

```

import React from "react";
import { Text, View, FlatList, StyleSheet } from "react-native";
import produtos from "../produtos";

/* FLATLIST:

    No React temos 2 tipos de criação de lista, podemos criar listas usando o "map" e renderizando cada item por vez, ou podemos usar
    o componente "FlatList" do React.

    Nesse exemplo vamos ver como criar uma lista usando FlatList, a forma mais recomendada de gerar uma lista...

*/

//Vamos ter abaixo uma lista de produtos gerada a partir de um array de produtos que importamos...
export default () => {
  return (
    <View style={styles.background}>
      <Text style={styles.titulo}>Lista de Produtos FlatList</Text>
      {/* Note que é apenas um único componente que vai ser usado para gerar a lista... */}
      <FlatList
        // Veja as propriedades...
        // data vai receber o array ou objeto com os produtos...
        data={produtos}
        // keyExtractor é usado para gerar as keys, ele recebe uma callback que automaticamente irá puxar os valores passados
em data...
        keyExtractor={item => `${item.id}`}
        //renderItem é usado para gerar os itens da lista, como se fosse um map.
        //Note que ele tem algumas particularidades, ele tem que receber uma callback onde o parâmetro da callback
obrigatoriamente deve ser uma chave item, TEM QUE TER ESSE NOME: item!
        //Podemos dar a item um nome do nosso desejo atribuindo um nome qualquer, como fizemos com "produto". E com o
parâmetro "item" ou "produto", podemos puxar os valores de cada iteração do array ou objeto.
        renderItem={({item: produto}) => {
          return <Text style={styles.item}>{produto.id} - {produto.produto} tem o preço de R$ {produto.preco}</Text>
        }}
      />
    </View>
  )
}

```

```
}  
  
const styles = StyleSheet.create({  
  
  background: {  
    marginTop: '10%',  
    marginHorizontal: '15%',  
  },  
  titulo: {  
    color: 'navy',  
    fontSize: 24,  
    fontWeight: 'bold',  
  },  
  item: {  
    color: 'black',  
    fontSize: 16,  
  }  
  
})
```

ARRAY DE PRODUTOS...

```
const produtos = [  
  {id: 1, produto: 'Cadeira', preco: '80,00'},  
  {id: 2, produto: 'Mesa', preco: '1.500,00'},  
  {id: 3, produto: 'Armário', preco: '1.300,00'},  
  {id: 4, produto: 'CookTop', preco: '450,00'}  
]  
  
export default produtos
```

ARQUIVO DO APP.JS...

```
import React from 'react'  
import ListaComFratList from './src/componentes/Listas/ListaComFratList'
```

```
export default () => {
  return (
    /* Coloque seus componentes abaixo...*/
    <ListaComFratList />
  )
}
```

RESULTADO NO CELULAR...

The screenshot displays a development environment with a mobile app preview on the left and a code editor on the right. The mobile app, titled "Lista de Produtos FlatList", shows a list of items with their prices:

- 1 - Cadeira tem o preço de R\$ 80,00
- 2 - Mesa tem o preço de R\$ 1.500,00
- 3 - Armário tem o preço de R\$ 1.300,00
- 4 - CookTop tem o preço de R\$ 450,00

The code editor shows the implementation of the `ListaComFratList.js` component. The code includes comments explaining the use of `keyExtractor` and `renderItem` props for the `FlatList` component. The `renderItem` function uses the `item` prop to display the item's ID and price.

```
23 // keyExtractor é usado para gerar as keys, ele recebe uma callback que
24 // automaticamente irá puxar os valores passados em data...
25 keyExtractor={item => `${item.id}`}
26 //renderItem é usado para gerar os itens da lista, como se fosse um map.
27 //Note que ele tem algumas particularidades, ele tem que receber uma
28 //callback onde o parâmetro da callback obrigatoriamente deve ser uma chave
29 //item, TEM QUE TER ESSE NOME: item!
30 //Podemos dar a item um nome do nosso desejo atribuindo um nome qualquer,
31 //como fizemos com "produto". E com o parâmetro "item" ou "produto",
32 //podemos puxar os valores de cada iteração do array ou objeto.
33 //OBS: geralmente não colocamos a callback aqui, nós extraímos ela e só
34 //referenciamos aqui dentro, mas como estamos exemplificando, fica melhor
35 //mostrar assim...
36 renderItem={({item: produto}) => {
37   return <Text style={styles.item}>{produto.id} - {produto.produto} tem
38     o preço de R$ {produto.preco}</Text>
39 }}
40 </View>
```

The bottom of the screen shows the terminal output, indicating a successful build and the start of the app on the emulator.

```
BUILD SUCCESSFUL in 9m 13s
38 actionable tasks: 2 executed, 36 up-to-date
info Connecting to the development server...
info Starting the app on "emulator-5554"...
Starting: Intent { cmp=com.componentes/.MainActivity }
```