

Tutorial CRUD em Android / iOS com React Native

Parte 4

João Ozorio

Tutorial CRUD em Android / iOS com React Native

Parte 4

Chegamos à última parte do nosso tutorial, a parte 4! Vamos finalmente terminar o que há de ser feito.

#4 Atualizando um item

Agora que já temos os campos preenchidos na tela AppForm.js quando clicamos no botão editar da tela de listagem, vamos ajustar a lógica do botão Salvar para que ele identifique quando é uma edição ou novo cadastro, para fazer a operação necessária em nosso banco de dados.

Esta alteração é em dois pontos: no módulo Database.js e outra perna no AppForm.js. Primeiro vamos no módulo de acesso a dados.

```
1  async function saveItem(listItem, id){  
2    listItem.id = id ? id : new Date().getTime()  
3    const savedItems = await getItems()  
4  
5    if(id){  
6      const index = await savedItems.findIndex(item => item.id === id);  
7      savedItems[index] = listItem;  
8    }  
9    else  
10      savedItems.push(listItem);  
11  
12    return AsyncStorage.setItem('items', JSON.stringify(savedItems));  
13 }
```

#4 Atualizando um item

Agora, o código abaixo vai na função do botão de Salvar da AppForm.js.

```
1 | async function handleButtonPress(){
2 |   const listItem = {descricao, quantidade: parseInt(quantidade)};
3 |   Database.saveItem(listItem, id)
4 |     .then(response => navigation.navigate("AppList", listItem));
5 | }
```

#4 Excluindo um item

E finalmente vamos à quarta e última letra do CRUD, o D de Delete!

Para fazer isso, não vou simplesmente definir uma function que exclui o registro com determinado id no Async Storage, mas antes disso quero questionar o usuário se ele tem certeza do que ele vai fazer.

Para exibir um popup de confirmação, adicione o seguinte código na function handleDeletePress do arquivo AppItem.js.

```
1 function handleDeletePress(){
2   Alert.alert(
3     "Atenção",
4     "Você tem certeza que deseja excluir este item?",
5     [
6       {
7         text: "Não",
8         onPress: () => console.log("Cancel Pressed"),
9         style: "cancel"
10      },
11      { text: "Sim", onPress: () => console.log(`${props.id} deleted`) }
12    ],
13    { cancelable: false }
14  );
15 }
```

#4 Excluindo um item

Para que o código acima funcione, você vai ter de adicionar um import adicional também, para o componente Alert e tem de adicionar a função handleDeletePress no onPress do botão de exclusão.

```
1 import React from 'react';
2 import {StyleSheet, Text, View, TouchableOpacity, Alert} from 'react-native';
3 import Database from './Database';
```

#4 Excluindo um item

A classe Alert possui uma function alert que serve para criar mensagens no estilo popup.

O primeiro parâmetro desta função é o título do alerta, o segundo a mensagem o terceiro é o array de botões para o usuário escolher uma opção. Cada botão tem um texto e uma função onPress própria.

E no app, ao clicar no botão de excluir de um dos itens cadastrados, você verá o alerta abaixo.

#4 Excluindo um item

Agora que temos o alerta funcionando, vamos na Database.js criar a function de exclusão.



#4 Excluindo um item

Aqui a function é bem direta: pega todos os itens, procura o índice daquele que possui o id recebido por parâmetro, exclui esse elemento do array e salva tudo de novo.

Para usar esta função, vamos mudar levemente o código do nosso Alert na AppItem.js.



#4 Adicionando ícones

Agora que finalizamos as funcionalidades principais, que tal fecharmos com um pequeno toque na interface, adicionando ícones aos botões de editar e excluir?

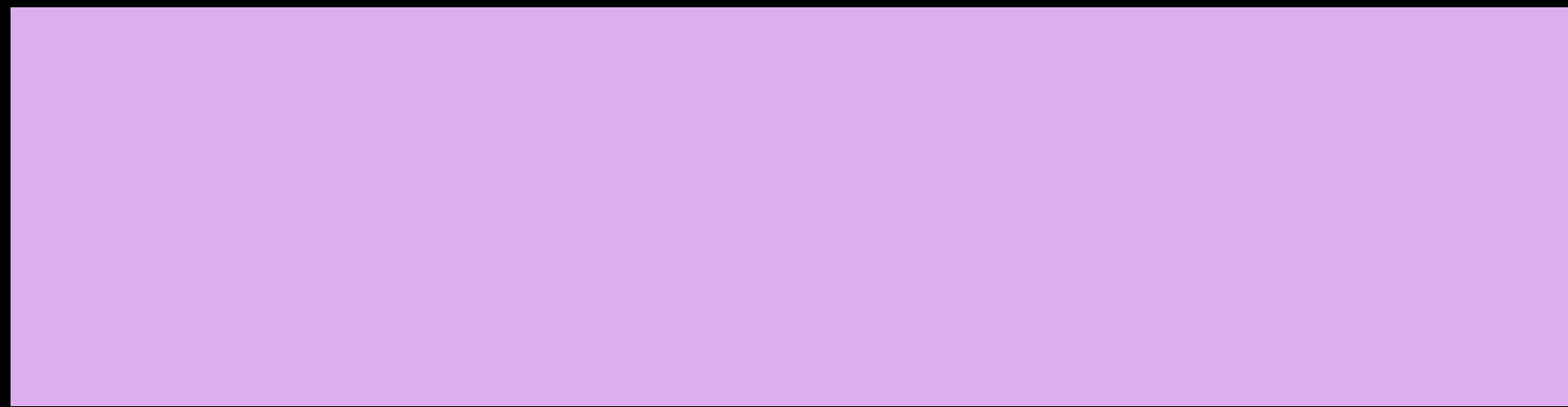
Para fazer isso, vamos importar um pacote que já vem no Expo por padrão com todos os ícones que podemos precisar na maioria dos apps dentro de uma lib `@expo/vector-icons`.

