 export default TelaInicial;

```

# Gerando um número

Baseado nos valores digitados pelo usuário

```
12 const validarCampos = (minimo, maximo) => {
13   if ( isNaN(minimo) || isNaN(maximo)) {
14     alert('Digite os valores');
15     return false;
16   }
17
18   if ( minimo > maximo ) {
19     alert('O valor mínimo deve ser menor que o valor máximo');
20     return false;
21   }
22
23   return true;
24 }
```

Criamos uma função para validar os valores digitados pelo usuário

```
26 const min = parseInt(valorMinimo);
```

```
12 const validarCampos = (minimo, maximo) => {
13   if ( isNaN(minimo) || isNaN(maximo)) {
14     alert('Digite os valores');
15     return false;
16   }
17
18   if ( minimo > maximo ) {
19     Caso o valor não tenha sido digitado vai retornar false.
20     alert('O valor mínimo deve ser menor que o valor máximo');
21     return false;
22   }
23   Isso acontece caso o usuário apague um dos valores deixando o
24   campo vazio.
25
26   return true;
27 }
```

```
26 const gerarNumero = () => {
27   const min = parseInt(valorMinimo);
28   const max = parseInt(valorMaximo);
29
30   const numero = Math.floor(Math.random() * (max - min + 1)) + min;
31
32   return numero;
33 }
```

```
12 const validarCampos = (minimo, maximo) => {
13     if ( isNaN(minimo) || isNaN(maximo)) {
14         alert('Digite os valores');
15         Se minimo for maior maximo também retornará false.
16     }
17
18     if ( minimo > maximo ) {
19         alert('O valor mínimo deve ser menor que o valor máximo');
20         return false;
21     }
22
23     return true;
24 }
25
26 const gerarNumero = () => {
27     const min = parseInt(valorMinimo);
28
```

