

# Mobile Application Development

Prof. Fernando Pinéo

# Async Storage

AsyncStorage é usado para armazenar e recuperar os dados de forma persistente no dispositivo.

# Async Storage



Somente armazena dados em String

The screenshot shows the Expo AsyncStorage documentation page in a web browser. The URL in the address bar is `docs.expo.dev/versions/latest/sdk/async-storage/`. The page has a dark theme. On the left, there is a sidebar with a search bar and navigation links: Home, Guides, EAS, Reference (highlighted), and Learn. Below the sidebar, there is a 'Reference version' dropdown set to 'SDK 52 (latest)' and a list of application types: Application, Asset, AsyncStorage (selected), Audio (expo-audio) with a 'NEW' badge, Audio (expo-av), and AuthSession. The main content area has the title 'AsyncStorage' and a description: 'A library that provides an asynchronous, unencrypted, persistent, key-value storage API.' It includes links to GitHub and npm, and platform-specific badges for Android, iOS, macOS, tvOS, and Web. Below this, it states 'An asynchronous, unencrypted, persistent, key-value storage API.' and has an 'Installation' section. The 'Installation' section has a 'Terminal' tab with a code block containing the command: `- npx expo install @react-native-async-storage/async-storage`. Below the code block, it says: 'If you are installing this in an existing React Native app, start by installing expo in your project. Then follow the additional instructions provided by the library's README or documentation.'

docs.expo.dev/versions/latest/sdk/async-storage/

Expo Docs

Blog Changelog Star Us on GitHub

Search Ctrl K

Home Guides EAS Reference Learn

Reference version

SDK 52 (latest)

Application Asset

- AsyncStorage
- Audio (expo-audio) NEW
- Audio (expo-av)
- AuthSession

## AsyncStorage

A library that provides an asynchronous, unencrypted, persistent, key-value storage API.

GitHub npm

Android iOS macOS tvOS Web

An asynchronous, unencrypted, persistent, key-value storage API.

## Installation

Terminal

```
- npx expo install @react-native-async-storage/async-storage
```

If you are installing this in an existing React Native app, start by installing expo in your project. Then follow the additional instructions provided by the library's README or documentation.

Link: <https://docs.expo.dev/versions/latest/sdk/async-storage/>

# Criando o projeto



```
npx create-expo-app ArmazenamentoLocal --template blank
```

# Importação necessárias



JS App.js M X

JS App.js > ...

```
1 // Importação das bibliotecas necessárias
2 import { StatusBar } from 'expo-status-bar'; // Importação para manipulação da barra de status
3 import { StyleSheet, Text, TextInput, TouchableOpacity, View, FlatList } from 'react-native'; //
4 import AsyncStorage from '@react-native-async-storage/async-storage'; // Importação do AsyncStor
5 import { useState, useEffect } from 'react'; // Importação do hook useState e useEffect do React
6
```

# Desenvolvendo a interface



```
JS App.js M X
JS App.js > App
9 export default function App() {
10   return (
11     <View style={styles.container}>
12       <Text>CADASTRO</Text> {/* Título do aplicativo */}
13
14       {/* Campo de entrada para o nome do produto */}
15       <TextInput
16         placeholder='DIGITE O NOME DO PRODUTO'
17         style={styles.input}
18         value={nomeProduto} // Valor do nome do produto
19         onChangeText={(value) => setNomeProduto(value)} // Atualiza o estado com o nome digitado
20       />
21
22       {/* Campo de entrada para o preço do produto */}
23       <TextInput
24         placeholder='DIGITE O PREÇO DO PRODUTO'
25         style={styles.input}
26         value={precoProduto} // Valor do preço do produto
27         onChangeText={(value) => setPrecoProduto(value)} // Atualiza o estado com o preço digitado
28     />
29   );
30 }
```

# Desenvolvendo a interface

JS App.js M X

JS App.js &gt; styles &gt; container &gt; marginTop

```
9 export default function App() {
```

```
  //
```

```
78
```

```
79      {/* Botão para salvar o produto */}
```

```
80      <TouchableOpacity style={styles.btn} onPress={salvar}>
```

```
81        <Text style={{ color: 'white' }}>CADASTRAR</Text>
```

```
82      </TouchableOpacity>
```

```
83
```

```
84      {/* Botão para buscar os dados salvos */}
```

```
85      <TouchableOpacity style={styles.btn} onPress={buscarDados}>
```

```
86        <Text style={{ color: 'white' }}>BUSCAR DADOS</Text>
```

```
87      </TouchableOpacity>
```



