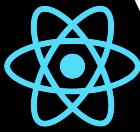


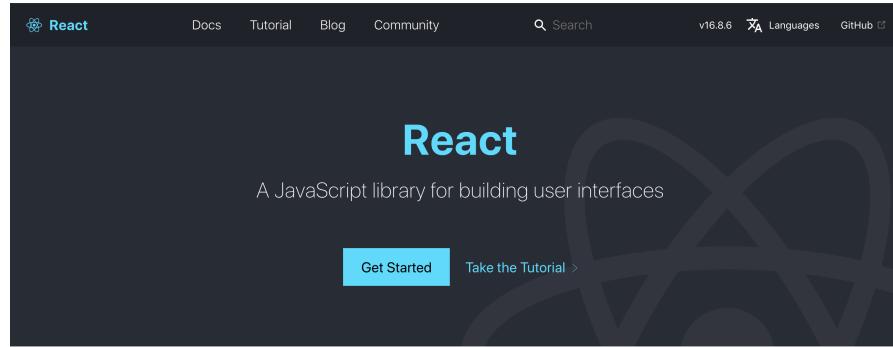
REACT NATIVE

FRAMEWORK PARA CRIAÇÃO DE APLICAÇÕES

Prof. Guilherme A. Madalozzo
Ph.D. em Ciência da Computação



REACT



Declarative

React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.

Declarative views make your code more predictable and easier to debug.

Component-Based

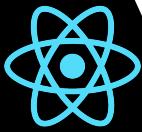
Build encapsulated components that manage their own state, then compose them to make complex UIs.

Since component logic is written in JavaScript instead of templates, you can easily pass rich data through your app and keep state out of the DOM.

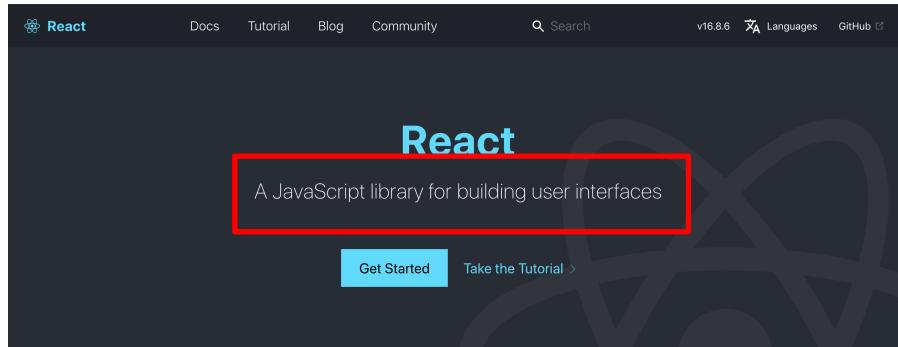
Learn Once, Write Anywhere

We don't make assumptions about the rest of your technology stack, so you can develop new features in React without rewriting existing code.

React can also render on the server using Node and power mobile apps using React Native.



REACT



Uma biblioteca JS
usada para criar
interfaces de
usuário

Declarative

React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.

Declarative views make your code more predictable and easier to debug.

Component-Based

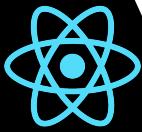
Build encapsulated components that manage their own state, then compose them to make complex UIs.

Since component logic is written in JavaScript instead of templates, you can easily pass rich data through your app and keep state out of the DOM.

Learn Once, Write Anywhere

We don't make assumptions about the rest of your technology stack, so you can develop new features in React without rewriting existing code.

React can also render on the server using Node and power mobile apps using React Native.



REACT



O desenvolvimento
é baseado em
componentes

Declarative

React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.

Declarative views make your code more predictable and easier to debug.

Component-Based

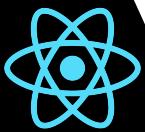
Build encapsulated components that manage their own state, then compose them to make complex UIs.

Since component logic is written in JavaScript instead of templates, you can easily pass rich data through your app and keep state out of the DOM.

Learn Once, Write Anywhere

We don't make assumptions about the rest of your technology stack, so you can develop new features in React without rewriting existing code.

React can also render on the server using Node and power mobile apps using React Native.



REACT NATIVE

React Native trabalha os mesmos conceitos do React



React Native 0.59

Docs Community Blog

Search

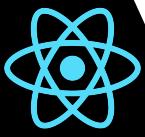
[GitHub](#)

React Native

Build native mobile apps using JavaScript and React

[Get Started](#)

[Learn the Basics](#)



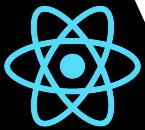
REACT NATIVE

Build native mobile apps using JavaScript and React

React Native lets you build mobile apps using only JavaScript. It uses the same design as React, letting you compose a rich mobile UI using declarative components.

```
import React, {Component} from 'react';
import {Text, View} from 'react-native';

class HelloReactNative extends Component {
  render() {
    return (
      <View>
        <Text>
          If you like React, you'll also like React Native.
        </Text>
        <Text>
          Instead of 'div' and 'span', you'll use native components
          like 'View' and 'Text'.
        </Text>
      </View>
    );
  }
}
```



REACT NATIVE

Build native mobile apps using JavaScript and React

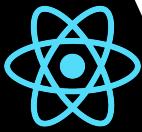
React Native lets you build mobile apps using only JavaScript. It uses the same design as React, letting you compose a rich mobile UI using declarative components.

```
import React, {Component} from 'react';
import {Text, View} from 'react-native';

class HelloReactNative extends Component {
  render() {
    return (
      <View>
        <Text>
          If you like React, you'll also like React Native.
        </Text>
        <Text>
          Instead of 'div' and 'span', you'll use native components
          like 'View' and 'Text'.
        </Text>
      </View>
    );
  }
}
```

Monta uma
interface baseada
em componentes

Configuração do Ambiente



CONFIGURAÇÃO

Dependências na máquina

NodeJS e NPM



Node.js® is a JavaScript runtime built on [Chrome's V8 JavaScript engine](#).

Download for macOS (x64)

10.15.3 LTS

Recommended For Most Users

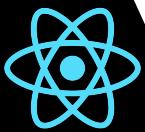
12.1.0 Current

Latest Features

[Other Downloads](#) | [Changelog](#) | [API Docs](#) [Other Downloads](#) | [Changelog](#) | [API Docs](#)

Or have a look at the [Long Term Support \(LTS\) schedule](#).

Sign up for [Node.js Everywhere](#), the official Node.js Monthly Newsletter.

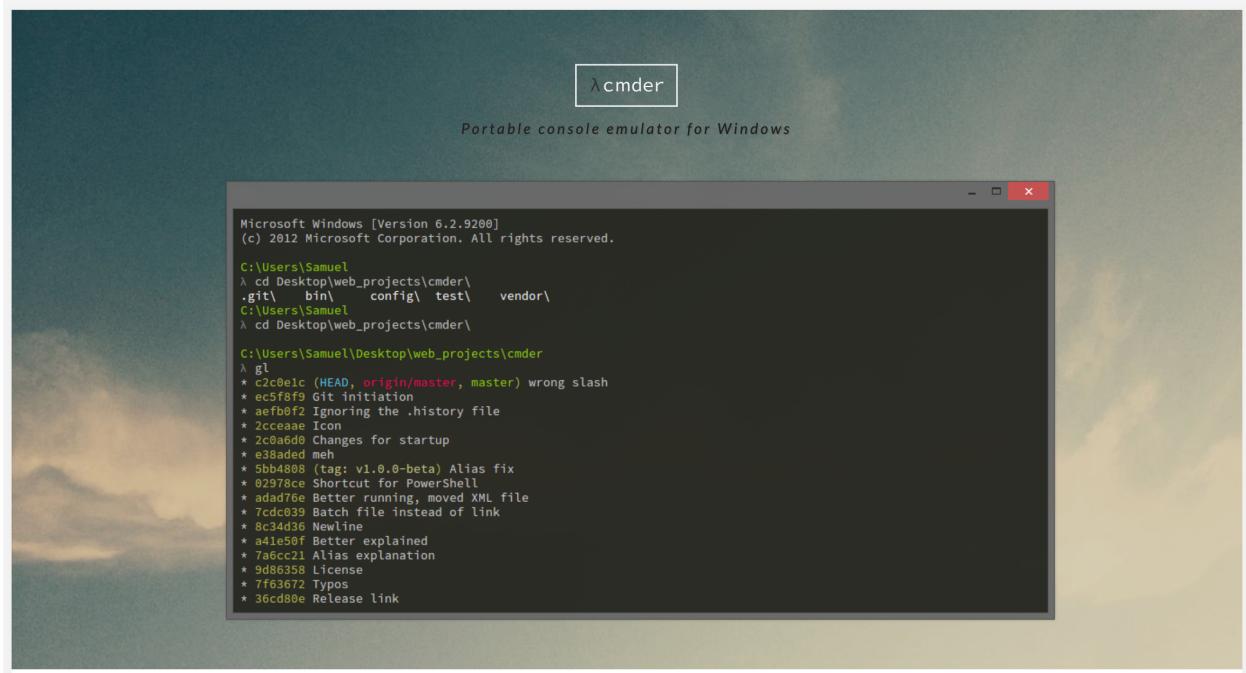


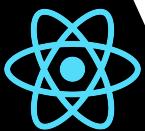
CONFIGURAÇÃO

Dependências na máquina

Caso usar windows

CMDER



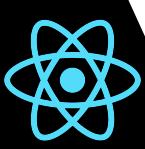


CONFIGURAÇÃO

Dependências na máquina

No terminar, instalar o React-Native e o Expo-CLI

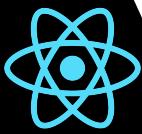
```
x npm install --save react-native  
x npm install expo-cli --global
```



CONFIGURAÇÃO

Dependências no celular

- Não vamos usar AndroidStudio e Xcode



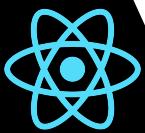
CONFIGURAÇÃO

Dependências na máquina

The screenshot shows the official Sublime Text website. At the top, there's a navigation bar with links for 'Download', 'Buy', 'Support', 'News', and 'Forum'. Below the navigation, a large banner reads 'A sophisticated text editor for code, markup and prose'. It features a 'DOWNLOAD FOR MAC' button, the text 'Sublime Text 3 (Build 3207)', and a 'Changelog' link. To the right of the banner is a small screenshot of the Sublime Text interface. At the bottom, there's a preview of the Mac OS desktop environment showing the Sublime Text application window.

The screenshot shows the official Atom website. On the left, there's a large image of a green alien wearing a space helmet, looking at a glowing circular interface. To the right of the image, the word 'ATOM' is written vertically. Below it, the version '1.36.1' is displayed along with 'Release notes' and 'macOS' (with a note 'For macOS 10.9 or later'). At the bottom right is a prominent yellow 'Download' button.

The screenshot shows the official Visual Studio Code website. The top navigation bar includes links for 'Visual Studio Code', 'Docs', 'Updates', 'Blog', 'API', 'Extensions', and 'FAQ'. There's also a search bar labeled 'Search Docs' and a 'Download' button. A message at the top says 'Version 1.33 is now available! Read about the new features and fixes from March.' Below this, a large dark blue banner features the text 'Code editing. Redefined.' In the center, there's a screenshot of the Visual Studio Code interface showing a file named 'www.ts - node-express-ts' with some TypeScript code. The interface also shows the 'EXTENSIONS' sidebar with several popular extensions listed, including 'C#', 'Python', and 'Node.js'.

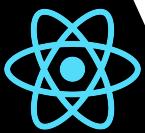


CONFIGURAÇÃO

Dependências no celular

- Não vamos usar **AndroidStudio** e Xcode

Muito pesado,
não vale a pena

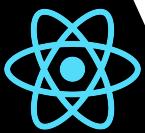


CONFIGURAÇÃO

Dependências no celular

- Não vamos usar AndroidStudio e **Xcode**

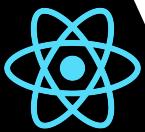
Só funciona em
MacOS e ocupa muito
espaço em disco



CONFIGURAÇÃO

Dependências no celular

- Não vamos usar AndroidStudio e Xcode
- Só usar estas ferramentas caso precise executar o aplicativo em alguma plataforma que não tenha acesso fisicamente
- Usar apenas os emuladores



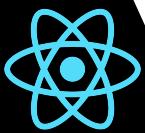
CONFIGURAÇÃO

Dependências no celular

- Não vamos usar AndroidStudio e Xcode
- Vamos debugar a aplicação diretamente no celular



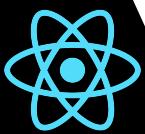
Projeto



REACT NATIVE

Qual projeto vamos desenvolver?

- Vamos montar um projeto para registro e controle de hábitos
 - Controle de Login
 - Cadastro de Hábitos
 - Cadastro de Históricos

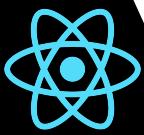


REACT NATIVE

Criando o projeto

```
expo init <nome_do_projeto>
```

```
x expo init MyHabitTimeline
```

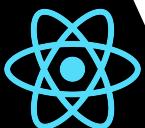


REACT NATIVE

Criando o projeto

```
There is a new version of expo-cli available (2.16.1).  
You are currently using expo-cli 2.6.14  
Run `npm install -g expo-cli` to get the latest version  
? Choose a template: (Use arrow keys)  
❯ blank  
    minimum dependencies to run and an empty root component  
tabs  
several example screens and tabs using react-navigation
```

The diagram illustrates two project templates side-by-side. On the left, a blue rounded rectangle labeled 'Blank' contains a white square with the text '<View />'. On the right, a grey rounded rectangle labeled 'Tab Navigation' contains a white square with a blue house icon and a grey notebook icon.

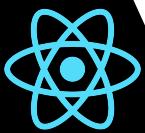


REACT NATIVE

```
[→ Ministrados expo init MyHabitTimeline
There is a new version of expo-cli available (2.16.1).
You are currently using expo-cli 2.6.14
Run `npm install -g expo-cli` to get the latest version
? Choose a template: expo-template-blank
[?] Yarn v1.13.0 found. Use Yarn to install dependencies? Yes
[09:29:02] Extracting project files...
[09:29:03] Customizing project...
[09:29:03] Initialized a git repository.
[09:29:03] Installing dependencies...
yarn install v1.13.0
info No lockfile found.
[1/4] ⚡ Resolving packages...
warning expo > fbemitter > core-js@1.2.7: core-js<2.6.5 is no longer maintained. Please, upgrade to core-js@3 or at least to actual version of core-js@2.
[2/4] 🛒 Fetching packages...
[3/4] 🔗 Linking dependencies...
warning "expo > expo-background-fetch@1.0.0" has unmet peer dependency "expo-task-manager-interface@~1.0.0".
warning "expo > expo-google-sign-in@2.0.0" has incorrect peer dependency "react-native@^0.55.4".
warning "expo > expo-location@2.0.1" has unmet peer dependency "expo-task-manager-interface@~1.0.0".
warning "expo > react-native-reanimated@1.0.0-alpha.11" has incorrect peer dependency "react@16.0.0-alpha.6".
warning "expo > react-native-reanimated@1.0.0-alpha.11" has incorrect peer dependency "react-native@^0.44.1".
warning "expo > expo-asset > url-loader@1.1.2" has unmet peer dependency "webpack@^3.0.0 || ^4.0.0".
[4/4] 🏺 Building fresh packages...
success Saved lockfile.
warning Your current version of Yarn is out of date. The latest version is "1.15.2", while you're on "1.13.0".
info To upgrade, run the following command:
$ curl --compressed -o- -L https://yarnpkg.com/install.sh | bash
★ Done in 71.42s.

Your project is ready at /Users/madalozzo/Dropbox/1-UPF/Cursos/ReactNative/Ministrados/MyHabitTimeline
To get started, you can type:

cd MyHabitTimeline
yarn start
```

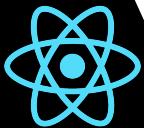


REACT NATIVE

Executando o projeto

```
cd <nome_do_projeto>  
expo start
```

```
→ MyHabitTimeline git:(master) ✘ expo start  
There is a new version of expo-cli available (2.16.1).  
You are currently using expo-cli 2.6.14  
Run `npm install -g expo-cli` to get the latest version  
[16:34:06] Starting project at /Users/madalozzo/Dropbox/1-UPF/Cursos/ReactNative/Ministrados/MyHabitTimeline  
[16:34:06] Expo DevTools is running at http://localhost:19002  
[16:34:06] Opening DevTools in the browser... (press shift-d to disable)
```



REACT NATIVE

Executando o projeto

```
cd <nome_do_projeto>
```

```
expo start
```

```
→ MyHabitTimeline git:(master) ✘ expo start
There is a new version of expo-cli available (2.16.1).
You are currently using expo-cli 2.6.14
Run `npm install -g expo-cli` to get the latest version
[16:34:06] Starting project at /Users/madalozzo/Dropbox/1-UPF/
[16:34:06] Expo DevTools is running at http://localhost:19002
[16:34:06] Opening DevTools in the browser... (press shift-d t
```

Metro Bundler
PROCESS (2) - 10:31:54 AM
LOGGED IN AS GUIMADALOZZO

METRO BUNDLER

INFO Starting Metro Bundler on port 19001.
10:31

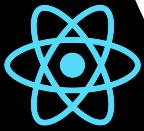
INFO Tunnel ready.
10:31

Run on Android device/emulator
Run on iOS simulator
Send link with email/SMS...
Publish or republish project... ↗

PRODUCTION MODE Tunnel LAN Local

exp://192.168.0.102:19000





REACT NATIVE

Executando o projeto

```
cd <nome_do_projeto>
```

```
expo start
```

```
→ MyHabitTimeline git:(master) ✘ expo start
There is a new version of expo-cli available (2.16.1).
You are currently using expo-cli 2.6.14
Run `npm install -g expo-cli` to get the latest version
[16:34:06] Starting project at /Users/madalozzo/Dropbox/1-UPF/
[16:34:06] Expo DevTools is running at http://localhost:19002
[16:34:06] Opening DevTools in the browser... (press shift-d t
```

Metro Bundler
PROCESS (2) - 10:31:54 AM
LOGGED IN AS GUIMADALOZZO

METRO BUNDLER

INFO Starting Metro Bundler on port 19001.
10:31

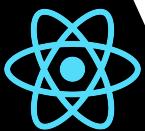
INFO Tunnel ready.
10:31

Run on Android device/emulator
Run on iOS simulator
Send link with email/SMS...
Publish or republish project... ↗

PRODUCTION MODE Tunnel LAN Local

exp://192.168.0.102:19000





REACT NATIVE

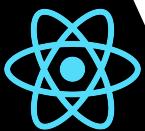
Tipos de conexões do Expo

- Tunnel
 - Executa o expo em redes diferentes
 - Precisa ter usuário cadastrado no expo

CONNECTION



🔗 <exp://pe-wmx.guimadalozzo.myhabitti...>



REACT NATIVE

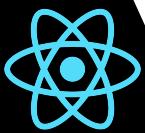
Tipos de conexões do Expo

- LAN (default)
 - Executa o expo na mesma rede do máquina hospedeira

CONNECTION



☞ exp://192.168.0.102:19000



REACT NATIVE

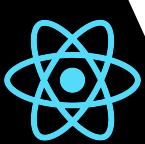
Tipos de conexões do Expo

- Local
 - Execução do hospedeiro e debug na mesma máquina

CONNECTION

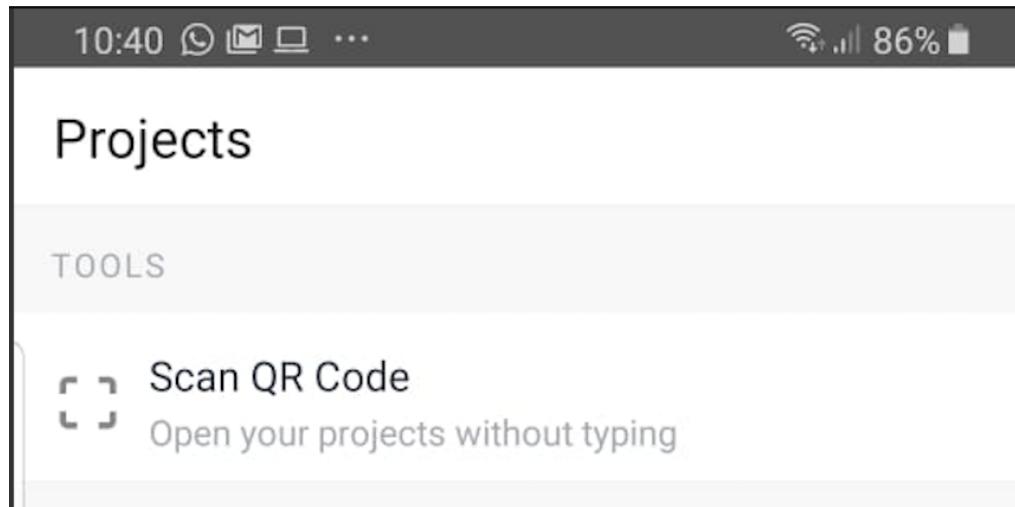


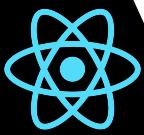
exp://127.0.0.1:19000



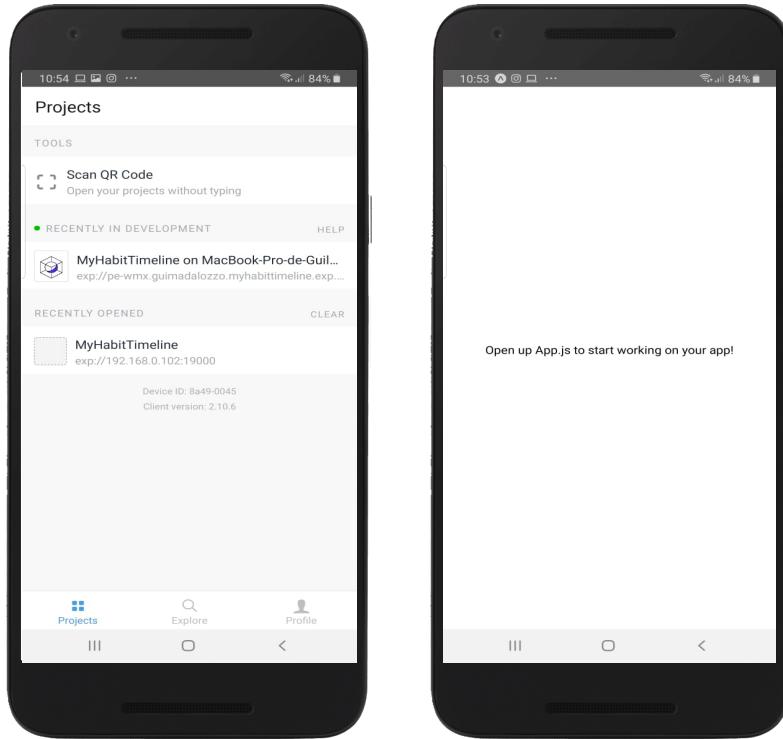
REACT NATIVE

QRCode

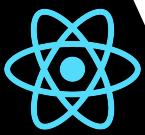




REACT NATIVE



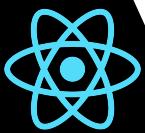
Arquitetura Gerada



REACT NATIVE

◀ MYHABITTIMELINE

- ▶ .expo
- ▶ assets
- ▶ node_modules
- ❖ .gitignore
- { } .watchmanconfig
- JS** App.js
- { } app.json
- JS** babel.config.js
- { } package.json
- 🐈 yarn.lock