```
25
26     const gerarNumero = () => [
27         const min = parseInt(valorMinimo);
28         const max = parseInt(valorMaximo);
29
30         if ( !validarCampos(min, max) ) {
31             return;
32         }
33
34         const novoNumero = Math.floor(Math.random() * (max + 1 - min) + min);
35         setNumeroSorteado(novoNumero);
36     ]
37
38     return (
39         <View style = { estilo.tela }>
40             <Text style = { estilo.titulo }>
41                 Criamos a função que vai gerar um número baseado nos valores
42                 Escolha os valores mínimo e máximo para fazer o sorteio
43                 digitados pelo usuário
44             </Text>
45     )
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1096
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1167
1168
1169
1170
1171
1172
1173
1174
1175
1175
1176
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1186
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1195
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1286
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1295
1296
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1367
1368
1369
1370
1371
1372
1373
1374
1375
1375
1376
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1386
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1395
1396
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1445
1446
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1475
1476
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1485
1486
1487
1488
1489
1489
1490
1491
1492
1493
1493
1494
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1506
1507
1508
1509
1510
1511
1512
1513
1514
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1535
1536
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1545
1546
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1575
1576
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1585
1586
1587
1588
1589
1589
1590
1591
1592
1593
1594
1594
1595
1596
1597
1598
1598
1599
1599
1600
1601
1602
1603
1604
1604
1605
1606
1607
1608
1608
1609
1610
1611
1612
1612
1613
1614
1615
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1623
1624
1625
1626
1627
1627
1628
1629
1630
1631
1632
1633
1633
1634
1635
1636
1637
1637
1638
1639
1640
1641
1642
1643
1643
1644
1645
1646
1646
1647
1648
1649
1649
1650
1651
1652
1653
1654
1654
1655
1656
1657
1658
1658
1659
1660
1661
1662
1663
1663
1664
1665
1666
1667
1667
1668
1669
1670
1671
1672
1672
1673
1674
1675
1676
1676
1677
1678
1679
1679
1680
1681
1682
1683
1684
1684
1685
1686
1687
1688
1688
1689
1690
1691
1692
1692
1693
1694
1695
1696
1696
1697
1698
1699
1699
1700
1701
1702
1703
1704
1704
1705
1706
1707
1708
1708
1709
1710
1711
1712
1712
1713
1714
1715
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1723
1724
1725
1726
1727
1727
1728
1729
1730
1731
1732
1733
1733
1734
1735
1736
1737
1737
1738
1739
1740
1741
1742
1742
1743
1744
1745
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1754
1755
1756
1757
1758
1758
1759
1760
1761
1762
1763
1763
1764
1765
1766
1767
1767
1768
1769
1770
1771
1772
1772
1773
1774
1775
1776
1776
1777
1778
1779
1779
1780
1781
1782
1783
1783
1784
1785
1786
1787
1787
1788
1789
1790
1791
1792
1792
1793
1794
1795
1796
1796
1797
1798
1799
1799
1800
1801
1802
1803
1803
1804
1805
1806
1807
1807
1808
1809
1810
1811
1812
1812
1813
1814
1815
1815
1816
1817
1818
1818
1819
1820
1821
1822
1822
1823
1824
1825
1826
1826
1827
1828
1829
1829
1830
1831
1832
1833
1833
1834
1835
1836
1837
1837
1838
1839
1840
1841
1842
1842
1843
1844
1845
1846
1846
1847
1848
1849
1849
1850
1851
1852
1853
1853
1854
1855
1856
1857
1857
1858
1859
1860
1861
1862
1862
1863
1864
1865
1866
1866
1867
1868
1869
1869
1870
1871
1872
1873
1873
1874
1875
1876
1877
1877
1878
1879
1880
1881
1882
1882
1883
1884
1885
1886
1886
1887
1888
1889
1889
1890
1891
1892
1893
1893
1894
1895
1896
1897
1897
1898
1899
1899
1900
1901
1902
1903
1903
1904
1905
1906
1907
1907
1908
1909
1910
1911
1912
1912
1913
1914
1915
1915
1916
1917
1918
1918
1919
1920
1921
1922
1922
1923
1924
1925
1926
1926
1927
1928
1929
1929
1930
1931
1932
1933
1933
1934
1935
1936
1936
1937
1938
1939
1939
1940
1941
1942
1943
1943
1944
1945
1946
1946
1947
1948
1949
1949
1950
1951
1952
1953
1953
1954
1955
1956
1957
1957
1958
1959
1960
1961
1962
1962
1963
1964
1965
1966
1966
1967
1968
1969
1969
1970
1971
1972
1973
1973
1974
1975
1976
1976
1977
1978
1979
1979
1980
1981
1982
1983
1983
1984
1985
1986
1987
1987
1988
1989
1989
1990
1991
1992
1993
1993
1994
1995
1996
1996
1997
1998
1999
1999
2000
2001
2002
2003
2003
2004
2005
2006
2006
2007
2008
2009
2009
2010
2011
2012
2012
2013
2014
2015
2015
2016
2017
2018
2018
2019
2020
2021
2022
2022
2023
2024
2025
2025
2026
2027
2028
2028
2029
2030
2031
2032
2032
2033
2034
2035
2035
2036
2037
2038
2038
2039
2040
2041
2042
2042
2043
2044
2045
2045
2046
2047
2048
2048
2049
2050
2051
2052
2052
2053
2054
2055
2056
2056
2057
2058
2059
2059
2060
2061
2062
2063
2063
2064
2065
2066
2066
2067
2068
2069
2069
2070
2071
2072
2073
2073
2074
2075
2076
2076
2077
2078
2079
2079
2080
2081
2082
2083
2083
2084
2085
2086
2086
2087
2088
2089
2089
2090
2091
2092
2093
2093
2094
2095
2096
2096
2097
2098
2099
2099
2100
2101
2102
2103
2103
2104
2105
2106
2106
2107
2108
2109
2109
2110
2111
2112
2112
2113
2114
2115
2115
2116
2117
2118
2118
2119
2120
2121
2122
2122
2123
2124
2125
2125
2126
2127
2128
2128
2129
2130
2131
2132
2132
2133
2134
2135
2135
2136
2137
2138
2138
2139
214
```

```
16 const gerarNumero = () => {
17     const min = parseInt(valorMinimo);
18     const max = parseInt(valorMaximo);
19
20     if ( !validarCampos(min, max) ) {
21         return (
22             <View style = { estilo.tela }>
23                 <Text style = { estilo.titulo }>
24                     Escolha os valores mínimo e máximo para fazer o sorteio
25                 </Text>
26         );
27     }
28
29     return (
30         <View style = { estilo.tela }>
31             <Text style = { estilo.titulo }>
32                 Escolha os valores mínimo e máximo para fazer o sorteio
33             </Text>
34     );
35 }
```

Os valores digitados pelo usuário no componente **TextInput** são do tipo **string** por isso convertemos para **number**

```
15
16 const gerarNumero = () => {
17     const min = parseInt(valorMinimo);
18     const max = parseInt(valorMaximo);
19
20     if ( !validarCampos(min, max) ) {
21         return;
22     }
23
24     const novoNumero = Math.floor(Math.random() * (max + 1 - min) + min);
25     setNumeroSorteado(novoNumero);
26
27 }
```

Validarmos os campos através da função **validarCampos** e se o valor retornado for **false** paramos a execução da função **gerarNumero**.