# const styles..

FIAP

```
JS App.js M X
JS App.js > styles > input
137 // Estilos do aplicativo
138 const styles = StyleSheet.create({
139   container: {
140     flex: 1,
141     backgroundColor: '#fff',
142     alignItems: 'center',
143     justifyContent: 'center',
144     gap: 10,
145     marginTop: 20
146   },
147   input: {
148     borderWidth: 1,
149     height: 50,
150     width: 300,
151     borderRadius: 15
152   },
153   btn: {
154     borderWidth: 1,
155     height: 50,
156     width: 300,
157     borderRadius: 15,
158     backgroundColor: 'blue',
159     justifyContent: 'center',
160     alignItems: 'center'
161   }
162 });
```

# Esse será o resultado

FIAP

Android Emulator - Medium\_Phone\_2:5554

12:54

CADASTRO

DIGITE O NOME DO PRODUTO

DIGITE O PREÇO DO PRODUTO

CADASTRAR

BUSCAR DADOS

# Criando o função para salvar as informações

```
JS App.js M X
JS App.js > App > salvar
9 export default function App() {
20 // Função para salvar o produto no AsyncStorage
21 async function salvar() {
22   let produtos = []; // Inicializa um array vazio para os produtos
23
24   // Verifica se já existe algum dado armazenado no AsyncStorage
25   if (await AsyncStorage.getItem("PRODUTOS") !== null) {
26     // Se houver, carrega os dados para o array
27     produtos = JSON.parse(await AsyncStorage.getItem("PRODUTOS"));
28   }
29
30   // Adiciona o novo produto à lista
31   produtos.push({ nome: nomeProduto, preco: precoProduto });
32
33   // Salva os dados atualizados no AsyncStorage
34   await AsyncStorage.setItem("PRODUTOS", JSON.stringify(produtos));
35
36   alert("PRODUTO SALVO"); // Exibe uma mensagem de sucesso
37
38   // Chama a função buscarDados para atualizar a lista de produtos exibida
39   buscarDados();
40 }
```

# Limpando o formulário após salvar:

FILAP

```
JS App.js M X
JS App.js > App > Cadastrar
 7  export default function App() {
15    // ...
16
17    async function Cadastrar() {
18      let produtos = [];
19
20      //Verificar se há alguma já armazenado no AsyncStorage
21      if (await AsyncStorage.getItem("PRODUTOS") != null) {
22        produtos = JSON.parse(await AsyncStorage.getItem("PRODUTOS"))
23      }
24
25      produtos.push({ nome: nomeProduto, preco: precoProduto })
26
27      //Salvando os dados no AsyncStorage
28      await AsyncStorage.setItem("PRODUTOS", JSON.stringify(produtos))
29
30      alert("PRODUTO CADASTRADO")
31
32      //Limpando o formulário:
33      setNomeProduto('')
34      setPrecoProduto('')
35
36      BuscarDados()
37    }
  }
```

# Garantindo que a FlatList só renderize se tiver dados

FLANP

```
JS App.js M x
JS App.js > App
7 export default function App() {
57   <TouchableOpacity style={styles.btn} onPress={Cadastrar}>
58     <Text style={{color:"white"}}>Salvar</Text>
59   </TouchableOpacity>
60
61   <FlatList
62     data={listaProdutos}
63     renderItem={({item, indice})=>{
64       if (!item || !item.nome) return null; // Garantir que o item não seja nulo antes de renderizar
65       return(
66         <View style={styles.listarFlat}>
67           <View>
68             <Text>NOME:{item.nome} - PRECO:{item.preco}</Text>
69           </View>
70         </View>
71       )
72     }}
73   />
74
75   <StatusBar style="auto" />
76 </View>
77 );
```

# JSON.stringify



O método `JSON.stringify()` converte objetos JavaScript (como arrays ou objetos) em uma string JSON.

JS const objeto = { nome: "Produto", preco: 100 } •

```
1 const objeto = { nome: "Produto", preco: 100 };
2 const objetoComoString = JSON.stringify(objeto);
3 console.log(objetoComoString); // Saída: '{"nome":"Produto","preco":100}'
```

# JSON.parse

O método `JSON.parse()` converte uma string JSON de volta para um objeto JavaScript.