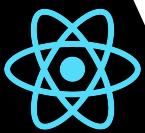
REACT NATIVE

MYHABITTIMELINE

- .expo
- assets
- node_modules
- .gitignore
- { } .watchmanconfig
- JS** App.js
- { } app.json
- JS** babel.config.js
- { } package.json
- yarn.lock

Pasta com arquivos de configuração do aplicativo Expo

- .expo
- { } packager-info.json
- { } settings.json



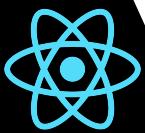
REACT NATIVE

MYHABITTIMELINE

- ▶ .expo
- ▶ assets
- ▶ node_modules
- ❖ .gitignore
- { } .watchmanconfig
- JS** App.js
- { } app.json
- JS** babel.config.js
- { } package.json
- 🔒 yarn.lock

Pasta com arquivos
de configuração do
ícone e splash
screen

- ▶ assets
- ☒ icon.png
- ☒ splash.png



REACT NATIVE

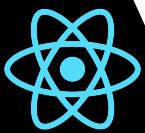
◀ MYHABITTIMELINE

- ▶ .expo
- ▶ assets
- ▶ node_modules
- ▷ .gitignore
- { } .watchmanconfig
- JS** App.js
- { } app.json
- JS** babel.config.js
- { } package.json
- yarn.lock

Pasta com arquivos
de dependências
do projeto

◀ node_modules

- ▶ .bin
- ▶ .cache
- ▶ @babel
- ▶ @expo
- ▶ @types
- ▶ abbrev
- ▶ absolute-path
- ▶ accents



REACT NATIVE

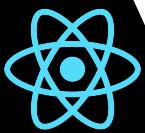
MYHABITTIMELINE

- ▶ .expo
- ▶ assets
- ▶ node_modules
- ❖ .gitignore
- { } .watchmanconfig
- JS App.js
- { } app.json
- JS babel.config.js
- { } package.json
- 📦 yarn.lock

Arquivo com conteúdo que deve ser ignorado no versionamento do git

❖ .gitignore

```
1 node_modules/**/*  
2 .expo/*  
3 npm-debug.*  
4 *.jks  
5 *.p12  
6 *.key  
7 *.mobileprovision
```



REACT NATIVE

MYHABITTIMELINE

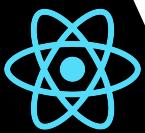
- ▶ .expo
- ▶ assets
- ▶ node_modules
- ❖ .gitignore
- {} .watchmanconfig
- JS App.js
- {} app.json
- JS babel.config.js
- {} package.json
- yc yarn.lock

Arquivo de configuração do Watchman
Ferramenta responsável pelo live reload



Watchman

A file watching service



REACT NATIVE

MYHABITTIMELINE

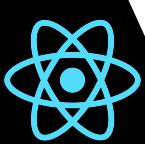
- ▶ .expo
- ▶ assets
- ▶ node_modules
- ▶ .gitignore
- { } .watchmanconfig
- JS** App.js
- { } app.json
- JS** babel.config.js
- { } package.json
- 🐈 yarn.lock

Principal arquivo
do projeto.
Primeiro arquivo a
ser executado

Não devemos
mudar ele de lugar
e nem renomear

JS App.js

```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 export default class App extends React.Component {
5   render() {
6     return (
7       <View style={styles.container}>
8         <Text>Open up App.js to start working on your app!</Text>
9       </View>
10    );
11  }
12}
13
14 const styles = StyleSheet.create({
15   container: {
16     flex: 1,
17     backgroundColor: '#fff',
18     alignItems: 'center',
19     justifyContent: 'center',
20   },
21});
```



REACT NATIVE

MYHABITTIMELINE

- ▶ .expo
- ▶ assets
- ▶ node_modules
- ▶ .gitignore
- {} .watchmanconfig
- JS App.js**
- {} app.json
- JS babel.config.js**
- {} package.json
- >yarn.lock

Principal arquivo
do projeto.
Primeiro arquivo a
ser executado

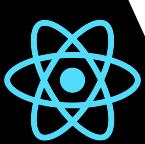
Não devemos
mudar ele de lugar
e nem renomear

JS App.js

```
1 import React
2 import { StyleSheet, View } from 'react-native'
3
4 export default function App() {
5   return (
6     <View style={styles.container}>
7       <Text>Hello world!</Text>
8     </View>
9   );
10 }
11
12 const styles = StyleSheet.create({
13   container: {
14     flex: 1,
15     backgroundColor: '#fff',
16     alignItems: 'center',
17     justifyContent: 'center',
18   },
19 });
20
21 
```

10:53 84% 84%

-native';
ent {
on your app!</Text>



REACT NATIVE

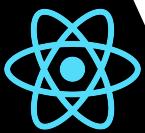
MYHABITTIMELINE

- ▶ .expo
- ▶ assets
- ▶ node_modules
- ◆ .gitignore
- { } .watchmanconfig
- JS App.js**
- { } app.json**
- JS babel.config.js**
- { } package.json
- 🐈 yarn.lock

Arquivo de configuração do projeto

{ } app.json

```
1 {  
2   "expo": {  
3     "name": "MyHabitTimeline",  
4     "slug": "MyHabitTimeline",  
5     "privacy": "public",  
6     "sdkVersion": "32.0.0",  
7     "platforms": [  
8       "ios",  
9       "android"  
10    ],  
11    "version": "1.0.0",  
12    "orientation": "portrait",  
13    "icon": "./assets/icon.png",  
14    "splash": {  
15      "image": "./assets/splash.png",  
16      "resizeMode": "contain",  
17      "backgroundColor": "#ffffff"  
18    },  
19    "updates": {  
20      "fallbackToCacheTimeout": 0  
21    },  
22    "assetBundlePatterns": [  
23      "**/*"  
24    ],  
25    "ios": {  
26      "supportsTablet": true  
27    }  
28  }  
29}
```

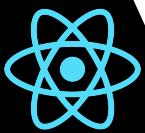


REACT NATIVE

◀ MYHABITTIMELINE

- ▶ .expo
- ▶ assets
- ▶ node_modules
- ❖ .gitignore
- { } .watchmanconfig
- JS** App.js
- { } app.json
- JS** babel.config.js
- { } package.json
- 🐈 yarn.lock

Arquivo de
configuração
babel, transpilador
javascript



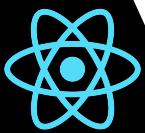
REACT NATIVE

MYHABITTIMELINE

- ▶ .expo
- ▶ assets
- ▶ node_modules
- ❖ .gitignore
- { } .watchmanconfig
- JS App.js
- { } app.json
- JS babel.config.js
- { } package.json
- YAML yarn.lock

```
(package.json) ●  
1  {  
2    "main": "node_modules/expo/AppEntry.js",  
3    "scripts": {  
4      "start": "expo start",  
5      "android": "expo start --android",  
6      "ios": "expo start --ios",  
7      "eject": "expo eject"  
8    },  
9    "dependencies": {  
10      "expo": "^32.0.0",  
11      "react": "16.5.0",  
12      "react-native": "https://github.com/expo/react-native/archive/sdk-32.0.0.tar.gz"  
13    },  
14    "devDependencies": {  
15      "babel-preset-expo": "5.0.0"  
16    },  
17    "private": true  
18  }
```

Arquivo com configuração de scripts do projeto e controle de dependências



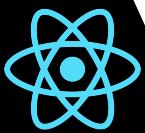
REACT NATIVE

◀ MYHABITTIMELINE

- ▶ .expo
- ▶ assets
- ▶ node_modules
- ❖ .gitignore
- { } .watchmanconfig
- JS** App.js
- { } app.json
- JS** babel.config.js
- { } package.json
- Yarn** yarn.lock

Arquivo com lock
de dados do
gerenciador de
pacotes do projeto:
Yarn

Entendendo o App.js



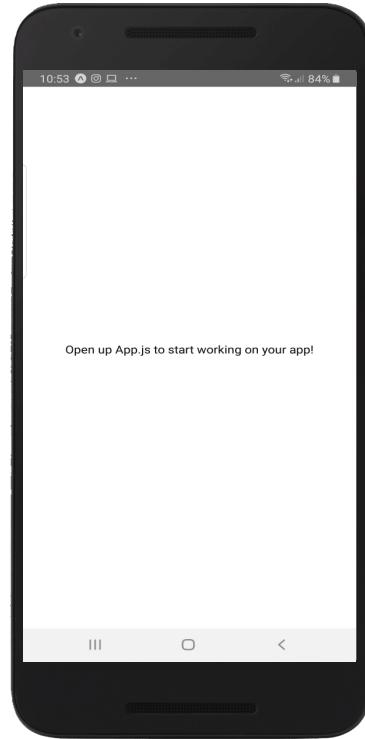
APP JS

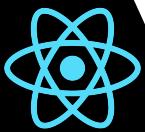
App.js está montando e renderizando nossa primeira tela

JS App.js

```
1  import React from 'react';
2    import { StyleSheet, Text, View } from 'react-native';
3
4  export default class App extends React.Component {
5    render() {
6      return (
7        <View style={styles.container}>
8          <Text>Open up App.js to start working on your app!</Text>
9        </View>
10     );
11   }
12 }
13
14 const styles = StyleSheet.create({
15   container: {
16     flex: 1,
17     backgroundColor: '#fff',
18     alignItems: 'center',
19     justifyContent: 'center',
20   },
21 });


```





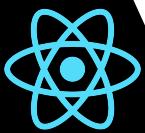
APP JS

App.js está montando e renderizando nossa primeira tela

```
JS App.js      x
1  import React from 'react';
2  import { StyleSheet, Text, View } from 'react-native';
3
4  export default class App extends React.Component {
5    render() {
6      return (
7        <View style={styles.container}>
8          <Text>Open up App.js to start working on your app!</Text>
9        </View>
10     );
11   }
12 }
13
14 const styles = StyleSheet.create({
15   container: {
16     flex: 1,
17     backgroundColor: '#fff',
18     alignItems: 'center',
19     justifyContent: 'center',
20   },
21 });


```

Sempre vamos importar ao projeto o React, para podermos criar os componentes

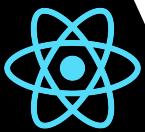


APP JS

App.js está montando e renderizando nossa primeira tela

```
JS App.js      ×  
1  import React from 'react';  
2  import { StyleSheet, Text, View } from 'react-native';  
3  
4  export default class App extends React.Component {  
5    render() {  
6      return (  
7        <View style={styles.container}>  
8          <Text>Open up App.js to start working on your app!</Text>  
9        </View>  
10     );  
11   }  
12 }  
13  
14 const styles = StyleSheet.create({  
15   container: {  
16     flex: 1,  
17     backgroundColor: '#fff',  
18     alignItems: 'center',  
19     justifyContent: 'center',  
20   },  
21});
```

Todos os componentes que utilizarmos na página devemos importar



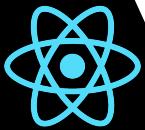
APP JS

App.js está montando e renderizando nossa primeira tela

```
JS App.js      x
1  import React from 'react';
2  import { StyleSheet, Text, View } from 'react-native';
3
4  export default class App extends React.Component {
5    render() {
6      return (
7        <View style={styles.container}>
8          <Text>Open up App.js to start working on your app!</Text>
9        </View>
10     );
11   }
12 }
13
14 const styles = StyleSheet.create({
15   container: {
16     flex: 1,
17     backgroundColor: '#fff',
18     alignItems: 'center',
19     justifyContent: 'center',
20   },
21 });


```

Todos os componentes que utilizarmos na página devemos importar



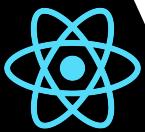
APP JS

App.js está montando e renderizando nossa primeira tela

```
JS App.js      x
1  import React from 'react';
2  import { StyleSheet, Text, View } from 'react-native';
3
4  export default class App extends React.Component {
5    render() {
6      return (
7        <View style={styles.container}>
8          <Text>Open up App.js to start working on your app!</Text>
9        </View>
10     );
11   }
12 }
13
14 const styles = StyleSheet.create({
15   container: {
16     flex: 1,
17     backgroundColor: '#fff',
18     alignItems: 'center',
19     justifyContent: 'center',
20   },
21 });


```

Todos os componentes que utilizarmos na página devemos importar



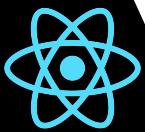
APP JS

App.js está montando e renderizando nossa primeira tela

```
JS App.js      x
1  import React from 'react';
2  import { StyleSheet, Text, View } from 'react-native';
3
4  export default class App extends React.Component {
5    render() {
6      return (
7        <View style={styles.container}>
8          <Text>Open up App.js to start working on your app!</Text>
9        </View>
10     );
11   }
12 }
13
14 const styles = StyleSheet.create({
15   container: {
16     flex: 1,
17     backgroundColor: '#fff',
18     alignItems: 'center',
19     justifyContent: 'center',
20   },
21 });


```

Todos os componentes que utilizarmos na página devemos importar



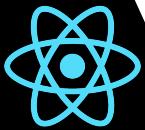
APP JS

App.js está montando e renderizando nossa primeira tela

```
JS App.js      x
1  import React from 'react';
2  import { StyleSheet, Text, View } from 'react-native';
3
4  export default class App extends React.Component {
5    render() {
6      return (
7        <View style={styles.container}>
8          <Text>Open up App.js to start working on your app!</Text>
9        </View>
10     );
11   }
12 }
13
14 const styles = StyleSheet.create({
15   container: {
16     flex: 1,
17     backgroundColor: '#fff',
18     alignItems: 'center',
19     justifyContent: 'center',
20   },
21 });


```

Todos os componentes que utilizarmos na página devemos importar



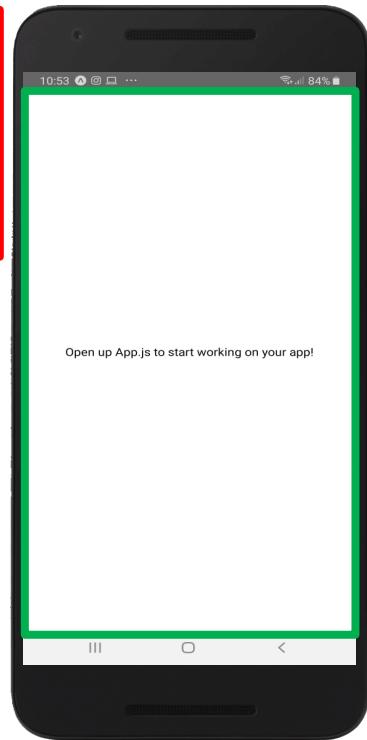
APP JS

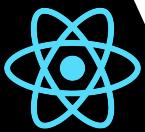
App.js está montando e renderizando nossa primeira tela

```
JS App.js ×  
1  import React from 'react';  
2    import { StyleSheet, Text, View } from 'react-native';  
3  
4  export default class App extends React.Component {  
5    render() {  
6      return (  
7        <View style={styles.container}>  
8          <Text>Open up App.js to start working on your app!</Text>  
9        </View>  
10     );  
11   }  
12 }  
13  
14 const styles = StyleSheet.create({  
15   container: {  
16     flex: 1,  
17     backgroundColor: '#fff',  
18     alignItems: 'center',  
19     justifyContent: 'center',  
20   },  
21});
```

Render()

Retorna a renderização de componentes na tela da página





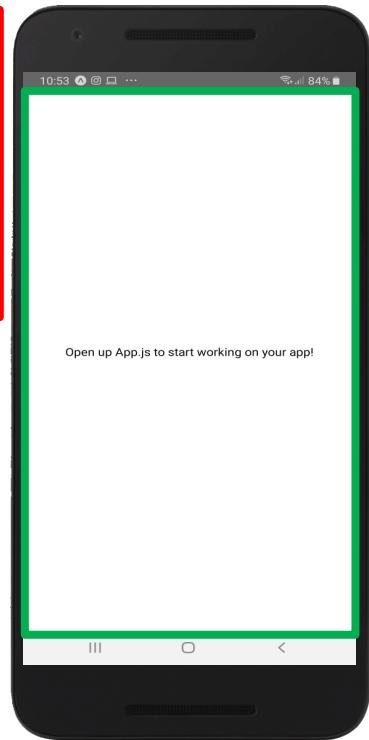
APP JS

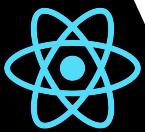
App.js está montando e renderizando nossa primeira tela

```
JS App.js ×  
1  import React from 'react';  
2    import { StyleSheet, Text, View } from 'react-native';  
3  
4  export default class App extends React.Component {  
5    render() {  
6      return (  
7        <View style={styles.container}>  
8          <Text>Open up App.js to start working on your app!</Text>  
9        </View>  
10     );  
11   }  
12 }  
13  
14 const styles = StyleSheet.create({  
15   container: {  
16     flex: 1,  
17     backgroundColor: '#fff',  
18     alignItems: 'center',  
19     justifyContent: 'center',  
20   },  
21});
```

View

Obrigatoriamente todos os componentes retornados devem estar inseridos em um único componente "pai"





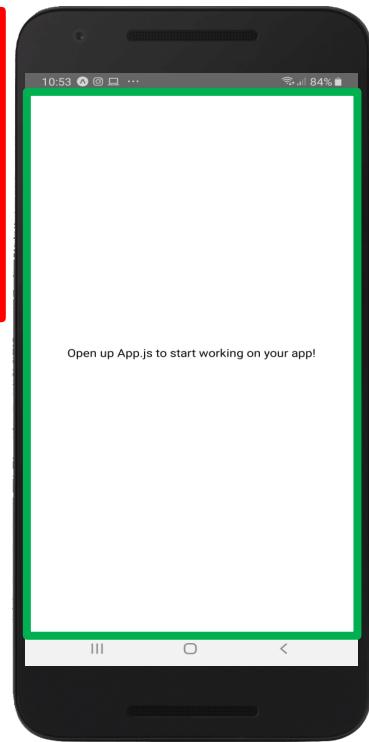
APP JS

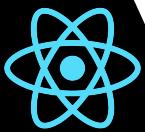
App.js está montando e renderizando nossa primeira tela

```
JS App.js ×  
1  import React from 'react';  
2    import { StyleSheet, Text, View } from 'react-native';  
3  
4  export default class App extends React.Component {  
5    render() {  
6      return (  
7        <View style={styles.container}>  
8          <Text>Open up App.js to start working on your app!</Text>  
9        </View>  
10     );  
11   }  
12 }  
13  
14 const styles = StyleSheet.create({  
15   container: {  
16     flex: 1,  
17     backgroundColor: '#fff',  
18     alignItems: 'center',  
19     justifyContent: 'center',  
20   },  
21});
```

View

O componente viu irá possuir o estilo "container"

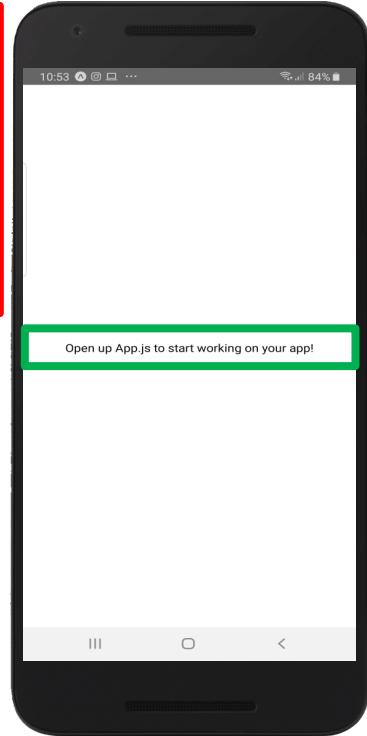


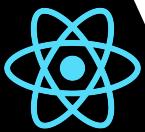


APP JS

App.js está montando e renderizando nossa primeira tela

```
JS App.js ×  
1  import React from 'react';  
2    import { StyleSheet, Text, View } from 'react-native';  
3  
4  export default class App extends React.Component {  
5    render() {  
6      return (  
7        <View style={styles.container}>  
8          <Text>Open up App.js to start working on your app!</Text>  
9        </View>  
10     );  
11   }  
12 }  
13  
14 const styles = StyleSheet.create({  
15   container: {  
16     flex: 1,  
17     backgroundColor: '#fff',  
18     alignItems: 'center',  
19     justifyContent: 'center',  
20   },  
21});
```





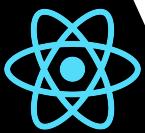
APP JS

App.js está montando e renderizando nossa primeira tela

```
JS App.js ×  
1  import React from 'react';  
2    import { StyleSheet, Text, View } from 'react-native';  
3  
4  export default class App extends React.Component {  
5    render() {  
6      return (  
7        <View style={styles.container}>  
8          <Text>Open up App.js to start working on your app!</Text>  
9        </View>  
10     );  
11   }  
12 }  
13  
14 const styles = StyleSheet.create({  
15   container: {  
16     flex: 1,  
17     backgroundColor: '#fff',  
18     alignItems: 'center',  
19     justifyContent: 'center',  
20   },  
21});
```

Registrarmos a classe App como um componente, React, do projeto

Modificando o App.js

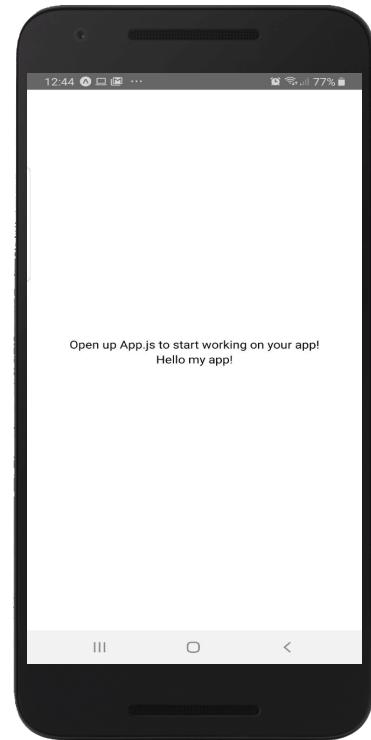


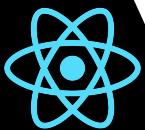
APP JS

App.js está montando e renderizando nossa primeira tela

```
JS App.js  x
1  import React from 'react';
2  import { StyleSheet, Text, View } from 'react-native';
3
4  export default class App extends React.Component {
5    render() {
6      return (
7        <View style={styles.container}>
8          <Text>Open up App.js to start working on your app!</Text>
9
10         <Text>Hello my app!</Text>
11       </View>
12     );
13   }
14 }
15
16 const styles = StyleSheet.create({
17   container: {
18     flex: 1,
19     backgroundColor: '#fff',
20     alignItems: 'center',
21     justifyContent: 'center',
22   },
23 });


```





APP.JS

LEMBRAM ???