```
18
19     <Text style = { estilo.titulo }>
20         Escolha os valores mínimo e máximo para fazer o sorteio
21     </Text>
22
23 }
```

Geramos um número aleatório baseado no que o usuário digitou

```
15
16 const gerarNumero = () => {
17     const min = parseInt(valorMinimo);
18     const max = parseInt(valorMaximo);
19
20     Geramos um número aleatório baseado no que o usuário digitou
21
22     if ( !validarCampos(min, max) ) {
23         return;
24     }
25
26     const novoNumero = Math.floor(Math.random() * (max + 1 - min) + min);
27     setNumeroSorteado(novoNumero);
28
29
30     return (
31         <View style = { estilo.tela }>
32             <Text style = { estilo.titulo }>
33                 Escolha os valores mínimo e máximo para fazer o sorteio
34             </Text>
35     );
36
37 }
```

```
const min = parseInt(valorMinimo);
const max = parseInt(valorMaximo);

Alteramos o valor da variável numeroSorteado para o novo valor gerado.

if ( !validarCampos(min, max) ) {
    return;
}

const novoNumero = Math.floor(Math.random() * (max + 1 - min)) + min;
setNumeroSorteado(novoNumero);

}

return (
    <View style = { estilo.tela }>
        <Text style = { estilo.título }>
            Escolha os valores mínimo e máximo para fazer o sorteio
    </Text>
</View>
)
```

Ao terminar este curso, além dos conhecimentos dos cursos anteriores, você também é capaz de:

1. Receber valores digitados pelo usuário através do componente **TextInput** utilizando:
  - a. A propriedade **value**.
  - b. O evento **onChangedText**.
2. Personalizar o componente **TextInput** através das propriedades:
  - a. **textAlign**
  - b. **keyboardType**
  - c. **autoFocus**
  - d. **maxLength**
3. Executar uma ação quando o componente **TextInput** entrar ou sair de foco através dos eventos:
  - a. **onFocus**
  - b. **onblur**

## React Native: Sorteador personalizado de números

Introdução

App.js

components

TelaInicial/index.js  
TelaInicial/estilo.js  
Titulo/index.js  
Titulo/estilo.js

## React Native: Sorteador personalizado de números

Neste projeto criaremos um sorteador utilizando dois inputs recebidos pelo usuário para sortear um número aleatório. Para receber estes inputs do usuário vamos utilizar o componente TextInput.

App.js

```
1 import React from 'react';
2 import { View } from 'react-native';
3 import Titulo from './componentes/Titulo';
4 import TelaInicial from './componentes/TelaInicial';
5
6 export default function App() {
7   return (
8     <View>
9       <Titulo />
10      <TelaInicial />
11    </View>
12  );
13}
```

## personalizado de números

Introdução

App.js

components

TelaInicial/index.js  
TelaInicial/estilo.js  
Titulo/index.js  
Titulo/estilo.js

## React Native: Sorteador personalizado de números

Neste projeto criaremos um sorteador utilizando dois inputs recebidos pelo usuário para sortear um número aleatório. Para receber estes inputs do usuário vamos utilizar o componente TextInput.

TelaInicial/index.js

```
1 import React, { useState } from 'react';
2 import { View, Text, Button, TextInput } from 'react-native';
3 import estilo from './estilo';
4
5 const TelaInicial = () => {
6   const [ numeroSorteado, setNumeroSorteado ] = useState(0);
7   const [ valorMinimo, setValorMinimo ] = useState(0);
8   const [ valorMaximo, setValorMaximo ] = useState(10);
9
10  const [ flagInputFocus, setFlagInputFocus ] = useState("");
11
12  const validarCampos = (minimo, maximo) => {
13    if ( isNaN(minimo) || isNaN(maximo)) {
14      alert('Digite os valores');
15      return false;
16    }
17
18    if ( minimo > maximo ) {
19      alert('O valor mínimo deve ser menor que o valor máximo');
20      return false;
21    }
22
23    return true;
24  }
25
26  const gerarNumero = () => {
27    const min = parseInt(valorMinimo);
28    const max = parseInt(valorMaximo);
29
30    if ( !validarCampos(min, max) ) {
31      return;
32    }
33}
```