JS const stringJson = '{"nome":"Produto","p Untitled-1 •

```
1 const stringJson = '{"nome":"Produto","preco":100}';
2 const objeto = JSON.parse(stringJson);
3 console.log(objeto); // Saída: { nome: 'Produto', preco: 100 }
```

# Função buscarDados()

JS App.js M X

JS App.js > App

```
9 export default function App() {  
41  
42   // Função para buscar os produtos salvos no AsyncStorage  
43   async function buscarDados() {  
44     const p = await AsyncStorage.getItem("PRODUTOS"); // Recupera os dados do AsyncStorage  
45     setListProdutos(JSON.parse(p)); // Atualiza a lista de produtos com os dados recuperados  
46   }  
47
```

# FlatList para exibir os dados



```
JS App.js M X
JS App.js > App
9  export default function App() {
10
11    <FlatList
12      data={listProdutos} // Passa os dados dos produtos para a FlatList
13      renderItem={({ item, index }) => {
14        return (
15          <View style={{
16            width: 300,
17            borderWidth: 1,
18            borderRadius: 15,
19            height: 80,
20            alignItems: 'center',
21            justifyContent: 'center',
22            marginVertical: 3
23          }}>
24
25            <View>
26              { /* Exibe o nome e preço do produto */ }
27              <Text style={{ fontSize: 18 }}>NOME: {item.nome} PREÇO: {item.preco}</Text>
28            </View>
29          )
30        }
31      />
32    />
33  }
34}
```



# Chamando os dados na inicialização do app utilizando o hook useEffect



```
JS App.js M X
JS App.js > App
1 // Importação das bibliotecas necessárias
2 import { StatusBar } from 'expo-status-bar'; // Importação para manipulação da barra de status
3 import { StyleSheet, Text, TextInput, TouchableOpacity, View, FlatList } from 'react-native'; //
4 import AsyncStorage from '@react-native-async-storage/async-storage'; // Importação do AsyncStor
5 import { useState, useEffect } from 'react'; // Importação do hook useState e useEffect do React
6 import { SafeAreaProvider, SafeAreaView } from 'react-native-safe-area-context';
7
8 // Função principal do componente App
9 export default function App() {
10   // Declaração dos estados do aplicativo
11   const [nomeProduto, setNomeProduto] = useState(''); // Armazena o nome do produto
12   const [precoProduto, setPrecoProduto] = useState(''); // Armazena o preço do produto
13   const [listProdutos, setListProdutos] = useState([]); // Armazena a lista de produtos
14
15   // useEffect para buscar os dados quando o componente for montado
16   useEffect(() => {
17     buscarDados(); // Chama a função para buscar os dados salvos no AsyncStorage
18   }, []); // O array vazio [] garante que a função só execute uma vez, após o primeiro render
19 }
```

Quando usar?: Quando você precisa executar código após a renderização do componente, como fazer requisições a APIs, manipular dados ou executar ações após mudanças no estado ou props.

Como funciona?: O `useEffect` executa o código dentro dele quando o componente é renderizado ou quando alguma dependência muda.

# Executar uma vez na inicialização



```
JS useEffect() => { Untitled-1 •
1  useEffect(() => {
2    console.log("O componente foi renderizado!");
3    // Você pode fazer algo aqui, como buscar dados de uma API.
4  }, []); // A lista vazia significa que vai executar apenas uma vez
```

O [] significa que não há dependências, então o `useEffect` só vai rodar uma vez (após a primeira renderização).

# Executar sempre que uma variável mudar de estado

JS const [nome, setNome] = useState(""); Untitled-1 •

```
1 const [nome, setNome] = useState("");
2
3 useEffect(() => {
4   console.log("O nome mudou:", nome);
5 }, [nome]); // O useEffect vai rodar toda vez que 'nome' mudar
```

Observe que agr há uma dependência sendo especificada, no caso a variável nome.

# Criando um botão para excluir um produto

FILAP

```
JS App.js M X
JS App.js > App
  9  export default function App() {
 92      renderItem=(({ item, index }) => {
109          /* Botão para excluir um produto */
110          <TouchableOpacity
111              style={{
112                  flexDirection: 'column',
113                  justifyContent: "space-around",
114                  alignItems: 'center',
115                  backgroundColor: 'red',
116                  borderRadius: 12,
117                  width: 100,
118                  alignSelf: 'flex-end',
119                  height: 25
120              }}
121              onPress={() => deletarProduto(index)} // Deleta o produto ao pressionar o botão
122          >
123              <Text>EXCLUIR</Text>
124          </TouchableOpacity>
125      </View>
126  );
127  }>
```