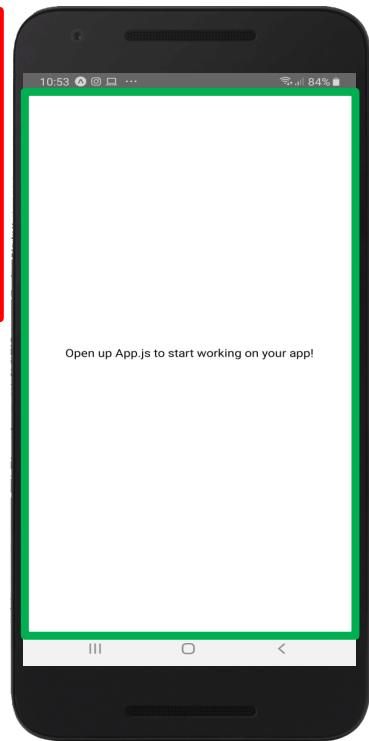
o e renderizando nossa primeira tela

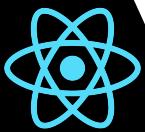
```
1  import React from 'react';
2  import { StyleSheet, Text, View } from 'react-native';
3
4  export default class App extends React.Component {
5    render() {
6      return (
7        <View style={styles.container}>
8          <Text>Open up App.js to start working on your app!</Text>
9        </View>
10     );
11   }
12 }
13
14 const styles = StyleSheet.create({
15   container: {
16     flex: 1,
17     backgroundColor: '#fff',
18     alignItems: 'center',
19     justifyContent: 'center',
20   },
21 });


```

View

Obrigatoriamente todos os componentes retornados devem estar inseridos em um único componente "pai"



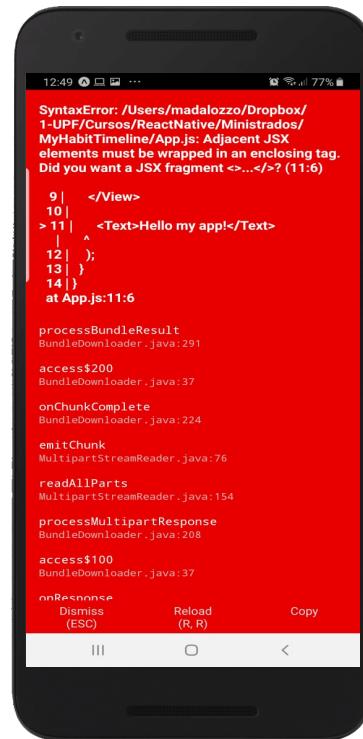


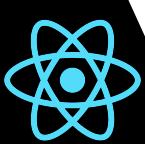
APP JS

App.js está montando e renderizando nossa primeira tela

```
JS App.js x

1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 export default class App extends React.Component {
5   render() {
6     return (
7       <View style={styles.container}>
8         <Text>Open up App.js to start working on your app!</Text>
9       </View>
10
11      <Text>Hello my app!</Text>
12    );
13  }
14}
15
16 const styles = StyleSheet.create({
17   container: {
18     flex: 1,
19     backgroundColor: '#fff',
20     alignItems: 'center',
21     justifyContent: 'center',
22   },
23});
```





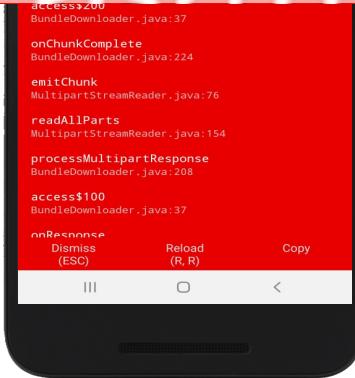
APP JS

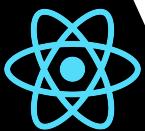
App.js está montando e renderizando nossa primeira tela

```
JS App.js x
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
```



```
11 |     <Text>Hello my app!</Text>
12 |   );
13 |
14 }
15
16 const styles = StyleSheet.create({
17   container: {
18     flex: 1,
19     backgroundColor: '#fff',
20     alignItems: 'center',
21     justifyContent: 'center',
22   },
23 });
```



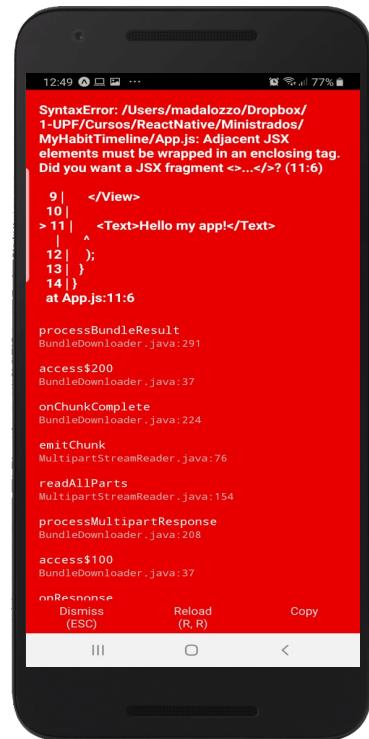


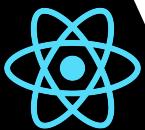
APP.JS

Voltamos para a versão anterior

o e renderizando nossa primeira tela

```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 export default class App extends React.Component {
5   render() {
6     return (
7       <View style={styles.container}>
8         <Text>Open up App.js to start working on your app!</Text>
9
10        <Text>Hello my app!</Text>
11      </View>
12    );
13  }
14}
15
16 const styles = StyleSheet.create({
17   container: {
18     flex: 1,
19     backgroundColor: '#ffff',
20     alignItems: 'center',
21     justifyContent: 'center',
22   },
23});
```



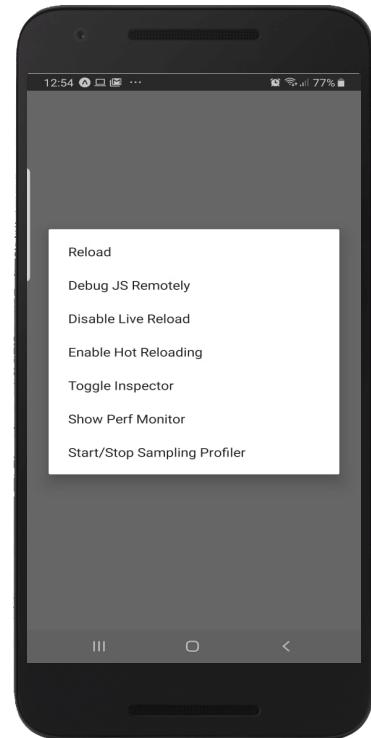


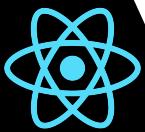
APP.JS

Se a tela não atualizar,
chacoalhe o celular

```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 export default class App extends React.Component {
5   render() {
6     return (
7       <View style={styles.container}>
8         <Text>Open up App.js to start working on your app!</Text>
9
10        <Text>Hello my app!</Text>
11      </View>
12    );
13  }
14}
15
16 const styles = StyleSheet.create({
17   container: {
18     flex: 1,
19     backgroundColor: '#fff',
20     alignItems: 'center',
21     justifyContent: 'center',
22   },
23});
```

do nossa primeira tela

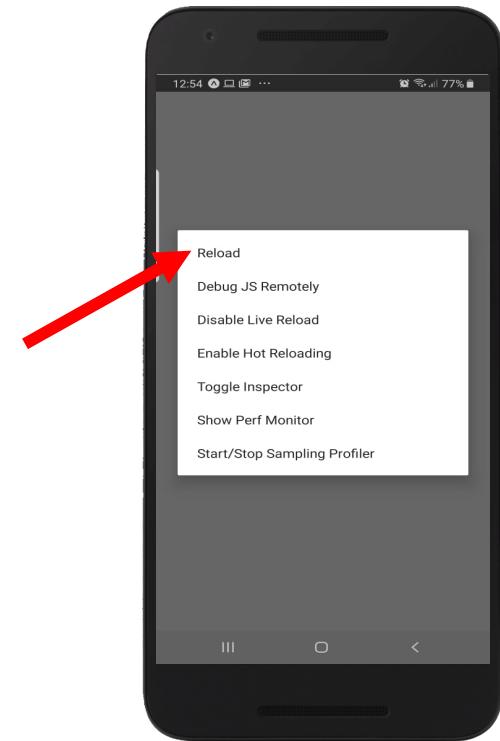


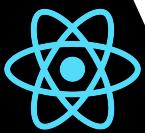


Reload atualiza a tela

do nossa primeira tela

```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 export default class App extends React.Component {
5   render() {
6     return (
7       <View style={styles.container}>
8         <Text>Open up App.js to start working on your app!</Text>
9
10        <Text>Hello my app!</Text>
11      </View>
12    );
13  }
14}
15
16 const styles = StyleSheet.create({
17   container: {
18     flex: 1,
19     backgroundColor: '#fff',
20     alignItems: 'center',
21     justifyContent: 'center',
22   },
23});
```

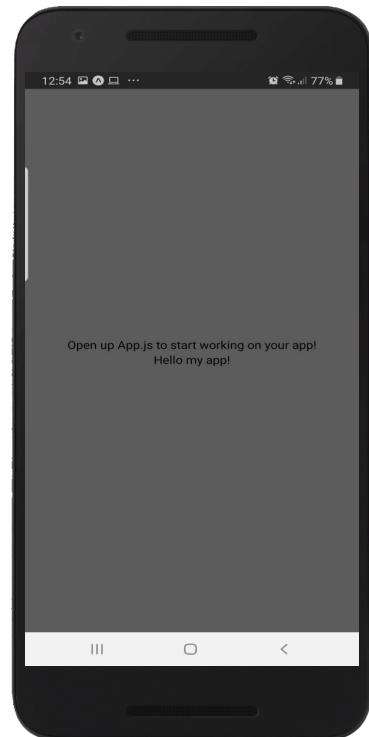


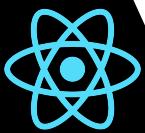


APP.JS

Vamos mudar a cor de fundo da página

```
JS App.js      ×  
1 import React from 'react';  
2 import { StyleSheet, Text, View } from 'react-native';  
3  
4 export default class App extends React.Component {  
5   render() {  
6     return (  
7       <View style={styles.container}>  
8         <Text>Open up App.js to start working on your app!</Text>  
9         <Text>Hello my app!</Text>  
10      </View>  
11    );  
12  }  
13}  
14  
15 const styles = StyleSheet.create({  
16   container: {  
17     flex: 1,  
18     backgroundColor: '#5C5C5C',  
19     alignItems: 'center',  
20     justifyContent: 'center',  
21   },  
22});
```

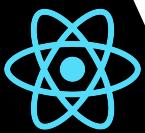




Vamos criar um novo estilo para o texto

```
JS App.js ×

15  const styles = StyleSheet.create({
16    container: {
17      flex: 1,
18      backgroundColor: '#5C5C5C',
19      alignItems: 'center',
20      justifyContent: 'center',
21    },
22    titulo: {
23      color: "red",
24      fontSize: 16,
25    },
26    conteudo: {
27      color: "white",
28      fontSize: 12,
29    },
30  });
31});
```



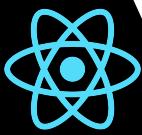
APP JS

Vamos criar um novo estilo para o texto

```
JS App.js      x
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 export default class App extends React.Component {
5   render() {
6     return (
7       <View style={styles.container}>
8         <Text style={styles.titulo}>MyHabitTimeline!</Text>
9         <Text style={styles.conteudo}>Controle de hábitos!</Text>
10      </View>
11    );
12  }
13}
14
15 const styles = StyleSheet.create({
```

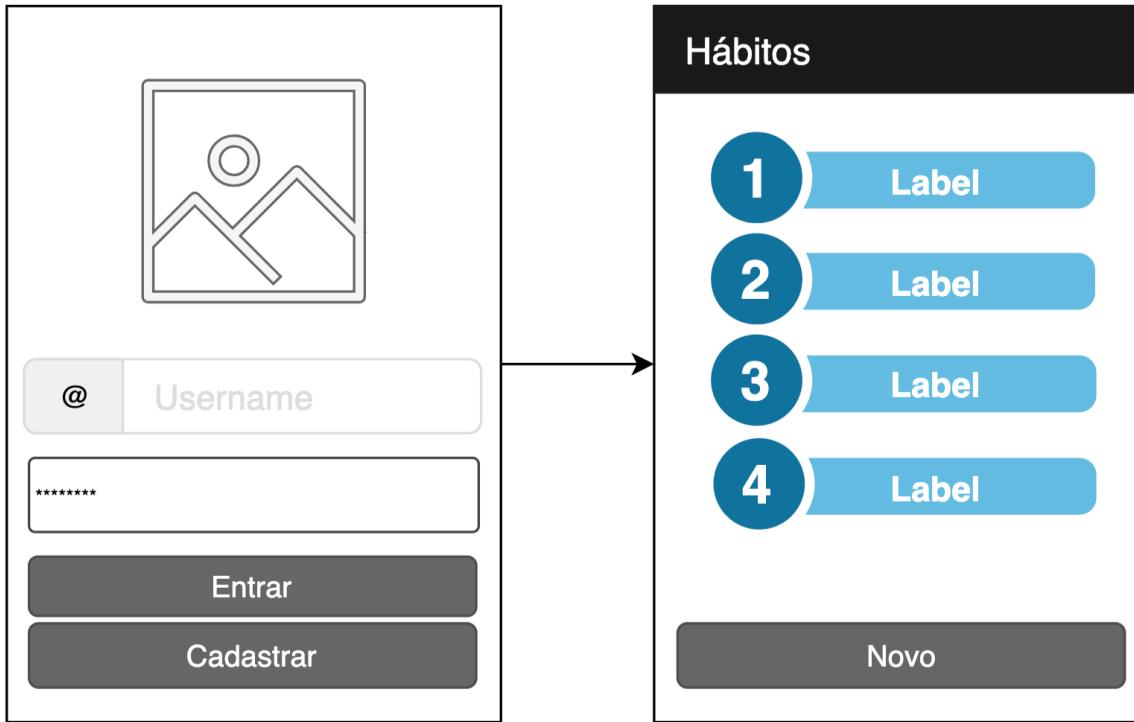


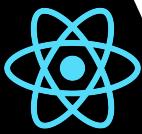
Pensando no nosso projeto



REACT NATIVE

Agora, vamos pensar o montar duas telas parecidas com essas

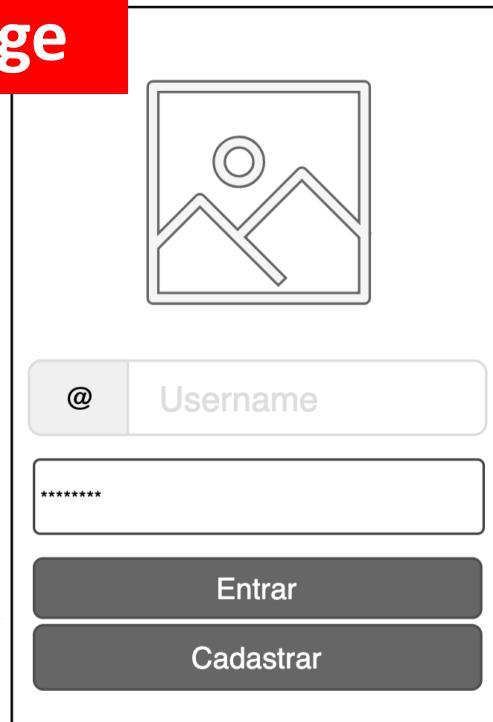




REACT NATIVE

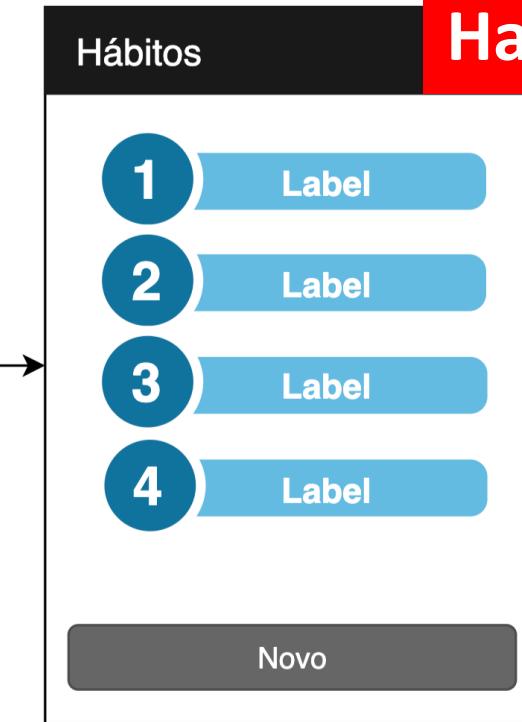
Agora, vamos pensar o montar duas telas parecidas com essas

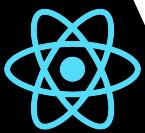
LoginPage



Hábitos

HabitosPage

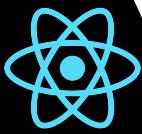




REACT NATIVE

Vamos limpar nosso App.js deixando apenas a estrutura

```
JS App.js ×  
1 import React from 'react';  
2 import { StyleSheet, Text, View } from 'react-native';  
3  
4 export default class App extends React.Component {  
5   render() {  
6     return (  
7       <View>  
8         <Text>MyHabitTimeline!</Text>  
9       </View>  
10    );  
11  }  
12}
```



REACT NATIVE

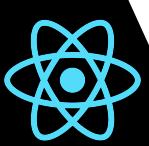
Agora criaremos uma pasta "src" para hospedarmos os códigos da aplicação

E, como o React é baseado em componentes, vamos criar a pasta "components"

Também, vamos criar a pasta "pages" para armazenar as páginas da aplicação

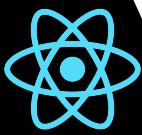
```
MYHABITTIMELINE
  ▶ .expo
  ▶ assets
  ▶ node_modules
  ▲ src
    ▲ components
    ▶ pages
  ◇ .gitignore
  { } .watchmanconfig
  JS App.js
  { } app.json
  JS babel.config.js
  { } package.json
  🔍 yarn.lock
```

The screenshot shows a file explorer interface with a dark theme. A red rectangular box highlights the 'src' directory, which contains 'components' and 'pages'. Other files and directories visible include '.expo', 'assets', 'node_modules', '.gitignore', '.watchmanconfig', 'App.js', 'app.json', 'babel.config.js', 'package.json', and 'yarn.lock'.



