

Tutorial CRUD em Android / iOS com React Native

Parte 3

João Ozorio

Tutorial CRUD em Android / iOS com React Native

Parte 3

Na parte dois, fizemos o código do cadastro dos itens, mas ainda de forma “fixa”. Hoje, prosseguiremos refaturando e aprimorando nosso app, até chegar na parte 4 final, onde tomará a tão desejada forma dele!

#3 Refatorando o acesso a dados

Agora que nossa aplicação está começando a ficar um pouco maior e o código mais complexo, vamos separar a lógica de acesso a dados do restante da aplicação.

Crie um arquivo Database.js e dentro dele vamos colocar toda a lógica de acesso a dados do nosso CRUD. A começar pela única função que lida com dados atualmente, a saveItem.

```
1 import AsyncStorage from '@react-native-async-storage/async-storage';
2
3 async function saveItem(listItem){
4     listItem.id = new Date().getTime();
5     let savedItems = [];
6     const response = await AsyncStorage.getItem('items');
7
8     if(response) savedItems = JSON.parse(response);
9     savedItems.push(listItem);
10
11    return AsyncStorage.setItem('items', JSON.stringify(savedItems));
12 }
13
14 module.exports = {
15     saveItem
16 }
```

#3 Refatorando o acesso a dados

Agora no AppForm.js, vamos importar alguns módulos.
Segue código abaixo:

```
1 import React, {useState, useEffect} from 'react';
2 import { StatusBar } from 'expo-status-bar';
3 import { StyleSheet, Text, View, TextInput, TouchableOpacity } from 'react-native';
4 import Database from './Database';
```

#3 Refatorando o acesso a dados

E depois modificar a parte que usava esse código para que ela apenas lide com os estados e com a navegação, responsabilidades da interface.

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

#3 Retornando os itens do banco de dados

Chegando na letra R do CRUD (retrieve) nosso desafio agora é retornar os dados salvos pela tela de cadastro para popular a nossa lista, tanto quando abrimos o app pela primeira vez, quanto quando um dado é cadastrado, atualizado ou excluído.

Vamos no nosso módulo Database.js para adicionar mais uma function.

```
1 function getItems(){
2     return AsyncStorage.getItem('items')
3         .then(response => {
4             if(response)
5                 return Promise.resolve(JSON.parse(response));
6             else
7                 return Promise.resolve([]);
8         })
9 }
```

#3 Retornando os itens do banco de dados

Basicamente ela pega os itens do AsyncStorage e retorna em forma de array através de uma promise. Adicione essa função no module.exports.

```
1 module.exports = {  
2     saveItem,  
3     getItems  
4 }
```

#3 Retornando os itens do banco de dados

Vamos importar alguns módulos no `AppList.js` : o `useEffect` e o `Database`.

```
1 import { StatusBar } from 'expo-status-bar';
2 import React, {useState, useEffect} from 'react';
3 import { StyleSheet, View, Text, ScrollView } from 'react-native';
4 import AppItem from './AppItem';
5 import Database from './Database';
```

#3 Retornando os itens do banco de dados

Hora de mudar a função adicionando dois componentes:
route e **navigation**.

```
1 | export default function AppList({ route, navigation }) {
```

O **navigation** usaremos mais tarde para trocar de tela.
O **route**, são os parâmetros de navegação que trouxeram
até esta tela. Isso ficará mais claro mais tarde também.

#3 Retornando os itens do banco de dados

No código abaixo, eu removi os itens que havia adicionado manualmente só para que o app tivesse dados para exibir.

```
1 const [items, setItems] = useState([]);  
2  
3 useEffect(() => {  
4   Database.getItems().then(items => setItems(items));  
5 }, [route]);
```

O useEffect irá ser disparado toda vez que a nossa variável route seja alterada, ou seja, toda vez que entrar nesta tela vindo de outra.

#3 Selecionando um item

A próxima etapa do nosso CRUD é o U, de update.

O funcionamento aqui é o de enviar o usuário para a tela de cadastro, mas com os campos já preenchidos e, quando o usuário fizer a alteração que deseja e clique em salvar, ele deve editar aquele registro ao invés de salvar um novo, como seria o comportamento normal.

Aqui temos dois comportamentos a serem programados: o de redirecionamento a partir da seleção de um item da lista e o de salvamento inteligente (atualização x novo cadastro).

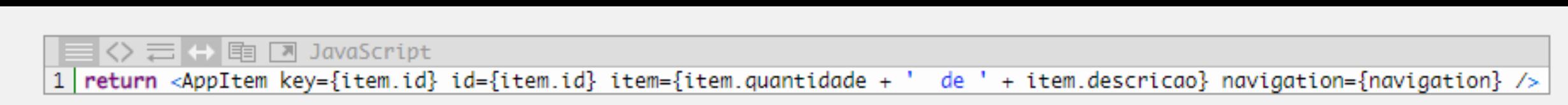
Vamos começar redirecionando o usuário no press do botão de Editar, que é o primeiro comportamento necessário.