# Função deletarProduto()

FIAP

JS App.js M X

JS App.js > App

```
9  export default function App() {  
47  
48    // Função para deletar um produto  
49    async function deletarProduto(index) {  
50      const tempDados = listProdutos; // Cria uma cópia da lista atual  
51      const dados = tempDados.filter((item, ind) => {  
52        return ind !== index; // Filtra o item que será excluído da lista  
53      });  
54  
55      setListProdutos(dados); // Atualiza a lista de produtos no estado  
56      await AsyncStorage.setItem("PRODUTOS", JSON.stringify(dados)); // Atualiza o AsyncStorage com  
57    }  
58  }
```

Se o índice for diferente, o produto é mantido na lista



# Função EditarProduto()

Criando um estado para o produto que está sendo editado.

```
JS App.js M X
JS App.js > App
1 // Importação das bibliotecas necessárias
2 import { StatusBar } from 'expo-status-bar'; // Importação para manipulação da barra de status
3 import { StyleSheet, Text, TextInput, TouchableOpacity, View, FlatList } from 'react-native'; // Importação
4 import AsyncStorage from '@react-native-async-storage/async-storage'; // Importação do AsyncStorage para
5 import { useState, useEffect } from 'react'; // Importação do hook useState e useEffect do React
6
7 // Função principal do componente App
8 export default function App() {
9   // Declaração dos estados do aplicativo
10   const [nomeProduto, setNomeProduto] = useState(''); // Armazena o nome do produto
11   const [precoProduto, setPrecoProduto] = useState(''); // Armazena o preço do produto
12   const [listProdutos, setListProdutos] = useState([]); // Armazena a lista de produtos
13   const [produtoEditado, setProdutoEditado] = useState(null); // Estado para o produto que está sendo edi
14
```

# Função EditarProduto()

FIAP

```
JS App.js M X
JS App.js > App
8  export default function App() {
49  async function deletarProduto(index) {
51    const dados = tempDados.filter((item, ind) => {
52      item.id !== index; // Remove o item que será excluído da lista
53    });
54
55    setListProdutos(dados); // Atualiza a lista de produtos no estado
56    await AsyncStorage.setItem("PRODUTOS", JSON.stringify(dados)); // Atualiza o AsyncStorage com os dados
57  }
58
59
60  function editarProduto(index) {
61    const produto = listProdutos[index]; // Obtém o produto com o índice selecionado
62    setNomeProduto(produto.nome); // Preenche o campo de nome com o nome do produto
63    setPrecoProduto(produto.preco); // Preenche o campo de preço com o preço do produto
64    setProdutoEditado({ index }); // Define o índice do produto a ser editado
65  }
```



# Modificação do botão Salvar

FIAP

```
JS App.js M X
JS App.js > App
8  t default function App() {
79  /* Campo de entrada para o preço do produto */
80  <TextInput
81    placeholder='DIGITE O PREÇO DO PRODUTO'
82    style={styles.input}
83    value={precoProduto} // Valor do preço do produto
84    onChangeText={({value) => setPrecoProduto(value)} // Atualiza o estado com o preço digitado
85  />
86
87  /* Botão de salvar ou atualizar o produto */
88  <TouchableOpacity style={styles.btn} onPress={salvar}>
89    <Text style={{ color: 'white' }}>
90      {produtoEditado ? 'ATUALIZAR' : 'CADASTRAR'} /* Alteração do txt do btn dependendo do esta
91    </Text>
92  </TouchableOpacity>
```

# FlatList, passando o índice para o Editar F1/\P

JS Appjs M X

JS Appjs > App > <function>

```
7 export default function App() {
68   <TouchableOpacity style={styles.btn} onPress={Cadastrar}>
69     <Text style={{color:"white"}}>{produtoEditado?"Atualizar":"Salvar"}</Text>
70   </TouchableOpacity>
71
72   <FlatList
73     data={listaProdutos}
74     renderItem={({item,index})=>{
75       if (!item || !item.nome) return null; // Garantir que o item não seja nulo antes de renderizar
76     return(
77       <View style={styles.listarFlat}>
78         <View>
79           <Text>NOME:{item.nome} - PRECO:{item.preco}</Text>
80         </View>
81         <View style={{flexDirection:'row'}}>
82           <TouchableOpacity style={styles.btnExcluir}>
83             <Text>Excluir </Text>
84           </TouchableOpacity>
85           <TouchableOpacity style={styles.btnEditar}onPress={()=>EditarProduto(index)}>
86             <Text>Editar</Text>
87           </TouchableOpacity>
88         </View>
89       </View>
90     )
  }
```