MYHABIT TIMELINE

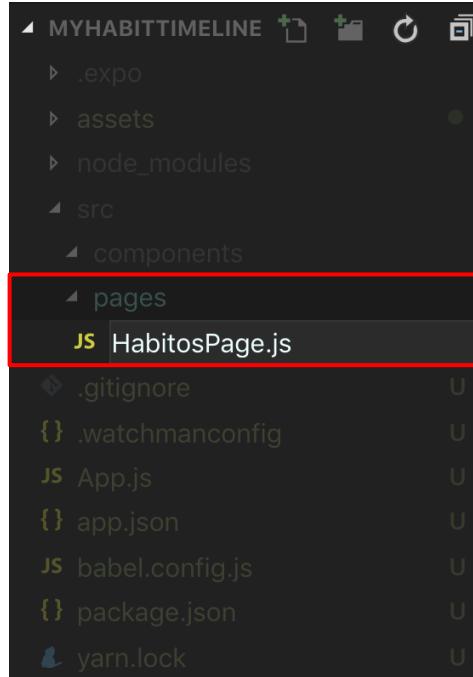
▶ .expo	
▶ assets	
▶ node_modules	
▶ src	darmos os códigos da aplicação
▶ components	, vamos criar a pasta "components"
▶ pages	
▶ .gitignore	U armazenar as páginas da aplicação
{ } .watchmanconfig	U
JS App.js	U
{ } app.json	U
JS babel.config.js	U
{ } package.json	U
YAML yarn.lock	U

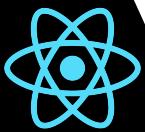


REACT NATIVE

Dentro de src/pages vamos criar a página que listará os hábitos

HabitosPage.js



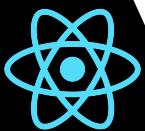


REACT NATIVE

Copiamos o conteúdo de App.js para HabitosPage.js alterando o nome do componente

JS HabitosPage.js ●

```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 export default class HabitosPage extends React.Component {
5   render() {
6     return (
7       <View>
8         <Text>Habitos Page!</Text>
9       </View>
10    );
11  }
12 }
```



REACT NATIVE

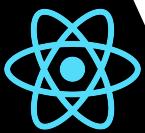
O App.js vamos alterar importando o HabitosPage.js

JS App.js

X

JS HabitosPage.js

```
1 import HabitosPage from './src/pages/HabitosPage';
2
3 export default HabitosPage;
```

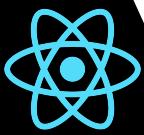


REACT NATIVE

Executamos este passo para que possamos criar e hospedar páginas da aplicação no local correto

Também, para podermos trabalhar com navegação entre páginas

Navegação

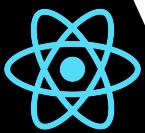


REACT NATIVE

Para trabalhar com navegação vamos instalar uma biblioteca do react
react-navigation

Antes, vamos instalar o gerenciador de dependencias do npm, o yarn

```
x npm install --global yarn
```

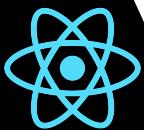


REACT NATIVE

Para trabalhar com navegação vamos instalar uma biblioteca do react
react-navigation

Agora sim, instalaremos o react-navigation

```
x yarn add react-navigation
```



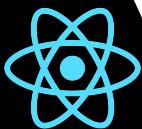
REACT NATIVE

Vamos começar a usar a biblioteca de navegação entre páginas

App.js

Importamos duas funções da navegação e criamos duas constantes para instanciar cada uma

```
JS App.js      JS Habitospage.js
1 import {createAppContainer, createStackNavigator} from 'react-navigation';
2 import Habitospage from './src/pages/Habitospage';
3
4 const appNavigator = createStackNavigator({
5
6 });
7
8 const AppContainer = createAppContainer(appNavigator);
9
10 export default Habitospage;
```



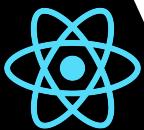
REACT NATIVE

Vamos começar a usar a biblioteca de navegação entre páginas

App.js

AppContainer é responsável por gerenciar o estado da aplicação e controlar a navegação

```
JS App.js  ✘ JS Habitospage.js
1 import {createAppContainer, createStackNavigator } from 'react-navigation';
2 import Habitospage from './src/pages/Habitospage';
3
4 const appNavigator = createStackNavigator({
5
6 });
7
8 const AppContainer = createAppContainer(appNavigator);
9
10 export default Habitospage;
```



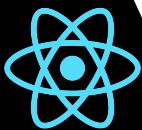
REACT NATIVE

Vamos começar a usar a biblioteca de navegação entre páginas

App.js

StackNavigator é responsável por disponibilizar uma forma de transitar uma página a outra

```
JS App.js  ✘ JS Habitospage.js
1 import {createAppContainer, createStackNavigator} from 'react-navigation';
2 import Habitospage from './src/pages/Habitospage';
3
4 const appNavigator = createStackNavigator({
5
6 });
7
8 const AppContainer = createAppContainer(appNavigator);
9
10 export default Habitospage;
```



REACT NATIVE

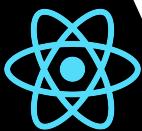
Vamos começar a usar a biblioteca de navegação entre páginas

App.js

Criamos um StackNavigator para permitir que uma página consiga transitar para o HabitosPage

```
JS App.js      ● JS HabitosPage.js

1  import {createAppContainer, createStackNavigator } from 'react-navigation';
2  import HabitosPage from './src/pages/HabitosPage';
3
4  const appNavigator = createStackNavigator({
5    'Main': {
6      screen: HabitosPage
7    }
8  );
9
10 const AppContainer = createAppContainer(appNavigator);
11
12 export default AppContainer;
```



REACT NATIVE

Vamos começar a usar a biblioteca de navegação entre páginas

App.js

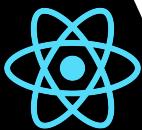
Alteramos o export para ao invés de executar o HabitosPage executa o AppContainer

Stack é uma pilha, o primeiro elemento dela é o primeiro a ser executado pelo AppNavigator

JS App.js

● JS HabitosPage.js

```
1 import {createAppContainer, createStackNavigator } from 'react-navigation';
2 import HabitosPage from './src/pages/HabitosPage';
3
4 const appNavigator = createStackNavigator({
5   'Main': {
6     screen: HabitosPage
7   }
8 });
9
10 const AppContainer = createAppContainer(appNavigator);
11
12 export default AppContainer;
```



REACT NATIVE

Vamos começar a usar a biblioteca de navegação entre páginas

App.js

Alteramos o export para ao invés de executar o HabitosPage executa o AppContainer

Stack é uma pilha, o primeiro elemento dela é o primeiro a ser executado pelo AppNavigator

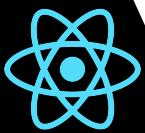
JS App.js

JS HabitosPage

Então, esta alteração, na prática, não muda o resultado

```
1 import {createAppContainer} from 'react-navigation';
2 import HabitosPage from './screens/HabitosPage';
3
4 const appNavigator = createStackNavigator({
5   'Main': {
6     screen: HabitosPage
7   }
8 });
9
10 const AppContainer = createAppContainer(appNavigator);
11
12 export default AppContainer;
```

Header

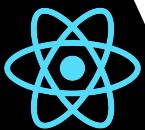


REACT NATIVE

Vamos trabalhar o Header da nossa aplicação

Podemos notar que agora o Header está ativo automaticamente



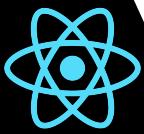


REACT NATIVE

Vamos trabalhar o Header da nossa aplicação

No StackNavigator configuramos primeiro as páginas

```
4  const AppNavigator = createStackNavigator(  
5    {  
6      'Main': {  
7        screen: HabitosPage  
8      }  
9    }  
10 );
```



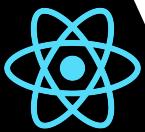
REACT NATIVE

Vamos trabalhar o Header da nossa aplicação

No StackNavigator configuramos primeiro as páginas

Mas podemos passar um outro objeto de configuração como segundo parâmetro do StackNavigator

```
4  const AppNavigator = createStackNavigator(  
5    {  
6      'Main': {  
7        screen: HabitosPage  
8      }  
9    },  
10   { }  
11 );
```

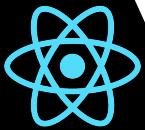


REACT NATIVE

Vamos trabalhar o Header da nossa aplicação

Este segundo parâmetro será o default da navegação

```
4  const AppNavigator = createStackNavigator(  
5    {  
6      'Main': ...  
15     }  
16   },  
17   {  
18     defaultNavigationOptions: {  
19       title: 'MyHabitTimeline',  
20       headerTintColor: 'white',  
21       headerStyle: {  
22         backgroundColor: '#6542f4',  
23         borderBottomColor: '#f4f2ff',  
24       },  
25       headerTitleStyle: {  
26         color: 'white',  
27         fontSize: 20,  
28         flexGrow: 1,  
29         textAlign: 'center',  
30       }  
31     }  
32   }
```



REACT NATIVE

Vamos trabalhar o Header da nossa aplicação

Este segundo parâmetro será o default da navegação e padrões do Header

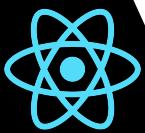
```
4  const AppNavigator = createStackNavigator(  
5    {  
6      'Main': ...  
15    }  
16  },  
17  {  
18    defaultNavigationOptions: {  
19      title: 'MyHabitTimeline',  
20      headerTintColor: 'white',  
21      headerStyle: {  
22        backgroundColor: '#6542f4',  
23        borderBottomColor: '#f4f2ff',  
24      },  
25      headerTitleStyle: {  
26        color: 'white',  
27        fontSize: 20,  
28        flexGrow: 1,  
29        textAlign: 'center',  
30      }  
31    }  
32  }
```

→ Título padrão de todas as telas que não foram configuradas

→ Cor do título e botão, se não for informado

→ Configuração do estilo do Header

→ Configuração do estilo do Título

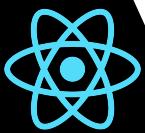


REACT NATIVE

Vamos trabalhar o Header da nossa aplicação

Na Rota Main vamos alterar e deixar atributos do navigationOptions estilizados para esta página

```
6     'Main': {
7       screen: HabitosPage,
8       navigationOptions: {
9         title: 'Hábitos',
10        headerTitleStyle: {
11          textAlign: 'left',
12          fontSize: 20,
13        },
14      }
15    }
16  },
17  {
18  [+]   defaultNavigationOptions: { ...
31    }
```



REACT NATIVE

Vamos trabalhar o Header da nossa aplicação

Na Rota Main vamos alterar e deixar atributos do navigationOptions estilizados para esta página

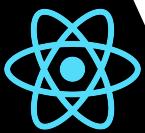
```
6   'Main': {
7     screen: HabitosPage,
8     navigationOptions: {
9       title: 'Hábitos',
10      headerTitleStyle: {
11        textAlign: 'left',
12        fontSize: 20,
13      },
14    }
15  }
16 },
17 {
18   defaultNavigationOptions: ...
31 }
```

Página que esta Rota vai executar

Vamos alterar o estilo das opções de navegação, não vamos usar o default

Título desta página

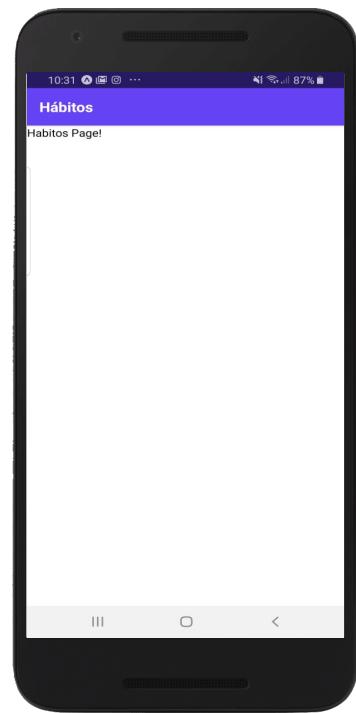
Estilo do Header desta página, alinhamos a esquerda porque, inicialmente, não tem botão de voltar



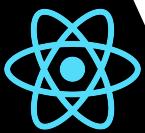
REACT NATIVE

Vamos trabalhar o Header da nossa aplicação

Podemos notar que agora o Header está ativo automaticamente



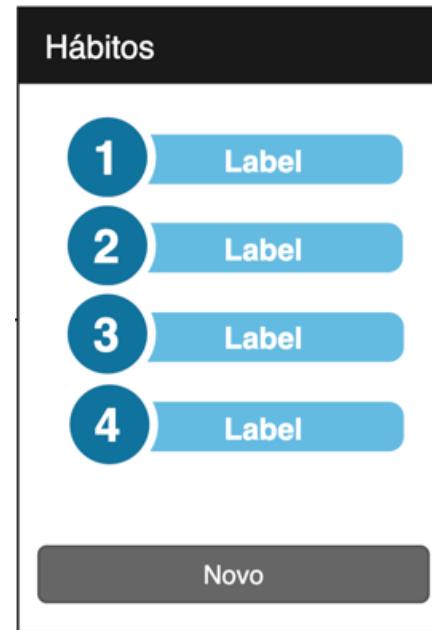
Listando Hábitos

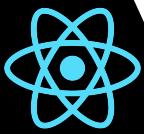


REACT NATIVE

Agora vamos montar nossa lista de Hábitos

Esse é apenas um rascunho da tela, não necessariamente será igual





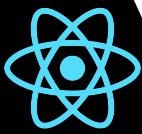
REACT NATIVE

Agora vamos montar nossa lista de Hábitos

JS Habitospage.js ×

```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 export default class Habitospage extends React.Component {
5   renderList() {
6     const habitos = [
7       'Correr', 'Musculação', 'Caminhar', 'Fast Food',
8       'Junk Food', 'Futebol', 'Seriado', 'Video Game'
9     ]
10  }
11
12  render() {
13    return (
14      <View>
15        <Text>Habitos Page!</Text>
16      </View>
17    );
18  }
19}
```

O React executa apenas a função
Render()



REACT NATIVE

Agora vamos montar nossa lista de Hábitos

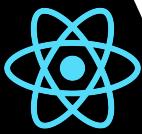
JS

HabitosPage.js

```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 export default class HabitosPage extends React.Component {
5     renderList() {
6         const habitos = [
7             'Correr', 'Musculação', 'Caminhar', 'Fast Food',
8             'Junk Food', 'Futebol', 'Seriado', 'Video Game'
9         ]
10    }
11
12    render() {
13        return (
14            <View>
15                <Text>Habitos Page!</Text>
16            </View>
17        );
18    }
19 }
```

Podemos programar outras funções para que a render seja configurada da forma que quisermos

Assim, programaremos a função renderList()
Nela, adicionamos alguns Hábitos para listagem



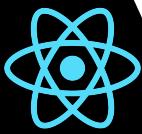
REACT NATIVE

Agora vamos montar nossa lista de Hábitos

```
JS HabitosPage.js ✘  
1 import React from 'react';  
2 import { StyleSheet, Text, View } from 'react-native';  
3  
4 export default class HabitosPage extends React.Component {  
5     renderList() {  
6         const habitos = [  
7             'Correr', 'Musculação', 'Caminhar', 'Fast Food',  
8             'Junk Food', 'Futebol', 'Seriado', 'Video Game'  
9         ]  
10    }  
11  
12    render() {  
13        return (  
14            <View>  
15                <Text>Habitos Page!</Text>  
16                { this.renderList() }  
17            </View>  
18        );  
19    }  
20}
```

Podemos programar outras funções para que a render seja configurada da forma que quisermos

Adicionamos a chamada da função renderList() dentro da função render() principal



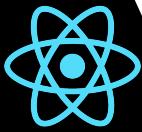
REACT NATIVE

Agora vamos montar nossa lista de Hábitos

```
JS HabitosPage.js ✘  
1 import React from 'react';  
2 import { StyleSheet, Text, View } from 'react-native';  
3  
4 export default class HabitosPage extends React.Component {  
5     renderList() {  
6         const habitos = [  
7             'Correr', 'Musculação', 'Caminhar', 'Fast Food',  
8             'Junk Food', 'Futebol', 'Seriado', 'Video Game'  
9         ]  
10    }  
11  
12    render() {  
13        return (  
14            <View>  
15                <Text>Habitos Page!</Text>  
16                { this.renderList() }  
17            </View>  
18        );  
19    }  
20}
```

Agora temos uma função para retornar a lista de hábitos e estamos executando esta função no método principal

Precisamos fazer com que a função renderList() retorne alguma coisa para que possamos testar o correto funcionamento



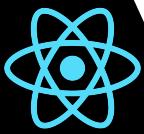
REACT NATIVE

Agora vamos montar nossa lista de Hábitos

```
JS HabitosPage.js ×  
1 import React from 'react';  
2 import { StyleSheet, Text, View } from 'react-native';  
3  
4 export default class HabitosPage extends React.Component {  
5     renderList() {  
6         const habitos = [  
7             'Correr', 'Musculação', 'Caminhar', 'Fast Food',  
8             'Junk Food', 'Futebol', 'Seriado', 'Video Game'  
9         ];  
10  
11         return <Text>Função para retorno da Listagem de Hábitos!</Text>  
12     }  
13  
14     render() {  
15         return (  
16             <View>  
17                 <Text>Habitos Page!</Text>  
18                 { this.renderList() }  
19             </View>  
20         );  
21     }  
22 }
```

Agora temos uma função para retornar a lista de hábitos e estamos executando esta função no método principal

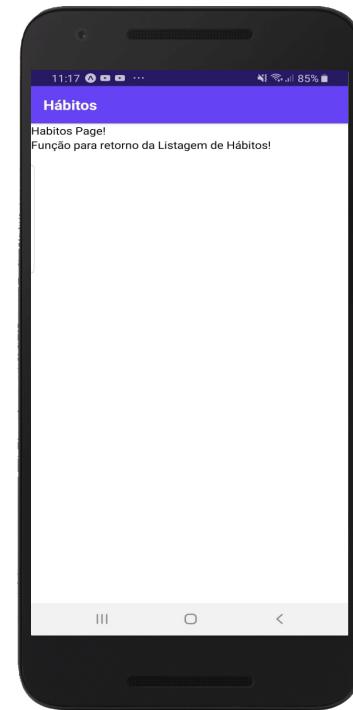
Precisamos fazer com que a função renderList() retorne alguma coisa para que possamos testar o correto funcionamento

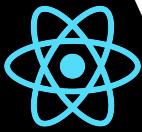


REACT NATIVE

Agora vamos montar nossa lista de Hábitos

```
js HabitosPage.js ✘
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 export default class HabitosPage extends React.Component {
5     renderList() {
6         const habitos = [
7             'Correr', 'Musculação', 'Caminhar', 'Fast Food',
8             'Junk Food', 'Futebol', 'Seriado', 'Video Game'
9         ];
10
11         return <Text>Função para retorno da Listagem de Hábitos!</Text>
12     }
13
14     render() {
15         return (
16             <View>
17                 <Text>Habitos Page!</Text>
18                 { this.renderList() }
19             </View>
20         );
21     }
22 }
```





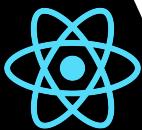
REACT NATIVE

Agora vamos montar nossa lista de Hábitos

Com a função `renderList()` funcionando corretamente, agora devemos retornar a lista dos hábitos configurados no array

Para isso, vamos usar o método "map", presente no objeto array do JavaScript

```
5  □    renderList() {
6  □      const habitos = [
7  □          'Correr', 'Musculação', 'Caminhar', 'Fast Food',
8  □          'Junk Food', 'Futebol', 'Seriado', 'Video Game'
9  □      ];
10
11 □      const ret = habitos.map(habito => {
12  □          return <Text>{ habito }</Text>
13  □      });
14
15  □      return ret;
16  □ }
```



REACT NATIVE

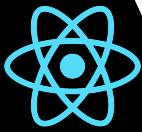
Agora vamos montar nossa lista de Hábitos

Com a função `renderList()` funcionando corretamente, agora devemos retornar a lista dos hábitos configurados no array

Para isso, vamos usar o método "map", presente no objeto array do JavaScript

Método `map` irá executar a `arrow function` (conceito JS) com retorno para cada item do array `habitos`

```
5  □    renderList() {
6  □      const habitos = [
7  □          'Correr', 'Musculação', 'Caminhar', 'Fast Food',
8  □          'Junk Food', 'Futebol', 'Seriado', 'Video Game'
9  □      ];
10
11     const ret = habitos.map(habito => {
12         return <Text>{ habito }</Text>
13     );
14
15     return ret;
16 }
```



REACT NATIVE

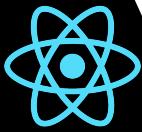
Agora vamos montar nossa lista de Hábitos

Com a função renderList() funcionando corretamente, agora devemos retornar a lista dos hábitos configurados no array.

Para isso, usaremos o método "map", presente no objeto array do JavaScript



```
5  □    renderList() {  
6  □      const habitos = [  
7  □          'Correr', 'Musculação', 'Caminhar', 'Fast Food',  
8  □          'Junk Food', 'Futebol', 'Seriado', 'Video Game'  
9  □      ];  
10  
11 □      const ret = habitos.map(habito => {  
12          return <Text>{ habito }</Text>  
13      });  
14  
15      return ret;  
16  }
```

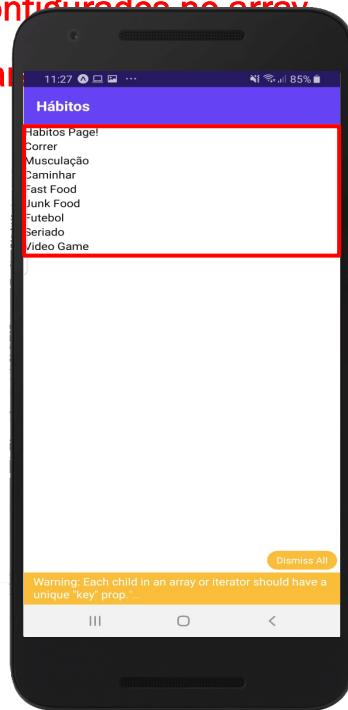


REACT NATIVE

Agora vamos montar nossa lista de Hábitos

Com a função `renderList()` função que irá renderizar a lista de hábitos configurados no array.

