App tarefas II

Funcionalidades para adicionar:

Passo 1:

Passo 2:

```
export default function App(){
  return(
    <View style={styles.container}>
      <Text style={styles.title}>Tarefas</Text>

      <View style={styles.containerInput}>
        <TextInput
          placeholder="Digite sua tarefa..."
          style={styles.input}
          value={tarefa}
          onChangeText={ (text) => setTarefa(text) }
        />

        <TouchableOpacity style={styles.buttonAdd} onPress={handleAdd}>
          <FontAwesome name="plus" size={20} color="#FFF" />
        </TouchableOpacity>
      </View>

      <FlatList
        data={list} Locaiza o array da lista
        keyExtractor={ (item) => item.key} extrai o ID ou chave da lista
        renderItem={ [redacted] } Renderiza cada item usando o componente `Tarefa`
        style={styles.list}
      />
    </View>
  )}
```

Passo 3:

```
import React, { useState } from 'react';
import {
  View,
  Text,
  StyleSheet,
  TouchableOpacity,
  TextInput,
  FlatList    // Componente para listas roláveis
} from 'react-native'
```

Passo 4: Criar uma pasta: scr e dentro dela o arquivo Tarefa.js

Passo 5:

Tarefa.js U

my-project > src > Tarefa.js > Tarefa

```
1
2   import React from 'react';
3   import { View, Text, StyleSheet, TouchableOpacity } from 'react-native'
4
5   import { FontAwesome } from '@expo/vector-icons'
6
7   export default function Tarefa({ data }){
8     return(
9       <View style={styles.container}>
10
11        <TouchableOpacity style={styles.button}>
12          <FontAwesome name="trash" size={20} color="#22272e" />
13        </TouchableOpacity>
14
15        <Text>{data.item}</Text>
16
17      </View>
18    )
19  }
```

Recebe uma propriedade `data`, que contém os detalhes de cada tarefa.

Botão que pode ser pressionado para ações futuras, como remover a tarefa.

ícone da lixeira

Exibe o texto da tarefa, passado pela propriedade `data.item`.

Passo 6: Retornar a pagina App.js e adicionar o componente tarefa

```
    <FlatList
      data={list}
      keyExtractor={ (item) => item.key}
      renderItem={ ({ item }) => <Tarefa data={item} /> }
      style={styles.list}
    />
  </View>
)}
```

Adicionar o componente tarefa

Passo 7:

```
export default function Tarefa({ data }){
  return(
    <View style={styles.container}>

      <TouchableOpacity style={styles.button}>
        <FontAwesome name="trash" size={20} color="#22272e"  />
      </TouchableOpacity>

      <Text>{data.item}</Text>

    </View>
  )
}
const styles = StyleSheet.create({
  container:{
    backgroundColor: 'rgba(196,196,196, 0.20 )',
    marginTop: 12,
    padding: 12,
    borderRadius: 4,
    flexDirection: 'row'
  },
  button:{
    marginRight: 8,
  }
})
```

Atenção: 📣

> Quando usamos medidas fixas, como `padding: 12` ou `marginTop: 20`, o React Native interpreta esses valores como **pixels**. Isso quer dizer que a interface vai parecer

> semelhante em diferentes densidades de tela, pois o React Native ajusta automaticamente para que tudo fique proporcional. Mas é importante saber que, se quisermos que o layout funcione bem em dispositivos com tamanhos de tela diferentes (como celulares pequenos e tablets grandes), precisamos pensar em alternativas como `flex`, valores proporcionais ou bibliotecas de responsividade.
>
> Nesse projeto, vamos focar mais na funcionalidade e menos na responsividade, mas lembrem-se que os valores que estamos usando são em **pixels**."