```

33     const novoNumero = Math.floor(Math.random() * (max + 1 - min) + min);
34     setNumeroSorteado(novoNumero);
35   }
36
37   return (
38     <View style = { estilo.tela }>
39       <Text style = { estilo.titulo }>
40         Escolha os valores mínimo e máximo para fazer o sorteio
41       </Text>
42
43       <View style = { estilo.linhaInput }>
44         <Text>Valor Mínimo: </Text>
45
46         <TextInput
47           textAlign = "center"
48           keyboardType = "number-pad"
49           maxLength = { 5 }
50           autoFocus = { true }
51           style = { flagInputFocus === "txt_min" ? estilo.inputFocus : estilo.inputNormal }
52           onFocus={ () => setFlagInputFocus("txt_min") }
53           onBlur = { () => setFlagInputFocus("") }
54           value = { valorMinimo.toString() }
55           onChangeText = { valor => setValorMinimo(valor) } />
56       </View>
57
58       <View style = { estilo.linhaInput }>
59         <Text>Valor Máximo: </Text>
60
61         <TextInput
62           textAlign="center"
63           keyboardType="number-pad"
64           maxLength={5}
65           style = { flagInputFocus === "txt_max" ? estilo.inputFocus : estilo.inputNormal }
66           onFocus = { ()=>setFlagInputFocus("txt_max") }
67           onBlur = { ()=>setFlagInputFocus("") }
68           value = { valorMaximo.toString() }
69           onChangeText = { setValorMaximo } />
70       </View>
71
72       <View style = { estilo.boxBotao }>
73         <Button title="Sortear" onPress = { gerarNumero } color="#1f4f66"/>
74       </View>
75
76       <View style = { estilo.boxNumero }>
77         <Text style = { estilo.numero }>{ numeroSorteado }</Text>
78       </View>
79     </View>
80   );
81 }
82
83
84 export default TelaInicial;

```

```

1 import { StyleSheet } from "react-native";
2
3 const estilo = StyleSheet.create({
4   tela: {
5     width: '100%',
6     justifyContent: "center",
7     alignItems: 'center',
8   },
9
10  boxNumero: {
11    borderColor: '#13b0c5',
12    backgroundColor: '#13b0c5',
13    borderwidth: 5,
14    height: 150,
15    width: 150,
16    borderRadius: 75,
17    justifyContent: "center",
18    alignItems: "center",
19    marginBottom: 50
20  },
21
22  titulo: {
23    fontSize: 14,
24    marginVertical: 30,
25    paddingHorizontal: 20,
26    textAlign: "center"
27  },
28
29  numero: {
30    fontSize: 70,
31    color: '#fff'
32  },
33  boxBotao: {
34    width: 200
35  }
36
37 });
38
39 export default estilo;

```

## React Native: Sorteador personalizado de números

Introdução

App.js

✓ componentes

- TelaInicial/index.js
- TelaInicial/estilo.js
- Titulo/index.js**
- Titulo/estilo.js

## React Native: Sorteador personalizado de números

Neste projeto criaremos um sorteador utilizando dois inputs recebidos pelo usuário para sortear um número aleatório. Para receber estes inputs do usuário vamos utilizar o componente TextInput.

### Titulo/index.js

```

1 import React from 'react';
2 import { View, Text } from 'react-native';
3 import estilo from './estilo';
4
5 const Titulo = () => {
6   return (
7     <View style={estilo.boxTitulo}>
8       <Text style={estilo.titulo}>Sorteio DevMedia</Text>
9       <Text style={estilo.subtitle}>Hora de ver quem é o vencedor</Text>
10    </View>
11  );
12}
13
14 export default Titulo;

```

## React Native: Sorteador personalizado de números

Introdução

App.js

✓ componentes

TelaInicial/index.js

TelaInicial/estilo.js

Titulo/index.js

Titulo/estilo.js

## React Native: Sorteador personalizado de números

Neste projeto criaremos um sorteador utilizando dois inputs recebidos pelo usuário para sortear um número aleatório. Para receber estes inputs do usuário vamos utilizar o componente TextInput.

### Titulo/estilo.js

```
1 import { StyleSheet } from "react-native";
2
3 const estilo = StyleSheet.create({
4   boxTitulo: {
5     height: 80,
6     backgroundColor: "#1f4f66",
7     paddingHorizontal: 10,
8     paddingTop: 10,
9     justifyContent: 'center'
10 },
11
12   titulo: {
13     color: '#0fc3d4',
14     fontWeight: '700',
15     fontSize: 20
16 },
17
18   subtítulo: {
19     color: '#fff'
20 },
21 });
22
23 export default estilo;
```