Para isso, precisaremos usar o método

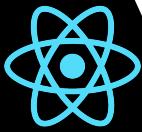


Listagem funcionando corretamente

nos retornar a lista dos hábitos

o JavaScript

```
5   renderList() {  
6     const habitos = [  
7       'Correr', 'Musculação', 'Caminhar', 'Fast Food',  
8       'Junk Food', 'Futebol', 'Seriado', 'Video Game'  
9     ];  
10  
11    const ret = habitos.map(habito => {  
12      return <Text>{ habito }</Text>  
13    });  
14  
15    return ret;  
16  }
```

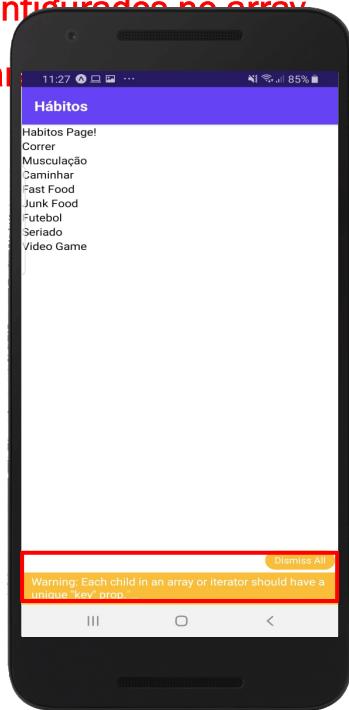


REACT NATIVE

Agora vamos montar nossa lista de Hábitos

Com a função `renderList()` podemos retornar a lista dos hábitos configurados no array.

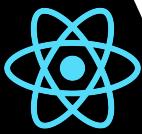
Para isso, precisamos usar o método



Mas, temos um warning para resolver

o JavaScript

```
5   renderList() {  
6     const habitos = [  
7       'Correr', 'Musculação', 'Caminhar', 'Fast Food',  
8       'Junk Food', 'Futebol', 'Seriado', 'Video Game'  
9     ];  
10  
11    const ret = habitos.map(habito => {  
12      return <Text>{ habito }</Text>  
13    });  
14  
15    return ret;  
16  }
```



REACT NATIVE

Agora vamos montar nossa lista de Hábitos

Com a função `renderList()` podemos retornar a lista dos hábitos configurados no array.

Para isso, precisamos usar o método



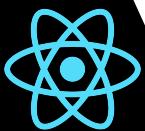
Mas, temos um warning para resolver

Cada elemento da lista deve conter uma chave única

nos retornar a lista dos hábitos

o JavaScript

```
5   renderList() {  
6     const habitos = [  
7       'Correr', 'Musculação', 'Caminhar', 'Fast Food',  
8       'Junk Food', 'Futebol', 'Seriado', 'Video Game'  
9     ];  
10  
11    const ret = habitos.map(habito => {  
12      return <Text>{ habito }</Text>  
13    });  
14  
15    return ret;  
16  }
```



REACT NATIVE

Agora vamos montar nossa lista de Hábitos

Com a função `renderList()` podemos retornar a lista dos hábitos configurados no array.



Mas, temos um warning para resolver

Cada elemento da lista deve conter uma chave única

nos retornar a lista dos hábitos

o JavaScript

```
renderList() {
```

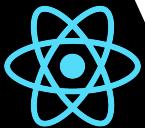
O React solicita uma key por elemento por causa de alterações dinâmicas

Vamos supor que a lista tenha 50 itens.

Em determinado momento o elemento 33 alterou seu estado e precisa ser atualizado.

Facilita para o React o uso da key para que atualize unicamente um elemento e não todo o array

'Fast Food',
'Video Game'



REACT NATIVE

Agora vamos montar nossa lista de Hábitos

Com a função `renderList()` podemos retornar a lista dos hábitos configurados no array.

Para isso usaremos o método



O segundo parâmetro do map é um índice sequencial

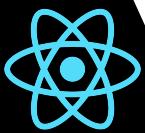
JavaScript

Usaremos ele para ser a key do elemento

```
4  export default class HabitosPage extends React.Component {  
5    renderList() {  
6      const habitos = [  
7        'Correr', 'Musculação', 'Caminhar', 'Fast Food',  
8        'Junk Food', 'Futebol', 'Seriado', 'Video Game'  
9      ];  
10  
11     const ret = habitos.map((habito, idx) => {  
12       return <Text key={ idx }>{ habito }</Text>  
13     });  
14  
15     return ret;
```

Consumindo API

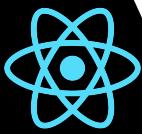
Json



REACT NATIVE

Para simular um comportamento real de um aplicativo, vamos listar os dados de um arquivo Json

Geralmente APIs Web retornam arquivos Json com estruturas de dados para comunicação entre diferentes sistemas

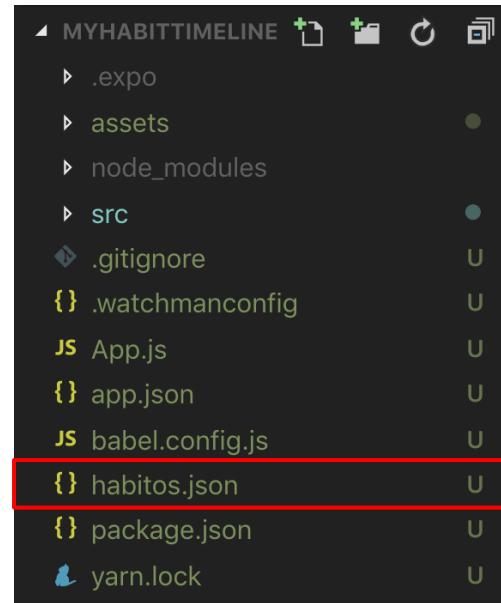


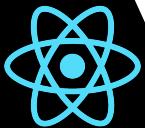
REACT NATIVE

Para simular um comportamento real de um aplicativo, vamos listar os dados de um arquivo Json

Geralmente APIs Web retornam arquivos Json com estruturas de dados para comunicação entre diferentes sistemas

Então, vamos criar um arquivo json chamado `habitos.json`





REACT NATIVE

Para simular um comportamento de API usaremos um arquivo Json

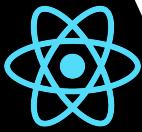
Geralmente APIs Web retornam JSON para facilitar a comunicação entre diferentes sistemas

Então, vamos criar um arquivo json

```
{} habitos.json ●  
1 [  
2 {  
3   "id": "0",  
4   "nome": "Correr",  
5   "descricao": "Corrida"  
6 },  
7 {  
8   "id": "1",  
9   "nome": "Musculação",  
10  "descricao": "Academia"  
11 },  
12 {  
13   "id": "2",  
14   "nome": "Caminhar",  
15   "descricao": "Cainhada"  
16 },  
17 {  
18   "id": "3",  
19   "nome": "Fast Food",  
20   "descricao": "Comer porcaria"  
21 },  
22 {  
23   "id": "4",  
24   "nome": "Junk Food",  
25   "descricao": "Comer comida menos saudável"  
26 },  
27 {  
28   "id": "5",  
29   "nome": "Futebol",  
30   "descricao": "Jogar futebol"  
31 },  
32 {  
33   "id": "6",  
34   "nome": "Seriado",  
35   "descricao": "Assistir NetFlix"  
36 },  
37 {  
38   "id": "7",  
39   "nome": "Video Game",  
40   "descricao": "Jogar Video Game"  
41 }]  
42 -
```

Nos listar os dados de um array

que representam os dados para comunicação entre sistemas

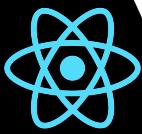


REACT NATIVE

Para simular um comportamento real de um aplicativo, vamos listar os dados de um arquivo Json

Podemos usar este json para a listagem dos Hábitos

```
JS HabitosPage.js ✘
1  import React from 'react';
2  import { StyleSheet, Text, View } from 'react-native';
3
4  import habitos from '../.../habitostest.json';
5
6  export default class Habitostest extends React.Component {
7      renderList() {
8          const ret = habitos.map((habito, idx) => {
9              return <Text key={ habito.id }>{ habito.nome }</Text>
10         });
11
12         return ret;
13     }
14
15     render() {
16         return (
17             <View>
18                 <Text>Habitos Page!</Text>
19                 { this.renderList() }
20             </View>
21         );
22     }
23 }
```



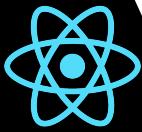
REACT NATIVE

Para simular um comportamento real de um aplicativo, vamos listar os dados de um arquivo Json

Podemos usar este json para a listagem dos Hábitos

```
JS HabitosPage.js ✘
1  import React from 'react';
2  import { StyleSheet, Text, View } from 'react-native';
3
4  import habitos from '../.../habitostest.json';
5
6  export default class Habitostest extends React.Component {
7      renderList() {
8          const ret = habitos.map((habito, idx) => {
9              return <Text key={ habito.id }>{ habito.nome }</Text>
10         });
11
12         return ret;
13     }
14
15     render() {
16         return (
17             <View>
18                 <Text>Habitos Page!</Text>
19                 { this.renderList() }
20             </View>
21         );
22     }
23 }
```

Faremos o import deste Json para dentro da variável "habitostest"



REACT NATIVE

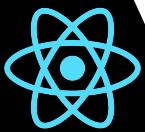
Para simular um comportamento real de um aplicativo, vamos listar os dados de um arquivo Json

Podemos usar este json para a listagem dos Hábitos

```
JS HabitosPage.js ×  
1  import React from 'react';  
2  import { StyleSheet, Text, View } from 'react-native';  
3  
4  import habitos from '../.../habitostest.json';  
5  
6  export default class HabitostestPage extends React.Component {  
7      renderList() {  
8          const ret = habitos.map((habito, idx) => {  
9              return <Text key={ habito.id }>{ habito.nome }</Text>  
10         });  
11  
12         return ret;  
13     }  
14  
15     render() {  
16         return (  
17             <View>  
18                 <Text>Habitos Page!</Text>  
19                 { this.renderList() }  
20             </View>  
21         );  
22     }  
23 }
```

Usamos os dados na const ret no lugar do array anterior

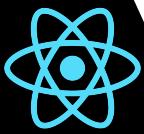
ComponentDidMount



REACT NATIVE

Para simular um comportamento real de um aplicativo, vamos listar os dados de um arquivo Json

Geralmente APIs Web retornam arquivos Json com estruturas de dados para comunicação entre diferentes sistemas

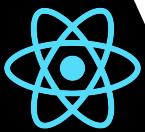


REACT NATIVE

ComponentDidMount é uma função executada depois que um componente é montado

JS Habitospage.js ×

```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 import habitos from '../habitostest.json';
5
6 export default class Habitospage extends React.Component {
7     constructor(props) {
8         super(props);
9
10        this.state = {
11            habitos: []
12        };
13    }
14
15    componentDidMount() {
16
17    }
```



REACT NATIVE

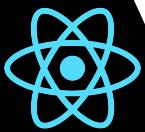
ComponentDidMount é uma função executada depois que um componente é montado

JS Habitospage.js ×

```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 import habitos from '../habitostest.json';
5
6 export default class Habitospage extends React.Component {
7     constructor(props) {
8         super(props);
9
10        this.state = {
11            habitos: []
12        };
13    }
14
15    componentDidMount() {
16
17    }

```

Para trabalharmos com
componentDidMount() precisamos
trabalhar com **props** e **states**



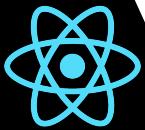
REACT NATIVE

ComponentDidMount é uma função executada depois que um componente é montado

JS Habitospage.js ×

```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 import habitos from '../habitostest.json';
5
6 export default class Habitospage extends React.Component {
7     constructor(props) {
8         super(props);
9
10        this.state = {
11            habitos: []
12        };
13    }
14
15    componentDidMount() {
16
17    }
}
```

Para trabalharmos com
componentDidMount() precisamos
trabalhar com **props** e **states**



REACT NATIVE

ComponentDidMount é uma função executada depois que um componente é montado

JS Habitospage.js ×

```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 import habitos from '../habitospage.json';
5
6 export default class Habitospage extends React.Component {
7     constructor(props) {
8         super(props);
9
10        this.state = {
11            habitos: []
12        };
13    }
14
15    componentDidMount() {
16
17    }
}
```

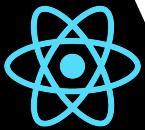
State

É um atributo da classe
React.Component

A ideia é trabalhar o state como
um variável da classe

Para ler o valor usamos
this.state.variavel

Para alterar o valor usamos
this.setState({
 variavel: novo_valor
})



REACT NATIVE

ComponentDidMount é uma função executada depois que um componente é montado

JS Habitospage.js ×

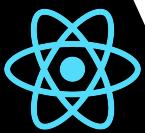
```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 import habitos from '../habitostest.json';
5
6 export default class Habitospage extends React.Component {
7   constructor(props) {
8     super(props);
9
10    this.state = {
11      habitos: []
12    };
13  }
14
15  componentDidMount() {
16
17  }
```

Props

É um atributo da classe
React.Component

A ideia é trabalhar o props como
um parâmetro da classe

Podemos criar as
classes/componentes (tags) com
fields específicos



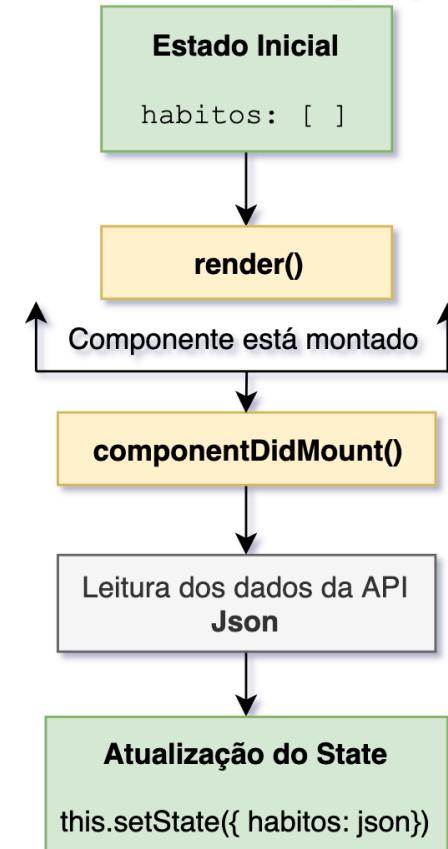
REACT NATIVE

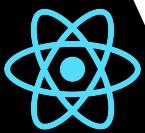


HabitosPage.js

- A página inicia configurando seu estado inicial, através do *constructor*

```
this.state = {  
  habitos: []  
};
```





REACT NATIVE

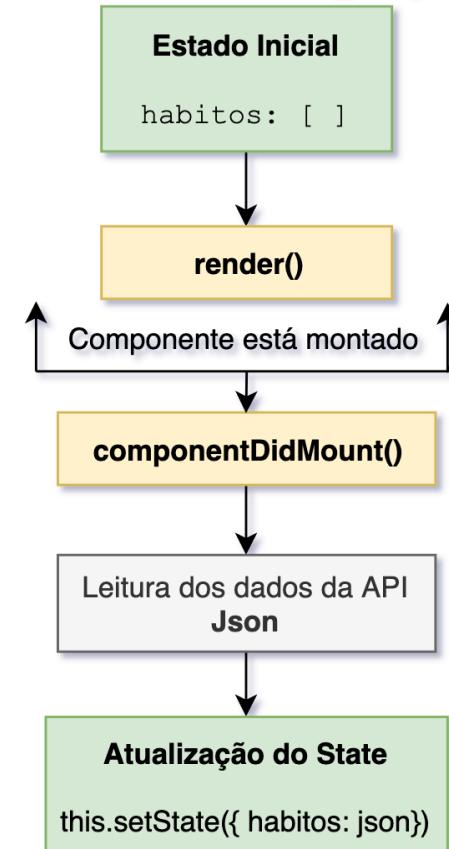


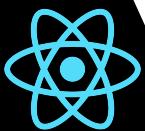
HabitosPage.js

- A página inicia configurando seu estado inicial, através do *constructor*

```
this.state = {  
  habitos: []  
};
```

- O próximo método a executar é o *render()*
 - Com isso, o componente já está montado





REACT NATIVE

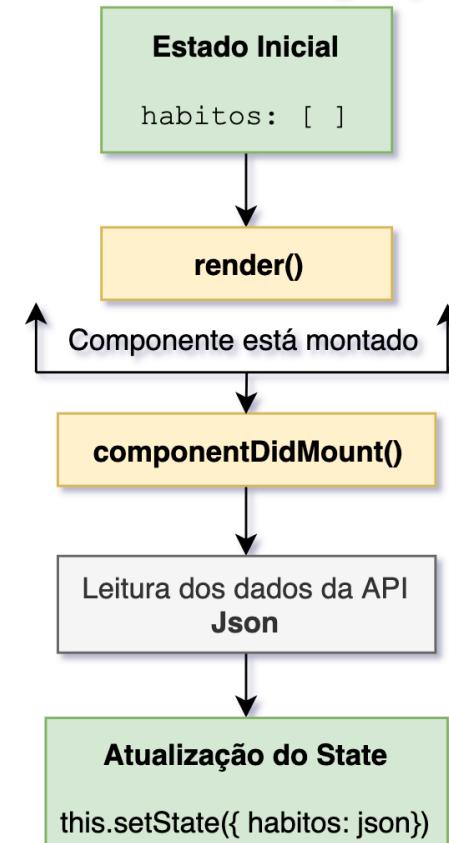


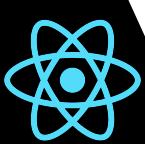
HabitosPage.js

- A página inicia configurando seu estado inicial, através do *constructor*

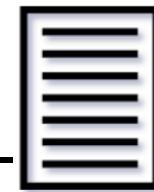
```
this.state = {  
  habitos: []  
};
```

- O próximo método a executar é o *render()*
 - Com isso, o componente já está montado
- A execução do *componentDidMount* irá fazer a leitura dos dados da API para que na sequência o estado da página seja alterado





REACT NATIVE

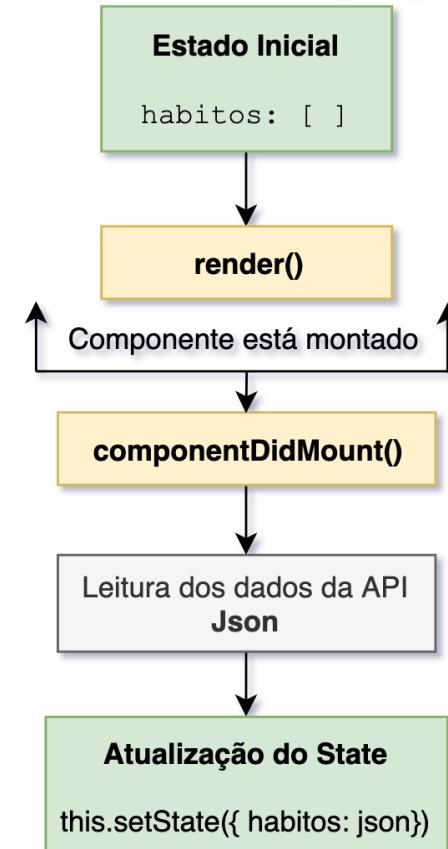


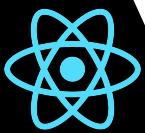
HabitosPage.js

- A página inicia configurando seu estado inicial, através do *constructor*

```
this.state = {  
  habitos: []  
};
```

- O próximo método a executar é o *render()*
 - Com isso, o componente já está montado
- A execução do *componentDidMount* irá fazer a leitura dos dados da API para que na sequência o estado da página seja alterado
- Fazemos esse controle de componente montado para que não ocorra da leitura dos dados ser efetuada antes dos componentes estarem prontos para serem renderizados



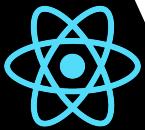


REACT NATIVE

Renomeamos habitos para habitosJson, apenas para facilitar entendimento

JS Habitospage.js ×

```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 import habitosJson from '../../habitosts.json';
```



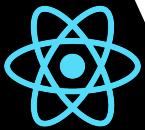
REACT NATIVE

Mantemos o código do renderList()

Mas retiramos o uso dele no render()

JS HabitosPage.js ×

```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 import habitosJson from '../../habitados.json';
5
6
7 export default class HabitosPage extends React.Component {
8   constructor(props) {
9     super(props);
10
11     this.state = {
12       habitos: []
13     };
14
15   componentDidMount() {
16     this.setState({
17       habitos: habitosJson
18     });
19
20
21   renderList() {
22     const ret = habitos.map((habito, idx) => {
23       return <Text key={ habito.id }>{ habito.nome }</Text>
24     });
25
26     return ret;
27   }
28
29   render() {
30     return (
31       <View>
32         <Text>Habitos Page!</Text>
33       </View>
34     );
35   }
36 }
```

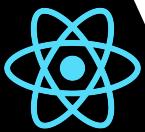


REACT NATIVE

No componentDidMount() atualizamos o state da página

JS HabitosPage.js ×

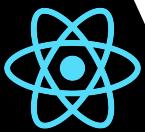
```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 import habitosJson from '../../habitados.json';
5
6 export default class HabitosPage extends React.Component {
7   constructor(props) {
8     super(props);
9
10    this.state = {
11      habitos: []
12    };
13  }
14
15  componentDidMount() {
16    this.setState({
17      habitos: habitosJson
18    });
19  }
20
21  renderList() {
22    const ret = habitos.map((habito, idx) => {
23      return <Text key={ habito.id }>{ habito.nome }</Text>
24    });
25
26    return ret;
27  }
28
29  render() {
30    return (
31      <View>
32        <Text>Habitos Page!</Text>
33      </View>
34    );
35  }
36}
```



REACT NATIVE

Alteramos o renderList() para agora buscar dados do state

```
20
21     renderList() {
22         const ret = this.state.habitos.map(habito => {
23             const { id, nome } = habito;
24             return <Text key={ id }>{ id } - { nome }</Text>
25         });
26
27         return ret;
28     }
29
30     render() {
31         return (
32             <View>
33                 <Text>Habitos Page!</Text>
34                 { this.renderList() }
35             </View>
36         );
37     }
38 }
```

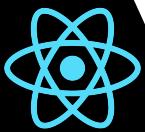


REACT NATIVE

Alteramos o `renderList()` para agora buscar dados do state

Aplicamos o método *map* ao state

```
20
21     renderList() {
22         const ret = this.state.habitos.map(habito => {
23             const { id, nome } = habito;
24             return <Text key={ id }>{ id } - { nome }</Text>
25         });
26
27         return ret;
28     }
29
30     render() {
31         return (
32             <View>
33                 <Text>Habitos Page!</Text>
34                 { this.renderList() }
35             </View>
36         );
37     }
38 }
```