Lembra que capturamos um objeto navigation dentro dos parâmetros da function AppList?

```
1 export default function AppList({ route, navigation }) {
```

#3 Selecionando um item

Agora vamos usar este navigation aí, no mesmo arquivo.
Note que tem uma propriedade navigation ali na tag AppItem também,
que é para passarmos o objeto de navegação para dentro do AppItem.



The image shows a screenshot of a code editor window. The title bar says "JavaScript". The code in the editor is:

```
1 | return <AppItem key={item.id} id={item.id} item={`${item.quantidade} de ${item.descricao}`} navigation={navigation} />
```

#3 Selecionando um item

Agora, voltando ao Database.js, vamos criar a function que vai retornar apenas um item (não esqueça de adicionar ela no module.exports).

```
1 async function getItem(id){  
2     const savedItems = await getItems();  
3     return savedItems.find(item => item.id === id);  
4 }
```

#3 Selecionando um item

E no AppItem.js, vamos importar o Database.js.

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

#3 Selecionando um item

E ainda no AppItem.js, vamos criar uma function que vai servir de handler para o press do botão de editar, usando esta outra function que acabamos de criar no Database.js, como abaixo.

```
1 | async function handleEditPress(){
2 |   const item = await Database.getItem(props.id);
3 |   props.navigation.navigate("AppForm", item);
4 | }
```

#3 Selecionando um item

Nesta function, nós estamos pegando o item com o id que foi passado na propriedade id do AppItem e retornando o item que queremos (buscando por id) e enviando ele para o AppForm na navegação de tela.

```
1 <TouchableOpacity  
2   style={styles.editButton}  
3   onPress={handleEditPress}>  
4   <Text style={styles.buttonText}>Editar</Text>  
5 </TouchableOpacity>
```

E vamos associar esta function ao onPress do botão de edit neste mesmo arquivo.

#3 Selecionando um item

Com isso, acontecerá a troca de tela ao clicar no botão editar, passando por parâmetro o item da lista de compras inteiro. Para poder capturar este item na AppForm.js, teremos de alterar a assinatura da function, como abaixo, incluindo o objeto route.

```
1 export default function AppForm({ route, navigation }) {  
2   const id = route.params ? route.params.id : undefined;  
3   const [descricao, setDescricao] = useState('');  
4   const [quantidade, setQuantidade] = useState('');
```

#3 Selecionando um item

Agora, logo abaixo do trecho de código acima, vamos adicionar um efeito que vai ser disparado toda vez que o objeto route for modificado.

```
1 useEffect(() => {
2   if(!route.params) return;
3   setDescricao(route.params.descricao);
4   setQuantidade(route.params.quantidade.toString());
5 }, [route])
```

#3 Selecionando um item

Assim, a cada nova navegação realizada (ou seja, route com params diferentes), nós pegamos os parâmetros do route para setar o estado da descrição e da quantidade.

Não esqueça de revisar os seus imports, para que estejam como abaixo.

```
1 import React, {useState, useEffect} from 'react';
2 import { StatusBar } from 'expo-status-bar';
3 import { StyleSheet, Text, View, TextInput, TouchableOpacity } from 'react-native';
4 import Database from './Database';
```

#3 Selecionando um item

E ainda no AppItem.js, vamos criar uma function que vai servir de handler para o press do botão de editar, usando esta outra function que acabamos de criar no Database.js, como abaixo.

```
1 <TextInput  
2   style={styles.input}  
3   onChangeText={handleDescriptionChange}  
4   placeholder="O que está faltando em casa?"  
5   clearButtonMode="always"  
6   value={descricao} />  
7 <TextInput  
8   style={styles.input}  
9   onChangeText={handleQuantityChange}  
10  placeholder="Digite a quantidade"  
11  keyboardType={'numeric'}  
12  clearButtonMode="always"  
13  value={quantidade.toString()} />
```

O fim

Hoje, finalizamos uma parte muito importante dessa jornada. Hoje, você descobriu que o verdadeiro motivo de terminar algo, não é a recompensa, mas sim o processo e o que aprendeu durante.

Você está quase lá, mas não entre em pânico,
Ainda há mais coisas por vir.

Antes de você ter absorvido conhecimento, e conquistar a arma dos fortes:
A constância e a disciplina.

Antes de tudo acabar, desejo uma boa sorte para executar a 4ª e última
parte do tutorial.

Observe as flores, elas não tentam parecer belas
Elas apenas abrem suas pétalas e se viram para a luz
Você está ouvindo bem?
Uma coisa é muito mais do que parece ser.

Que a força esteja com você.