Assim, volte na sua `AppItem.js` e adicione a importação abaixo do pacote de ícones Feather.



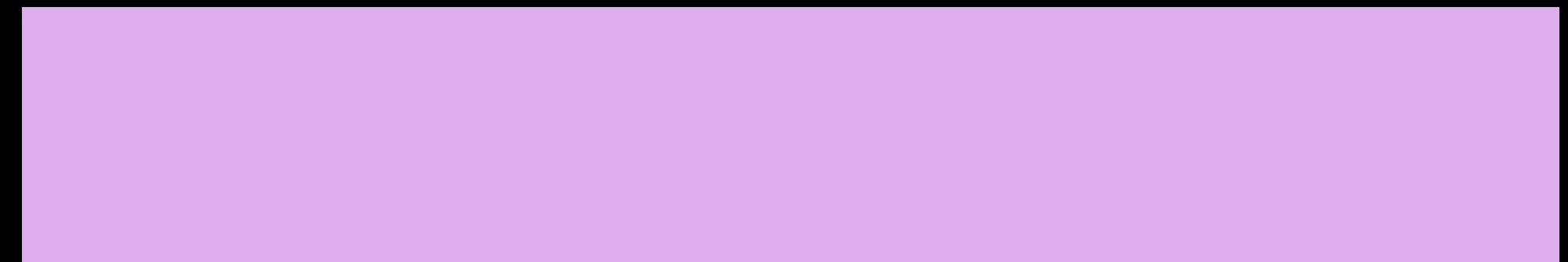
#4 Adicionando ícones

E agora, dentro dos seus botões, você poderá usar a tag `Icon` ao invés de `Text`, indicando qual o ícone da biblioteca Feather você vai querer.



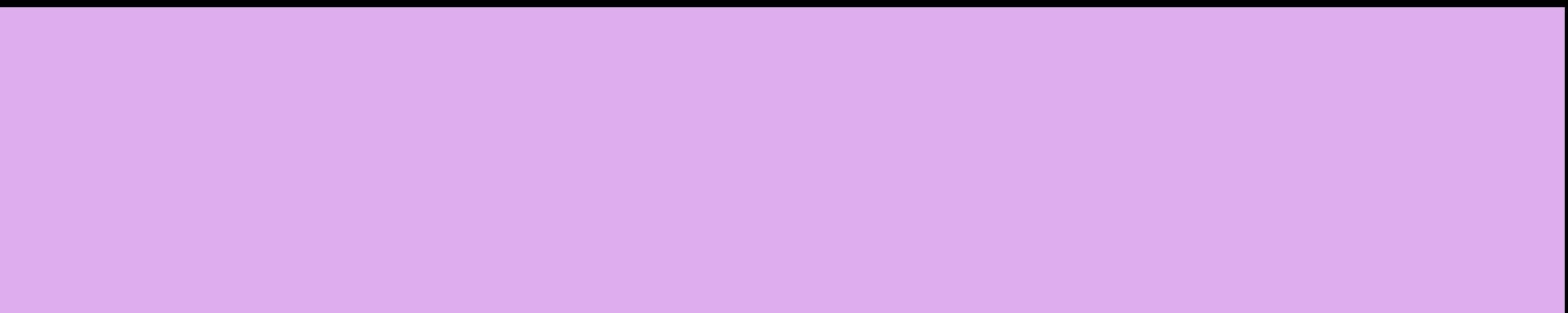
#4 Adicionando ícones

Opcionalmente, você pode alterar o botão de Salvar da tela AppForm.js para que, além do texto, exiba um ícone também. Antes de fazer isso, adicione o import aos ícones, como fizemos na AppItem.js. Somente depois, modifique o trecho de código do TouchableOpacity para ficar como abaixo.



#4 Adicionando ícones

Para que eles fiquem lado a lado e em tamanhos condizentes, temos de fazer dois ajustes nos estilos deste módulo. Um deles é um ajuste no estilo `buttonText` e o outro é a criação de um estilo novo chamado `buttonContainer`.



O fim

E com isso finalizamos esta série de tutoriais.

A força esteve com você.