# Lógica do atualizar na função Salvar

FIA/P

JS App.js M X

JS App.js > App > Cadastrar

```
7   export default function App() {
15   async function Cadastrar(){
19     if(await AsyncStorage.getItem("PRODUTOS")!=null){
20       produtos = JSON.parse(await AsyncStorage.getItem("PRODUTOS"))
21     }
22
23     if (produtoEditado) {
24       // Atualizar o produto existente
25       produtos[produtoEditado.index] = { nome: nomeProduto, preco: precoProduto };
26     } else {
27       // Adicionar um novo produto
28       produtos.push({ nome: nomeProduto, preco: precoProduto });
29     }
30
31     //Salvando os dados no AsyncStorage
32     await AsyncStorage.setItem("PRODUTOS",JSON.stringify(produtos))
33
34     alert(produtoEditado ? "PRODUTO ATUALIZADO" : "PRODUTO CADASTRADO");
35
36     BuscarDados()
37
38   }
39 }
```

# Colocando o botão editar do lado do botão excluir

FILAP

```
JS App.js M X
JS App.js > App > <function>
  8 export default function App() {
102   renderItem={({ item, index }) => {
141     flexDirection: 'column',
142     justifyContent: "center",
143     alignItems: 'center',
144     backgroundColor: 'red',
145     borderRadius: 12,
146     width: 100,
147     alignSelf: 'flex-end',
148     height: 25
149   }}
150   onPress={() => deletarProduto(index)} // Chama a função para deletar o produto
151 >
152   <Text>EXCLUIR</Text>
153 </TouchableOpacity>
154 </View>
155 </View>
156   };
157 } />
```

Explicação dos comentários:

Estado do produto sendo editado: A variável `produtoEditado` é usada para armazenar o índice do produto que está sendo editado. Se um produto for editado, esse índice é utilizado para substituir o produto na lista de produtos.

Função de salvar ou atualizar: Quando o botão "Salvar" ou "Atualizar" é pressionado, o código verifica se um produto está sendo editado (`produtoEditado != null`). Se sim, ele substitui o produto na lista. Caso contrário, ele adiciona um novo produto.

Função de edição: A função `editarProduto` é chamada quando o botão "Editar" é pressionado para carregar os dados do produto selecionado nos campos de entrada.

Renderização dos produtos: A `FlatList` exibe a lista de produtos cadastrados, com botões para editar ou excluir cada produto. O botão "Editar" carrega os dados do produto no formulário, e o botão "Excluir" remove o produto da lista.

# Criando máscara para o campo do Preço

```
npm install react-native-masked-text
```

Em seguida importe o componente `TextInputMask`

# Criando máscara para o campo do Preço

```
{/* Usando o TextInputMask para aplicar a máscara de reais no campo de preço */}  
<TextInputMask  
  placeholder='DIGITE O PREÇO DO PRODUTO'  
  style={styles.input}  
  value={precoProduto}  
  onChangeText={(value) => setPrecoProduto(value)}  
  type={'money'} // Define o tipo de máscara como 'money' (dinheiro)  
  options={{  
    precision: 2, // Define 2 casas decimais (R$ 100,00)  
    separator: ',', // Separador de casas decimais (ex: R$ 100,50)  
    delimiter: '.', // Separador de milhares (ex: R$ 1.000,00)  
  }}  
>
```

# Resultado

FIAP

Android Emulator - Medium\_Phone\_2:5554

2:20

CADASTRO

DIGITE O NOME DO PRODUTO

R\$500,00

CADASTRAR

BUSCAR DADOS

NOME: Maca PREÇO: 250

EDITAR EXCLUIR

NOME: Macacao PREÇO: 250

EDITAR EXCLUIR

NOME: Macarrao PREÇO: 30

EDITAR EXCLUIR



# Ocultando mensagem de erro



```
JS App.js M X
JS App.js > App
1 // Importação das bibliotecas necessárias
2 import { StatusBar } from 'expo-status-bar'; // Importação para manipulação da barra de status
3 import { StyleSheet, Text, TextInput, TouchableOpacity, View, FlatList } from 'react-native'; // Importação
4 import AsyncStorage from '@react-native-async-storage/async-storage'; // Importação do AsyncStorage para a
5 import { useState, useEffect } from 'react'; // Importação do hook useState e useEffect do React
6 import { TextInputMask } from 'react-native-masked-text';
7
8 import { LogBox } from 'react-native';
9
10 LogBox.ignoreAllLogs(true); // Desabilita todos os logs (não recomendado para produção)
11 // Função principal do componente App
12 export default function App() {
13   // Declaração dos estados do aplicativo
14   const [nomeProduto, setNomeProduto] = useState(''); // Armazena o nome do produto
```

Dúvidas?