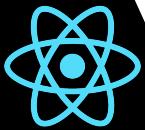


REACT NATIVE

Alteramos o renderList() para agora buscar dados do state

Utilizamos o conceito de desestruturação para separar os dados do habitó em 2 variáveis

```
20
21     renderList() {
22         const ret = this.state.habitos.map(habito => {
23             const { id, nome } = habito;
24             return <Text key={ id }>{ id } - { nome }</Text>
25         });
26
27         return ret;
28     }
29
30     render() {
31         return (
32             <View>
33                 <Text>Habitos Page!</Text>
34                 { this.renderList() }
35             </View>
36         );
37     }
38 }
```

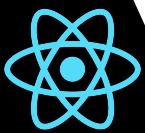


REACT NATIVE

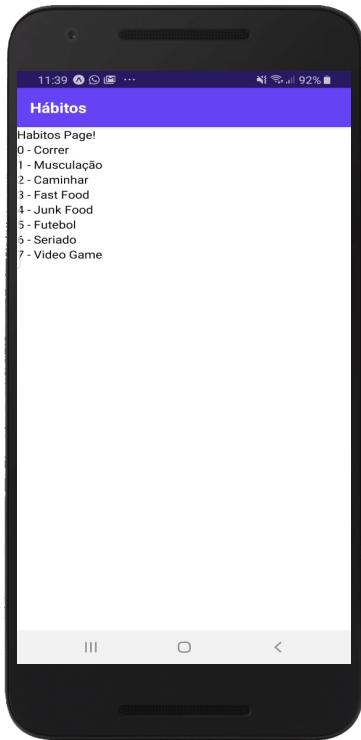
Alteramos o renderList() para agora buscar dados do state

Utilizamos estas 2 variáveis para montar o componente de retorno

```
20
21     renderList() {
22         const ret = this.state.habitos.map(habito => {
23             const { id, nome } = habito;
24             return <Text key={ id }>{ id } - { nome }</Text>
25         });
26
27         return ret;
28     }
29
30     render() {
31         return (
32             <View>
33                 <Text>Habitos Page!</Text>
34                 { this.renderList() }
35             </View>
36         );
37     }
38 }
```



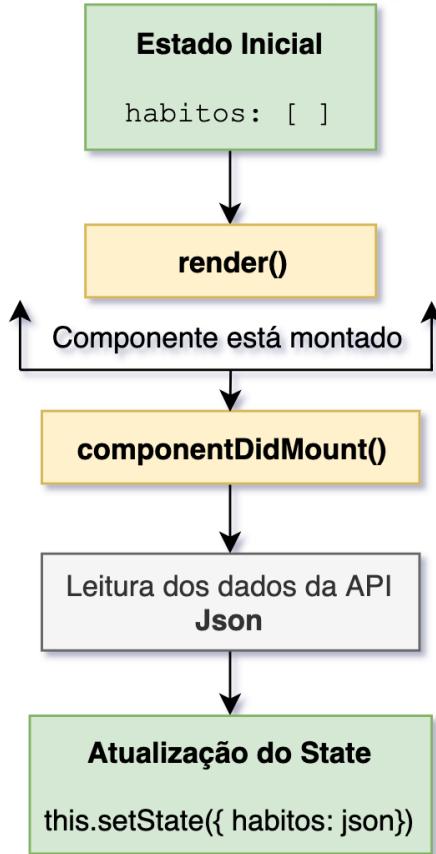
REACT NATIVE



```
20
21     renderList() {
22         const ret = this.state.habitos.map(habito => {
23             const { id, nome } = habito;
24             return <Text key={ id }>{ id } - { nome }</Text>
25         });
26
27         return ret;
28     }
29
30     render() {
31         return (
32             <View>
33                 <Text>Habitos Page!</Text>
34                 { this.renderList() }
35             </View>
36         );
37     }
38 }
```

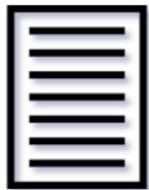


HabitosPage.js

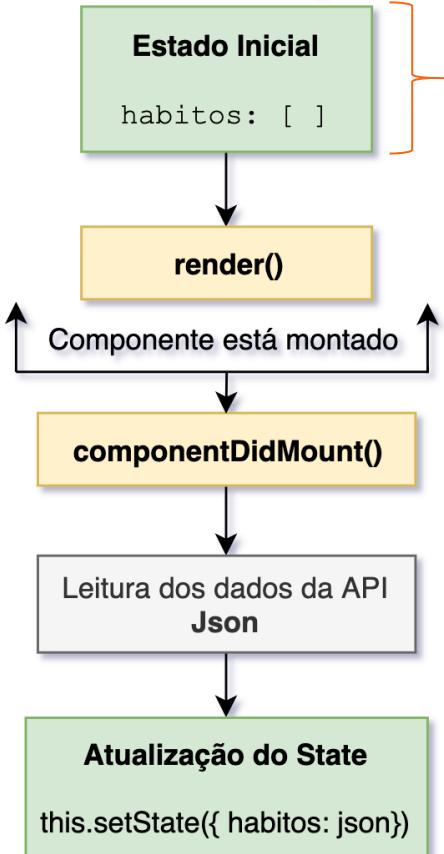


JS HabitosPage.js

```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 import habitosJson from '../../../../../habitios.json';
5
6 export default class HabitosPage extends React.Component {
7   constructor(props) {
8     super(props);
9
10    this.state = {
11      habitos: []
12    };
13  }
14
15  componentDidMount() {
16    this.setState({
17      habitos: habitosJson
18    });
19  }
20
21  renderList() {
22    const ret = this.state.habitos.map(habito => {
23      const { id, nome } = habito;
24      return <Text key={ id }>{ id } - { nome }</Text>
25    });
26
27    return ret;
28  }
29
30  render() {
31    return (
32      <View>
33        <Text>Habitos Page!</Text>
34        { this.renderList() }
35      </View>
36    );
37  }
38}
```

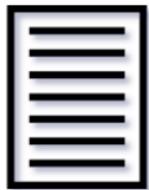


HabitosPage.js

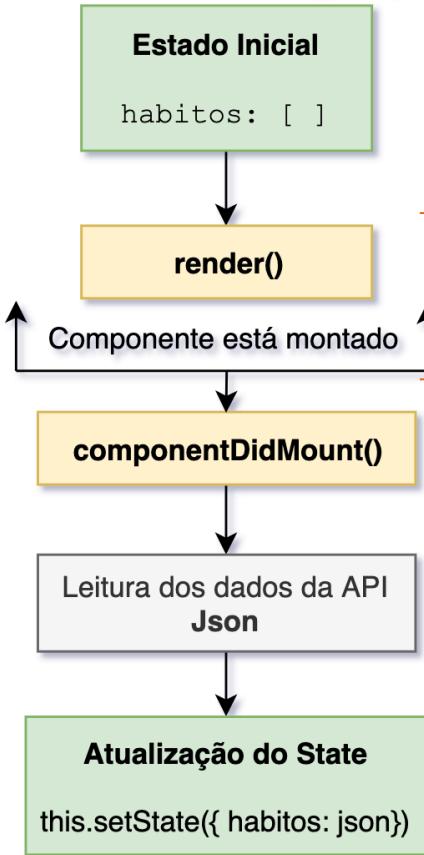


JS HabitosPage.js ×

```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 import habitosJson from '../../../../../habitos.json';
5
6 export default class HabitosPage extends React.Component {
7   constructor(props) {
8     super(props);
9
10    this.state = {
11      habitos: []
12    };
13  }
14
15  componentDidMount() {
16    this.setState({
17      habitos: habitosJson
18    });
19  }
20
21  renderList() {
22    const ret = this.state.habitos.map(habito => {
23      const { id, nome } = habito;
24      return <Text key={ id }>{ id } - { nome }</Text>
25    });
26
27    return ret;
28  }
29
30  render() {
31    return (
32      <View>
33        <Text>Habitos Page!</Text>
34        { this.renderList() }
35      </View>
36    );
37  }
38}
```

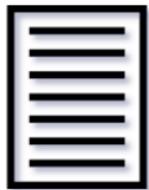


HabitosPage.js

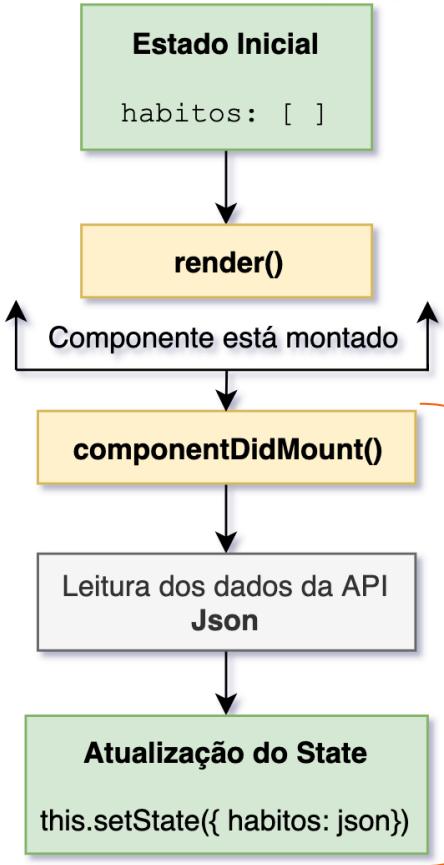


JS HabitosPage.js ×

```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 import habitosJson from '../../../../../habitos.json';
5
6 export default class HabitosPage extends React.Component {
7   constructor(props) {
8     super(props);
9
10    this.state = {
11      habitos: []
12    };
13  }
14
15  componentDidMount() {
16    this.setState({
17      habitos: habitosJson
18    });
19  }
20
21  renderList() {
22    const ret = this.state.habitos.map(habito => {
23      const { id, nome } = habito;
24      return <Text key={ id }>{ id } - { nome }</Text>
25    });
26
27    return ret;
28  }
29
30  render() {
31    return (
32      <View>
33        <Text>Habitos Page!</Text>
34        { this.renderList() }
35      </View>
36    );
37  }
38}
```



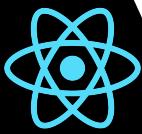
HabitosPage.js



JS HabitosPage.js ×

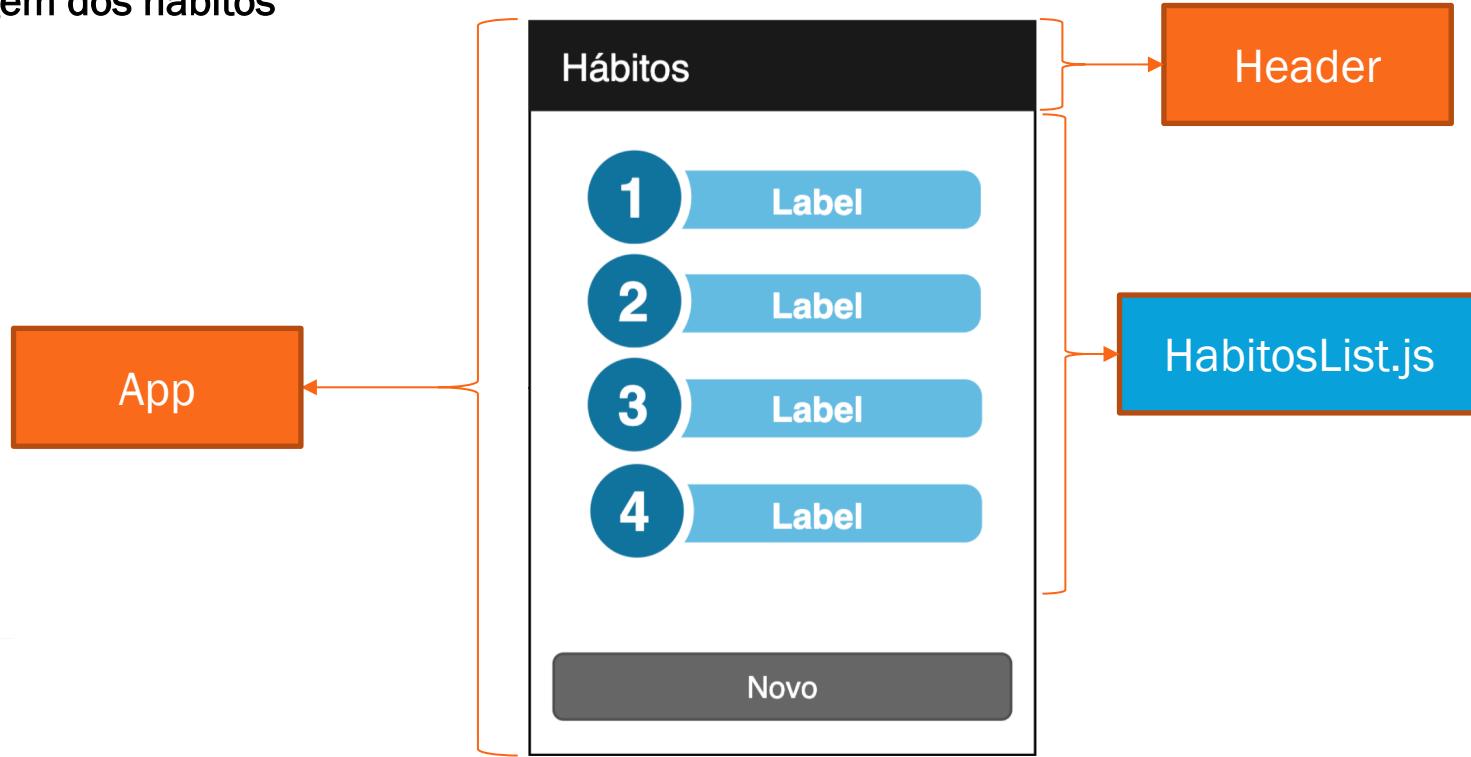
```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 import habitosJson from '../../../../../habitos.json';
5
6 export default class HabitosPage extends React.Component {
7   constructor(props) {
8     super(props);
9
10    this.state = {
11      habitos: []
12    };
13  }
14
15  componentDidMount() {
16    this.setState({
17      habitos: habitosJson
18    });
19  }
20
21  renderList() {
22    const ret = this.state.habitos.map(habito => {
23      const { id, nome } = habito;
24      return <Text key={ id }>{ id } - { nome }</Text>
25    });
26
27    return ret;
28  }
29
30  render() {
31    return (
32      <View>
33        <Text>Habitos Page!</Text>
34        { this.renderList() }
35      </View>
36    );
37  }
38}
```

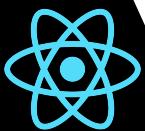
Criando Novo Componente



REACT NATIVE

Para melhor organização, vamos retirar o uso do renderList() e criar um componente para a listagem dos hábitos



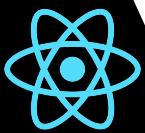


REACT NATIVE

Para melhor organização, vamos retirar o uso do renderList() e criar um componente para a listagem dos hábitos

Criaremos arquivo **HabitosList.js**

```
▲ src          •  
  ▲ components •  
    JS HabitosList.js  U  
  ▲ pages        •  
    JS HabitosPage.js U
```

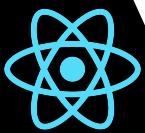


REACT NATIVE

Criando componente HabitosList

JS HabitosList.js ✘

```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 const HabitosList = props => (
5   <View>
6     <Text>NOSSA LISTA</Text>
7   </View>
8 );
9
10 const styles = StyleSheet.create({
11
12 }) ;
```



REACT NATIVE

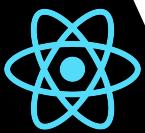
Criando componente HabitosList

JS HabitosList.js ×

```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 const HabitosList = props => (
5   <View>
6     <Text>NOSSA LISTA</Text>
7   </View>
8 );
9
10 const styles = StyleSheet.create({
11 });
12 
```

Como neste componente
não vamos fazer uso de States,
deixamos ele como
FUNCTIONAL COMPONENT

e não como
CLASS COMPONENT



REACT NATIVE

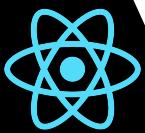
Criando componente HabitosList

JS HabitosList.js ×

```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 const HabitosList = props => (
5   <View>
6     <Text>NOSSA LISTA</Text>
7   </View>
8 );
9
10 const styles = StyleSheet.create({
11 });
12 
```

Este componente terá PROPS

Quando usarmos ele na página
vamos passar os dados via
props (like parameters)



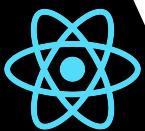
REACT NATIVE

Criando componente HabitosList

JS HabitosList.js ✘

```
1  import React from 'react';
2  import { StyleSheet, Text, View } from 'react-native';
3
4  const HabitosList = props => (
5    <View>
6      <Text>NOSSA LISTA</Text>
7    </View>
8  );
9
10 const styles = StyleSheet.create({
11
12 });
13
14 export default HabitosList;
```

Sempre que criamos um componente devemos exportar nosso componente para que possamos usar em outras páginas



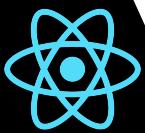
REACT NATIVE

Criando componente HabitosList

Para podermos usar este novo componente
na página de Habitos, devemos fazer a importação dele

JS HabitosPage.js

```
1 import React from 'react';
2 import { Text, View } from 'react-native';
3 import HabitosList from '../components/HabitosList';
```

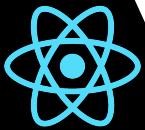


REACT NATIVE

Criando componente HabitosList

Para usar o componente, retiramos o `this.renderList()` e adicionamos a tag do novo componente

```
31     render() {  
32         return (  
33             <View>  
34                 <Text>Habitos Page!</Text>  
35                 <HabitosList />  
36             </View>  
37         );  
38     };
```

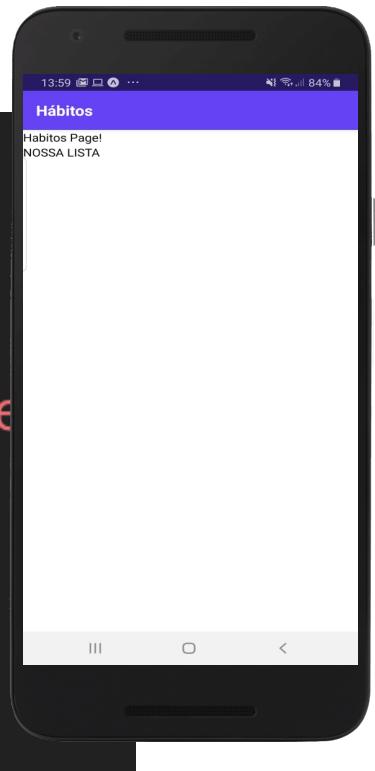


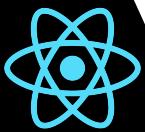
REACT NATIVE

Criando componente HabitosList

Para usar o componente, retiramos o `this.renderList()` e adicionamos a tag do novo componente

```
31     render() {  
32         return (  
33             <View>  
34                 <Text>Habitos Page!</Text>  
35                 <HabitosList />  
36             </View>  
37         );  
38     };
```

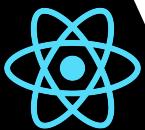




REACT NATIVE

JS HabitosList.js ×

```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 const HabitosList = props => {
5     const { habitos } = props;
6     const ret = habitos.map(habito => {
7         const { id, nome } = habito;
8         return <Text key={ id }>{ id } - { nome }</Text>
9     });
10
11    return (
12        <View>
13            |   { ret }
14        </View>
15    )
16};
17
18 const styles = StyleSheet.create({
19
20});
21
22 export default HabitosList;
```

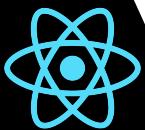


REACT NATIVE

Aplicamos conceito de desestruturação de props

JS HabitosList.js ×

```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 const HabitosList = props => {
5     const { habitos } = props;
6     const ret = habitos.map(habito => {
7         const { id, nome } = habito;
8         return <Text key={ id }>{ id } - { nome }</Text>
9     });
10
11    return (
12        <View>
13            { ret }
14        </View>
15    )
16};
17
18 const styles = StyleSheet.create({
19
20 });
21
22 export default HabitosList;
```

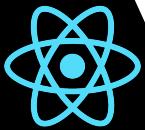


REACT NATIVE

Passamos a lógica do `renderList()`
para dentro do componente

JS HabitosList.js ×

```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 const HabitosList = props => {
5   const { habitos } = props;
6   const ret = habitos.map(habito => {
7     const { id, nome } = habito;
8     return <Text key={ id }>{ id } - { nome }</Text>
9   });
10
11   return (
12     <View>
13       { ret }
14     </View>
15   )
16 };
17
18 const styles = StyleSheet.create({
19
20 });
21
22 export default HabitosList;
```

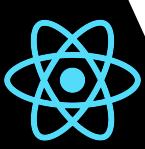


REACT NATIVE

Adicionamos a props na chamada do HabitosList

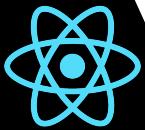
JS HabitosPage.js ×

```
1 import React from 'react';
2 import { Text, View } from 'react-native';
3 import HabitosList from '../components/HabitosList';
4
5 import habitosJson from '../../habitos.json';
6
7 export default class HabitosPage extends React.Component {
8   constructor(props) {
9     super(props);
10
11     this.state = {
12       habitos: []
13     };
14   }
15
16   componentDidMount() {
17     this.setState({
18       habitos: habitosJson
19     });
20   }
21
22   render() {
23     return (
24       <View>
25         <HabitosList habitos={ this.state.habitos } />
26       </View>
27     );
28   }
29 }
```



REACT NATIVE

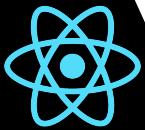
Agora, vamos estilizar nossa listagem



REACT NATIVE

Agora, vamos estilizar nossa listagem

```
JS HabitosList.js ✘
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 const HabitosList = props => {
5   const { habitos } = props;
6   const ret = habitos.map(habito => {
7     const { id, nome } = habito;
8     return <Text style={ styles.list } key={ id }>{ id } - { nome }</Text>
9   });
10
11   return (
12     <View>
13       { ret }
14     </View>
15   )
16 };
17
18 const styles = StyleSheet.create({
19   list: {
20     padding: 40
21   }
22 });
23
24 export default HabitosList;
```

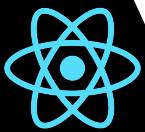


REACT NATIVE

Agora, vamos estilizar nossa listagem

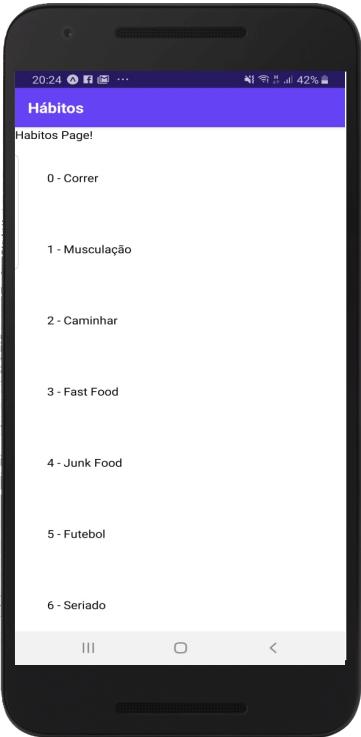
Criamos um estilo chamado list e usamos ele no retorno do componente

```
JS HabitosList.js ×
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 const HabitosList = props => {
5   const { habitos } = props;
6   const ret = habitos.map(habito => {
7     const { id, nome } = habito;
8     return <Text style={ styles.list } key={ id }>{ id } - { nome }</Text>
9   });
10
11   return (
12     <View>
13       { ret }
14     </View>
15   );
16 };
17
18 const styles = StyleSheet.create({
19   list: {
20     padding: 40
21   }
22 });
23
24 export default HabitosList;
```

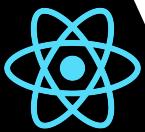


REACT NATIVE

Agora, vamos estilizar nossa listagem

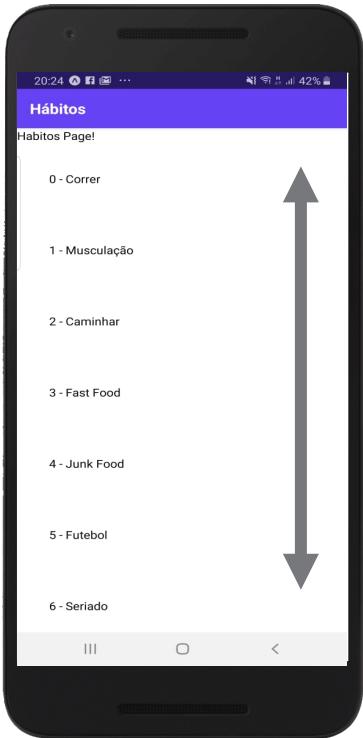


```
JS HabitosList.js ✘
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 const HabitosList = props => {
5   const { habitos } = props;
6   const ret = habitos.map(habito => {
7     const { id, nome } = habito;
8     return <Text style={ styles.list } key={ id }>{ id } - { nome }</Text>;
9   });
10
11   return (
12     <View>
13       { ret }
14     </View>
15   );
16 };
17
18 const styles = StyleSheet.create({
19   list: {
20     padding: 40
21   }
22 });
23
24 export default HabitosList;
```



REACT NATIVE

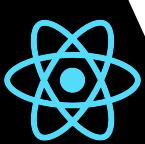
Agora, vamos estilizar nossa listagem



```
JS HabitosList.js ×
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 const HabitosList = props => {
5   const { habitos } = props;
6   const ret = habitos.map(habito => {
7     const { id, nome } = habito;
8     return <Text style={ styles.list } key={ id }>{ id } - { nome }</Text>;
9   });
10
11
12
13
14
15   )
16 };
17
18 const styles = StyleSheet.create({
19   list: {
20     padding: 40
21   }
22 });
23
24 export default HabitosList;
```

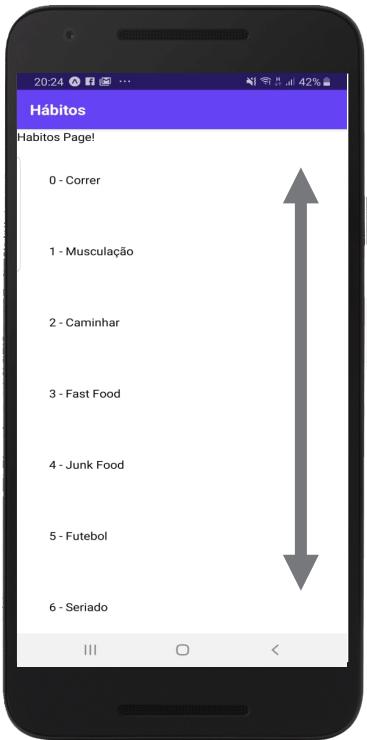
Tentem ver o item 7
Testem o "scroll" da tela

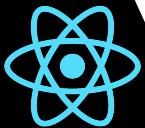
```
15   )
16 };
17
18 const styles = StyleSheet.create({
19   list: {
20     padding: 40
21   }
22 });
23
24 export default HabitosList;
```



REACT NATIVE

OPS!!!! NÃO TEMOS SCROLL



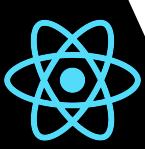


REACT NATIVE

Para ajustar o scroll vamos substituir o componente View por ScrollView

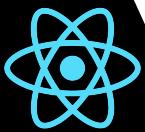
JS HabitosPage.js ×

```
1 import React from 'react';
2 import { Text, ScrollView } from 'react-native';
3 import HabitosList from '../components/HabitosList';
4
5 import habitosJson from '../../../../../habitost.json';
6
7 export default class HabitosPage extends React.Component {
8     constructor(props) {
9         super(props);
10
11         this.state = {
12             habitos: []
13         };
14     }
15
16     componentDidMount() {
17         this.setState({
18             habitos: habitosJson
19         });
20     }
21
22     render() {
23         return (
24             <ScrollView>
25                 <Text>Habitos Page!</Text>
26                 <HabitosList habitos={ this.state.habitos } />
27             </ScrollView>
28         );
29     }
30 }
```



REACT NATIVE

Vamos deixar a listagem apresentável



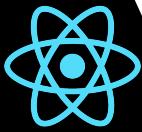
REACT NATIVE

Vamos deixar a listagem apresentável

JS HabitatsList.js x

```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 const HabitatsList = props => {
5     const { habitos } = props;
6     const ret = habitos.map(habito => {
7         const { id, nome } = habito;
8         return (
9             <View style={ styles.container } key={ id }>
10                 <Text style={ [styles.default, styles.id] }>{ id }</Text>
11                 <Text style={ [styles.default, styles.nome] }>{ nome }</Text>
12             </View>
13     );
14 });
15
16 return (
17     <View>
18         { ret }
19     </View>
20 )
21};
```

Usamos o conceito de
desestruturação para buscar id e
nome do hábito indexado



REACT NATIVE

Vamos deixar a listagem apresentável

JS HabitatsList.js x

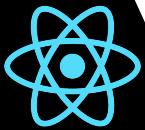
```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 const HabitatsList = props => {
5     const { habitos } = props;
6     const ret = habitos.map(habito => {
7         const { id, nome } = habito;
8         return (
9             <View style={ styles.container } key={ id }>
10                 <Text style={ [styles.default, styles.id] }>{ id }</Text>
11                 <Text style={ [styles.default, styles.nome] }>{ nome }</Text>
12             </View>
13         );
14     });
15
16     return (
17         <View>
18             { ret }
19         </View>
20     );
21 }
```

O retorno será mudado para uma View contendo dois textos:

- ID
- Nome

Também, criamos 4 estilos

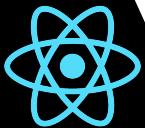
- 1 para a View
- 1 para determinado padrão
- 1 para o texto de ID
- 1 para o texto do Nome



REACT NATIVE

Vamos deixar a listagem apresentável

```
22
23 const styles = StyleSheet.create({
24   container: {
25     flex: 1,
26     flexDirection: "row",
27   },
28   default: {
29     padding: 20,
30     borderRadius: 4,
31     borderWidth: 0.5,
32     borderColor: '#d6d7da',
33     borderStyle: 'solid',
34   },
35   id: {
36     flex: 0.1,
37     textAlign: 'center',
38   },
39   nome: {
40     flex: 0.9,
41   }
42 });
43 });
44
45 export default HabitosList;
```



REACT NATIVE

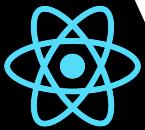
Vamos deixar a listagem apresentável

O container usará conceito de **Flex**

FlexBox é a técnica utilizada pelo react para construção de layouts de forma mais amigável

Com ele conseguimos distribuir mais facilmente os elementos em tela

```
22
23 const styles = StyleSheet.create({
24   container: {
25     flex: 1,
26     flexDirection: "row",
27   },
28   default: {
29     padding: 20,
30     borderRadius: 4,
31     borderWidth: 0.5,
32     borderColor: '#d6d7da',
33     borderStyle: 'solid',
34   },
35   id: {
36     flex: 0.1,
37     textAlign: 'center',
38   },
39   nome: {
40     flex: 0.9,
41   }
42 });
43 });
44
45 export default HabitosList;
```

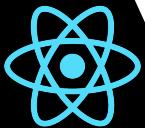


REACT NATIVE

Vamos deixar a listagem apresentável

Informamos que o container (View) terá o tamanho de flex = 1 e direção em linha (por padrão a direção é por coluna)

```
22
23 const styles = StyleSheet.create({
24   container: {
25     flex: 1,
26     flexDirection: "row",
27   },
28   default: {
29     padding: 20,
30     borderRadius: 4,
31     borderWidth: 0.5,
32     borderColor: '#d6d7da',
33     borderStyle: 'solid',
34   },
35   id: {
36     flex: 0.1,
37     textAlign: 'center',
38   },
39   nome: {
40     flex: 0.9,
41   }
42 });
43 });
44
45 export default HabitosList;
```

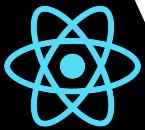


REACT NATIVE

Vamos deixar a listagem apresentável

Montamos um estilo default para não precisar replicar os dados

```
22
23 const styles = StyleSheet.create({
24   container: {
25     flex: 1,
26     flexDirection: "row",
27   },
28   default: {
29     padding: 20,
30     borderRadius: 4,
31     borderWidth: 0.5,
32     borderColor: '#d6d7da',
33     borderStyle: 'solid',
34   },
35   id: {
36     flex: 0.1,
37     textAlign: 'center',
38   },
39   nome: {
40     flex: 0.9,
41   }
42 });
43 });
44
45 export default HabitosList;
```

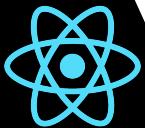


REACT NATIVE

Vamos deixar a listagem apresentável

Montamos um estilo para o ID, dizendo que usará 10% do tamanho do Flex do componente pai e com alinhamento de texto centralizado

```
22
23 const styles = StyleSheet.create({
24   container: {
25     flex: 1,
26     flexDirection: "row",
27   },
28   default: {
29     padding: 20,
30     borderRadius: 4,
31     borderWidth: 0.5,
32     borderColor: '#d6d7da',
33     borderStyle: 'solid',
34   },
35   id: {
36     flex: 0.1,
37     textAlign: 'center',
38   },
39   nome: {
40     flex: 0.9,
41   }
42 });
43 });
44
45 export default HabitosList;
```

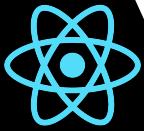


REACT NATIVE

Vamos deixar a listagem apresentável

Montamos um estilo para o Nome, dizendo que usará 90% do tamanho do Flex do componente pai

```
22
23 const styles = StyleSheet.create({
24   container: {
25     flex: 1,
26     flexDirection: "row",
27   },
28   default: {
29     padding: 20,
30     borderRadius: 4,
31     borderWidth: 0.5,
32     borderColor: '#d6d7da',
33     borderStyle: 'solid',
34   },
35   id: {
36     flex: 0.1,
37     textAlign: 'center',
38   },
39   nome: {
40     flex: 0.9,
41   }
42 });
43 });
44
45 export default HabitosList;
```



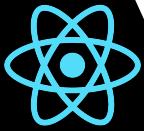
REACT NATIVE

Vamos deixar a listagem apresentável

JS HabitosList.js x

```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 const HabitosList = props => {
5   const { habitos } = props;
6   const ret = habitos.map(habito => {
7     const { id, nome } = habito;
8     return (
9       <View style={ styles.container } key={ id }>
10         <Text style={ [styles.default, styles.id] }>{ id }</Text>
11         <Text style={ [styles.default, styles.nome] }>{ nome }</Text>
12       </View>
13     );
14   });
15
16   return (
17     <View>
18       { ret }
19     </View>
20   )
21};
```

```
22
23   const styles = StyleSheet.create({
24     container: {
25       flex: 1,
26       flexDirection: "row",
27     },
28     default: {
29       padding: 20,
30       borderRadius: 4,
31       borderWidth: 0.5,
32       borderColor: '#d6d7da',
33       borderStyle: 'solid',
34     },
35     id: {
36       flex: 0.1,
37       textAlign: 'center',
38     },
39     nome: {
40       flex: 0.9,
41     }
42   });
43 });
44
45 export default HabitosList;
```



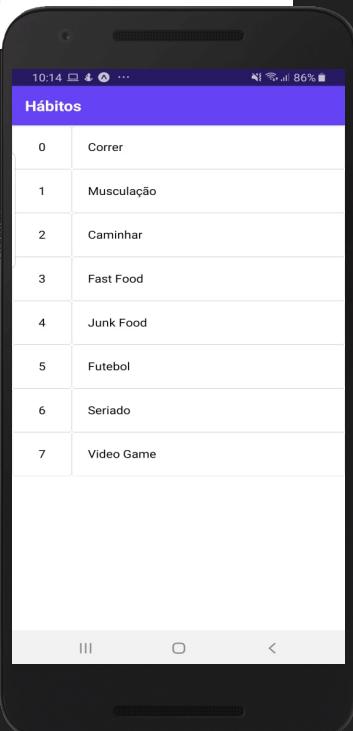
REACT NATIVE

Vamos deixar a listagem apresentável

JS HabitList.js ×

```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 const HabitosList = props => {
5     const { habitos } = props;
6     const ret = habitos.map(habito => {
7         const { id, nome } = habito;
8         return (
9             <View style={ styles.container } key={ id }>
10                <Text style={ [styles.default, styles.id] }>
11                    <Text style={ [styles.default, styles.no] }>
12                        </View>
13                );
14            });
15
16            return (
17                <View>
18                    { ret }
19                </View>
20            )
21        };

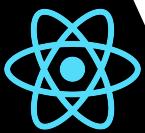
```



```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    flexDirection: "row",
  },
  default: {
    padding: 20,
    borderRadius: 4,
    borderWidth: 0.5,
    borderColor: '#d6d7da',
    borderStyle: 'solid',
  },
  id: {
    flex: 0.1,
    textAlign: 'center',
  },
  nome: {
    flex: 0.9,
  }
});

export default HabitosList;
```

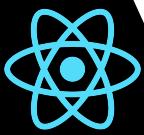
Criando LoginPage



REACT NATIVE

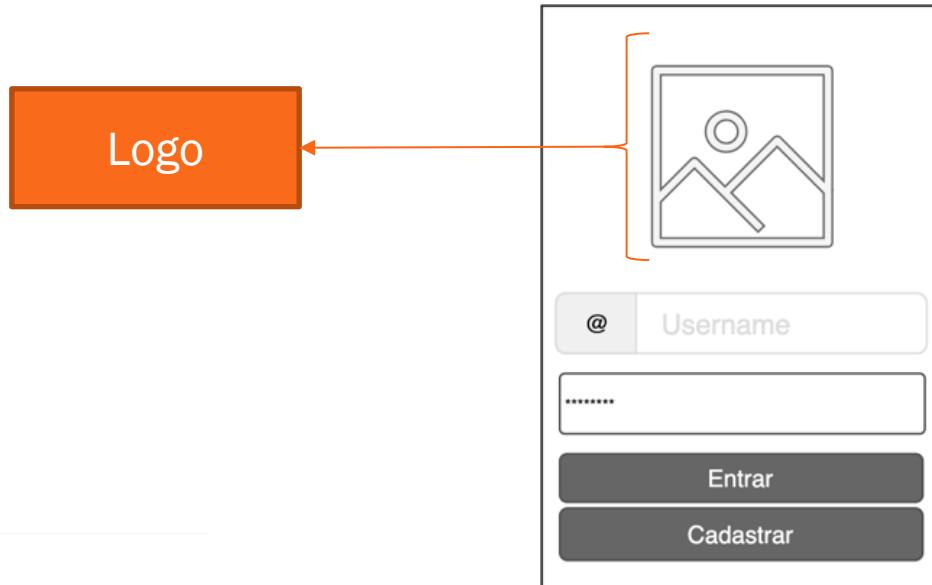
Para podermos deixar nosso app mais próximo do mundo real, vamos criar nossa tela de login

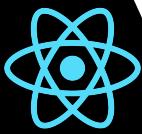




REACT NATIVE

Para podermos deixar nosso app mais próximo do mundo real, vamos criar nossa tela de login

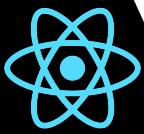




REACT NATIVE

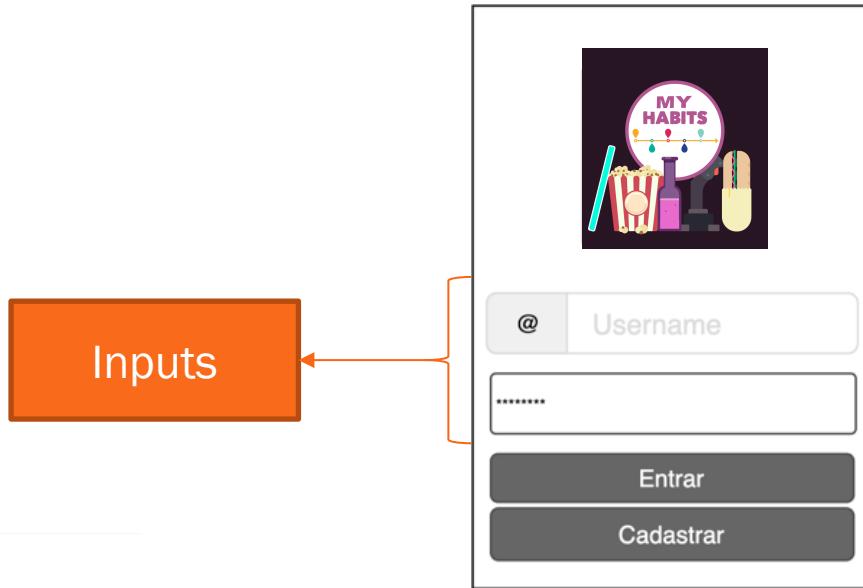
Para podermos deixar nosso app mais próximo do mundo real, vamos criar nossa tela de login

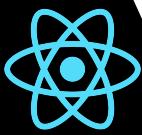




REACT NATIVE

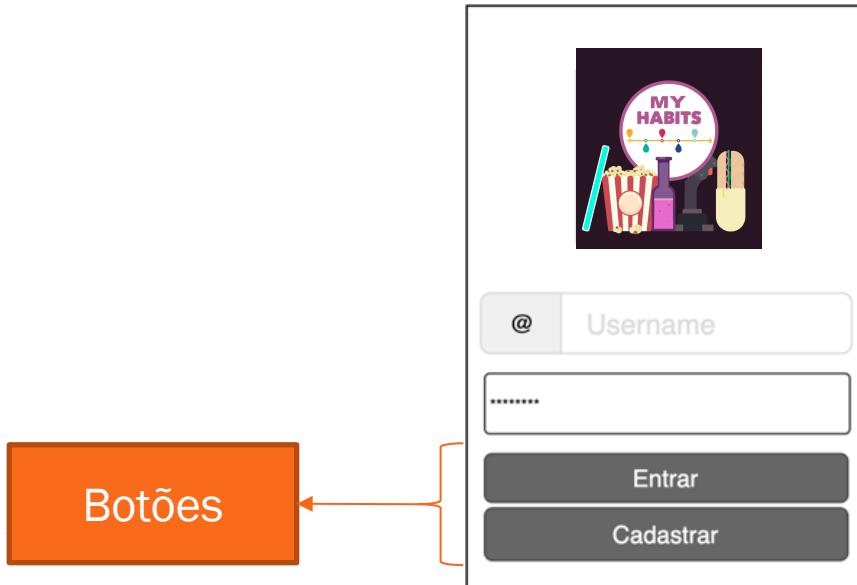
Para podermos deixar nosso app mais próximo do mundo real, vamos criar nossa tela de login

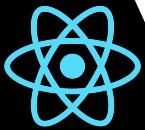




REACT NATIVE

Para podermos deixar nosso app mais próximo do mundo real, vamos criar nossa tela de login



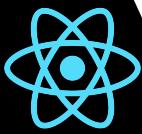


REACT NATIVE

Para podermos deixar nosso app mais próximo do mundo real, vamos criar nossa tela de login

Vamos criar arquivo para página de login

```
▶ src
  ▶ components
    JS HabitosList.js
  ▶ pages
    JS LoginPage.js
    JS HabitosPage.js
```



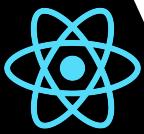
REACT NATIVE

Para podermos deixar nosso app mais próximo do mundo real, vamos criar nossa tela de login

Vamos criar arquivo para página de login

JS LoginPage.js X

```
1 import React from 'react';
2 import { TextInput, StyleSheet, Text, ScrollView } from 'react-native';
3
4 export default class LoginPage extends React.Component {
5
6 }
7
8 const styles = StyleSheet.create({
9
10});
```



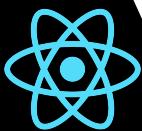
REACT NATIVE

Para podermos deixar nosso app mais próximo do mundo real, vamos criar nossa tela de login

Primeiro renomeamos a rota da listagem de Hábitos de "Main" para "Habitos"

```
JS App.js      ×
1 import {createAppContainer, createStackNavigator } from 'react-navigation';
2 import HabitosPage from './src/pages/HabitosPage';
3
4 const AppNavigator = createStackNavigator(
5   {
6     'Habitos': {
7       screen: HabitosPage,
8       navigationOptions: {
9         title: 'Hábitos',
10        headerTitleStyle: {
11          textAlign: 'left',
12          fontSize: 20,
13        },
14      }
15    }
16  )

```

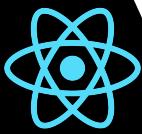


REACT NATIVE

Para podermos deixar nosso app mais próximo do mundo real, vamos criar nossa tela de login

Criamos uma nova rota na pilha de navegação para a tela de login

```
JS App.js      ×  
1 import {createAppContainer, createStackNavigator } from 'react-navigation';  
2 import HabitosPage from './src/pages/HabitosPage';  
3  
4 const AppNavigator = createStackNavigator(  
5   {  
6     'Login': {  
7       screen: LoginPage,  
8       navigationOptions: {  
9         header: null,  
10       }  
11     },  
12     'Habitos': {  
13       screen: HabitosPage,  
14     }  
15   }  
16 );  
17  
18 export default createAppContainer(AppNavigator);
```



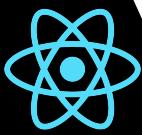
REACT NATIVE

Para podermos deixar nosso app mais próximo do mundo real, vamos criar nossa tela de login

Criamos uma nova rota na pilha de navegação para a tela de login

```
JS App.js      ×  
1 import {createAppContainer, createStackNavigator } from 'react-navigation';  
2 import HabitosPage from './src/pages/HabitosPage';  
3  
4 const AppNavigator = createStackNavigator(  
5   {  
6     'Login': {  
7       screen: LoginPage,  
8       navigationOptions: {  
9         header: null,  
10       }  
11     },  
12     'Habitos': {  
13       screen: HabitosPage,  
14     }  
15   },  
16   {  
17     initialRouteName: 'Login'  
18   }  
19 );  
20  
21 export default createAppContainer(AppNavigator);
```

Telas de login não tem Header
Deixamos em NULL

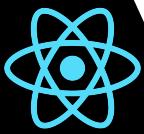


REACT NATIVE

Para podermos deixar nosso app mais próximo do mundo real, vamos criar nossa tela de login

Adicionamos dois TextInput para e-mail e senha

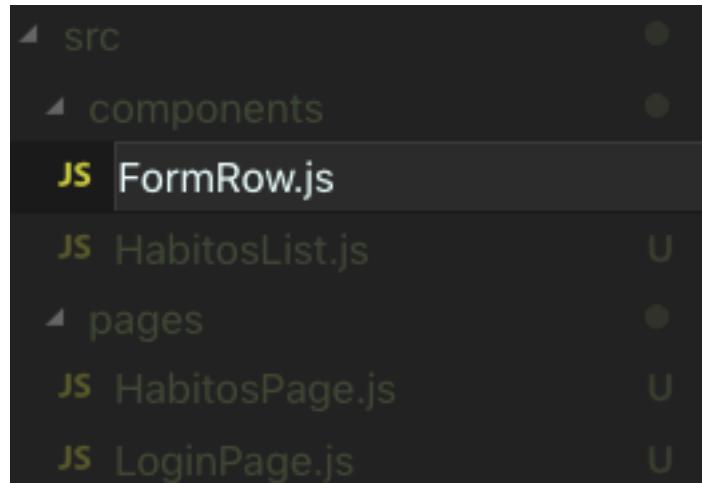
```
4  export default class LoginPage extends React.Component {  
5      render() {  
6          return (  
7              <ScrollView>  
8                  <TextInput placeholder="user@email.com" />  
9                  <TextInput placeholder="*****" />  
10             </ScrollView>  
11         )  
12     }  
13 }
```

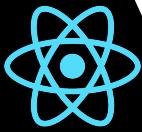


REACT NATIVE

Para podermos deixar nosso app mais próximo do mundo real, vamos criar nossa tela de login

Para melhor organização dos componentes em tela, vamos criar um componente para que possamos criar formulários padronizados na aplicação





REACT NATIVE

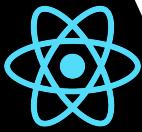
Para podermos deixar nosso app mais próximo do mundo real, vamos criar nossa tela de login

Para melhor organização dos componentes em tela, vamos criar um componente para que possamos criar formulários padronizados na aplicação

JS

LoginPage.js

```
1 import React from 'react';
2 import { TextInput, StyleSheet, Text, ScrollView } from 'react-native';
3 import FormRow from '../components/FormRow';
4
5 export default class LoginPage extends React.Component {
6   render() {
7     return (
8       <ScrollView>
9         <FormRow>
10           <TextInput placeholder="user@email.com" />
11         </FormRow>
12         <FormRow>
13           <TextInput placeholder="*****" />
14         </FormRow>
15       </ScrollView>
16     )
17   }
18 }
```



REACT NATIVE

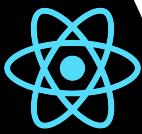
Para podermos deixar nosso app mais próximo do mundo real, vamos criar nossa tela de login

Para melhor organização dos componentes em tela, vamos criar um componente para que possamos criar formulários padronizados na aplicação

Analizando o código, podemos notar que o TextInput é um componente filho do FormRow

Então poderíamos ter vários filhos do componente FormRow

```
JS LoginPage.js ✘
1 import React from 'react';
2 import { TextInput, StyleSheet, Text, ScrollView } from 'react-native';
3 import FormRow from '../components/FormRow';
4
5 export default class LoginPage extends React.Component {
6   render() {
7     return (
8       <ScrollView>
9         <FormRow>
10           <TextInput placeholder="user@email.com" />
11         </FormRow>
12         <FormRow>
13           <TextInput placeholder="*****" />
14         </FormRow>
15       </ScrollView>
16     )
17   }
18 }
```



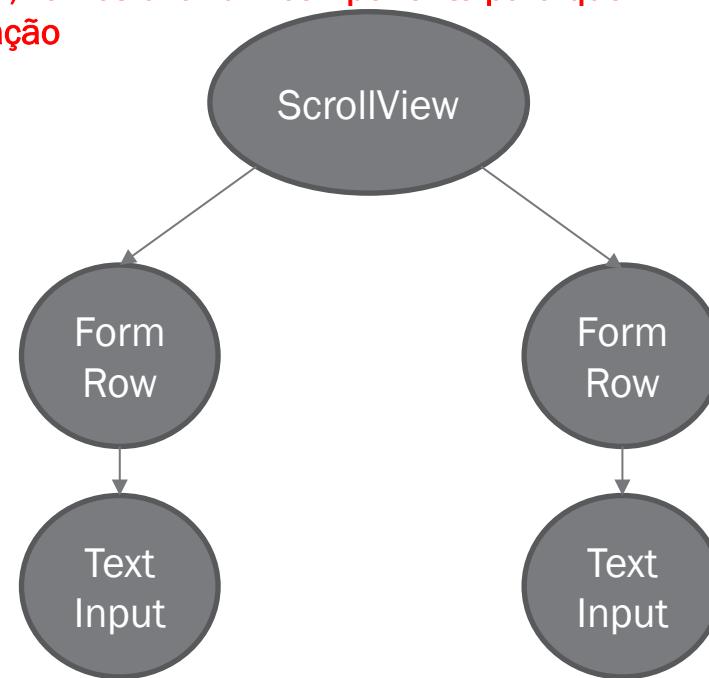
REACT NATIVE

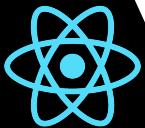
Para podermos deixar nosso app mais próximo do mundo real, vamos criar nossa tela de login

Para melhor organização dos componentes em tela, vamos criar um componente para que possamos criar formulários padronizados na aplicação

Analizando o código, podemos notar que o TextInput é um componente filho do FormRow

Então poderíamos ter vários filhos do componente FormRow



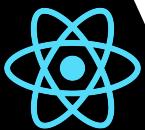


REACT NATIVE

O componente FormRow vai servir para mantermos um padrão de estilo aos componentes de um formulário

JS FormRow.js

```
1 import React from 'react';
2 import { View, StyleSheet } from 'react-native';
3
4 const FormRow = props => {
5     const { children } = props;
6
7     return (
8         <View style={ styles.container }>
9             { children }
10        </View>
11    )
12};
13
14 const styles = StyleSheet.create({
15     container: {
16         padding: 10,
17         backgroundColor: 'white',
18         marginTop: 5,
19         marginBottom: 5,
20         elevation: 0.5,
21     }
22 });
23
24 export default FormRow;
```

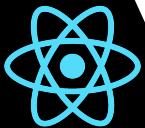


REACT NATIVE

Ele renderiza os componentes FILHOS
(children) deles dentro de uma view com estilo
padronizado

JS FormRow.js ×

```
1 import React from 'react';
2 import { View, StyleSheet } from 'react-native';
3
4 const FormRow = props => {
5     const { children } = props;
6
7     return (
8         <View style={ styles.container }>
9             { children }
10        </View>
11    )
12};
13
14 const styles = StyleSheet.create({
15     container: {
16         padding: 10,
17         backgroundColor: 'white',
18         marginTop: 5,
19         marginBottom: 5,
20         elevation: 0.5,
21     }
22 });
23
24 export default FormRow;
```

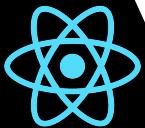


REACT NATIVE

Ele renderiza os componentes FILHOS
(children) deles dentro de uma view com estilo
padronizado

JS FormRow.js ×

```
1 import React from 'react';
2 import { View, StyleSheet } from 'react-native';
3
4 const FormRow = props => {
5     const { children } = props;
6
7     return (
8         <View style={ styles.container }>
9             { children }
10        </View>
11    )
12 };
13
14 const styles = StyleSheet.create({
15     container: {
16         padding: 10,
17         backgroundColor: 'white',
18         marginTop: 5,
19         marginBottom: 5,
20         elevation: 0.5,
21     }
22 });
23
24 export default FormRow;
```

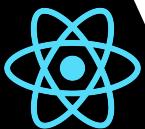


REACT NATIVE

Children é uma props que contem todos os componentes encapsulados nele

JS FormRow.js ×

```
1 import React from 'react';
2 import { View, StyleSheet } from 'react-native';
3
4 const FormRow = props => {
5     const { children } = props;
6
7     return (
8         <View style={ styles.container }>
9             { children }
10        </View>
11    )
12 };
13
14 const styles = StyleSheet.create({
15     container: {
16         padding: 10,
17         backgroundColor: 'white',
18         marginTop: 5,
19         marginBottom: 5,
20         elevation: 0.5,
21     }
22 });
23
24 export default FormRow;
```



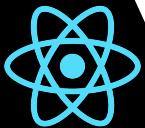
REACT NATIVE

Children é uma props que contem todos os componentes encapsulados nele

```
<ScrollView>
  <FormRow>
    <TextInput placeholder="user@email.com" />
  </FormRow>
  <FormRow>
    <TextInput placeholder="*****" />
  </FormRow>
</ScrollView>
```

JS FormRow.js ×

```
1 import React from 'react';
2 import { View, StyleSheet } from 'react-native';
3
4 const FormRow = props => {
5   const { children } = props;
6
7   return (
8     <View style={ styles.container }>
9       { children }
10    </View>
11  )
12};
13
14 const styles = StyleSheet.create({
15   container: {
16     padding: 10,
17     backgroundColor: 'white',
18     marginTop: 5,
19     marginBottom: 5,
20     elevation: 0.5,
21   }
22 });
23
24 export default FormRow;
```



REACT NATIVE

Criamos dois estilos

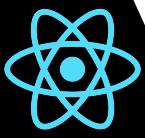
Um para a ScrollView e outro para os inputs

Ambos estilos servem para afastar o componente
das bordas da tela

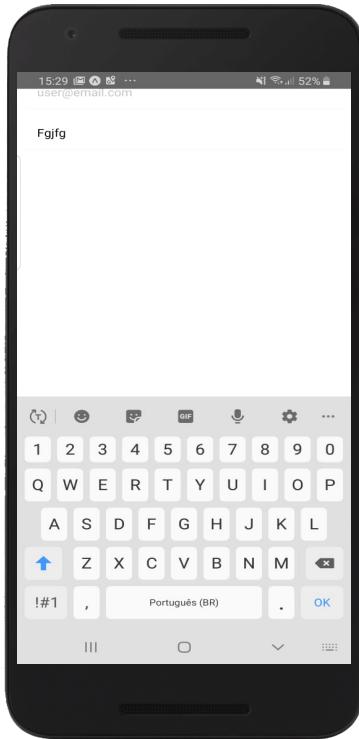
Isso melhora a experiência de usuário

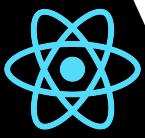
JS LoginPage.js ×

```
1 import React from 'react';
2 import { TextInput, StyleSheet, Text, ScrollView } from 'react-native';
3 import FormRow from '../components/FormRow';
4
5 export default class LoginPage extends React.Component {
6   render() {
7     return (
8       <ScrollView style={ styles.container }>
9         <FormRow>
10           <TextInput
11             style={ styles.input }
12             placeholder="user@email.com"
13           />
14         </FormRow>
15         <FormRow>
16           <TextInput
17             style={ styles.input }
18             placeholder="*****"
19           />
20         </FormRow>
21       </ScrollView>
22     )
23   }
24 }
25
26 const styles = StyleSheet.create({
27   container: {
28     paddingRight: 10,
29     paddingLeft: 10,
30   },
31   input: {
32     paddingLeft: 5,
33     paddingRight: 5,
34   },
35});
```

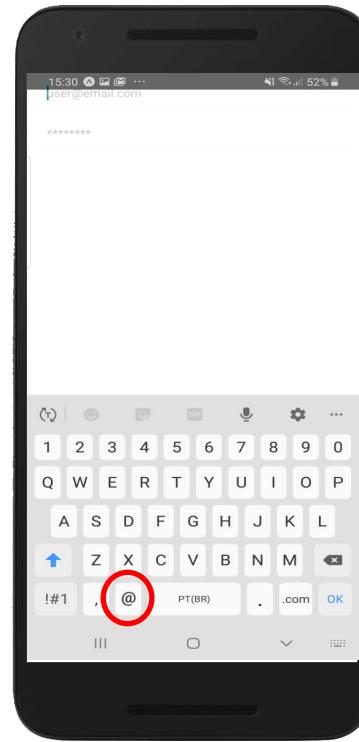
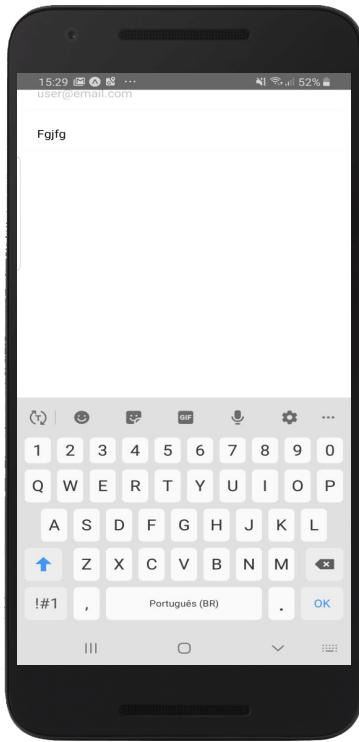


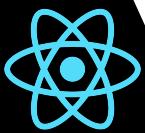
REACT NATIVE





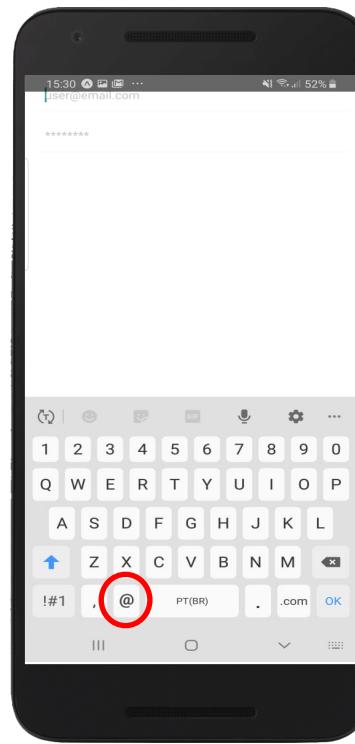
REACT NATIVE

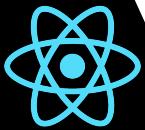




REACT NATIVE

```
<ScrollView style={ styles.container }>
  <FormRow>
    <TextInput
      style={ styles.input }
      placeholder="user@email.com"
      keyboardType="email-address"
    />
  </FormRow>
```





REACT NATIVE

```
<ScrollView style={ styles.container }>
  <FormRow>
    <TextInput
      style={ styles.input }
      placeholder="user@email.com"
      keyboardType="email-address"
    />
  </FormRow>
```

keyboardType

Determines which keyboard to open, e.g. `numeric`.

See screenshots of all the types [here](#).

The following values work across platforms:

- `default`
- `number-pad`
- `decimal-pad`
- `numeric`
- `email-address`
- `phone-pad`

iOS Only

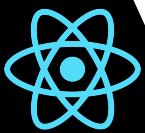
The following values work on iOS only:

- `ascii-capable`
- `numbers-and-punctuation`
- `url`
- `name-phone-pad`
- `twitter`
- `web-search`

Android Only

The following values work on Android only:

- `visible-password`

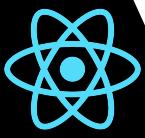


REACT NATIVE

Esconde a senha

```
<FormRow>
  <TextInput
    style={ styles.input }
    placeholder="*****"
    secureTextEntry
  />
</FormRow>
```

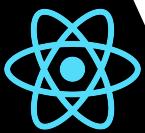
Componentes Controlados



REACT NATIVE

Um conceito interessante do ReactNative é o uso de Controlled Component

Para esse conceito, cria-se estados para cada componente



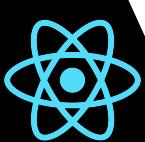
REACT NATIVE

Um conceito interessante do ReactNative é o uso de Controlled Component

Para esse conceito, cria-se estados para cada componente

Sempre que usarmos states, temos que fazer sua declaração no constructor

```
5  ↳ export default class LoginPage extends React.Component {  
6    ↳   constructor(props) {  
7      ↳     super(props);  
8  
9    ↳     this.state = {  
10       ↳       email: '',  
11       ↳       senha: ''  
12     ↳   }  
13   ↳ }
```



REACT NATIVE

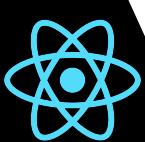
Um conceito interessante

Para esse conceito, cria-se

Para usarmos states em componentes controlados usamos *data binding*

Conceito para manter sincronia entre os dados e suas estruturas de controle

```
15 render() {
16     return (
17         <ScrollView style={ styles.container }>
18             <FormRow>
19                 <TextInput
20                     style={ styles.input }
21                     placeholder="user@email.com"
22                     keyboardType="email-address"
23                     value={ this.state.email } />
24             </FormRow>
25             <FormRow>
26                 <TextInput
27                     style={ styles.input }
28                     placeholder="*****"
29                     secureTextEntry
30                     value={ this.state.senha } />
31             </FormRow>
32         </ScrollView>
```



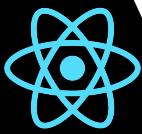
REACT NATIVE

Um conceito interessante

Para esse conceito, cria-se o

Como o react native trabalha com alteração dinâmica, sempre que o valor dos states senha e email mudarem, o componente terá o valor alterado

```
15 render() {
16     return (
17         <ScrollView style={ styles.container }>
18             <FormRow>
19                 <TextInput
20                     style={ styles.input }
21                     placeholder="user@email.com"
22                     keyboardType="email-address"
23                     value={ this.state.email } />
24             </FormRow>
25             <FormRow>
26                 <TextInput
27                     style={ styles.input }
28                     placeholder="*****"
29                     secureTextEntry
30                     value={ this.state.senha } />
31             </FormRow>
32         </ScrollView>
```



REACT NATIVE

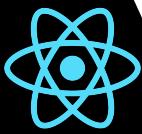
Um conceito interessante do ReactNative é o uso de Controlled Component

Para esse conceito, cria-se estados para cada componente

```
15  onChangeHandler(field, value) {  
16      this.setState({ [field]: value });  
17  }  
18  
19  render() {  
20      return (  
21          <ScrollView style={ styles.container }>  
22              <FormRow>  
23                  <TextInput  
24                      style={ styles.input }  
25                      placeholder="user@email.com"  
26                      keyboardType="email-address"  
27                      value={ this.state.email }  
28                      onChangeText={ value => this.onChangeHandler('email', value) }  
29                  />  
30          </FormRow>
```

Criamos o método onChangeHandler com parâmetro de field e valor

Será utilizado em todos os campos para alterar o valor do state



REACT NATIVE

Um conceito interessante do ReactNative é o uso de Controlled Component

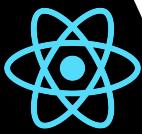
Para esse conceito, cria-se estados para cada componente

```
15  onChangeHandler(field, value) {  
16      this.setState({ [field]: value });  
17  }  
18  
19  render() {  
20      return (  
21          <ScrollView style={ styles.container }>  
22              <FormRow>  
23                  <TextInput  
24                      style={ styles.input }  
25                      placeholder="user@email.com"  
26                      keyboardType="email-address"  
27                      value={ this.state.email }  
28                      onChangeText={ value => this.onChangeHandler('email', value) }  
29                  />  
30          </FormRow>
```

Criamos o método onChangeHandler com parâmetro de field e valor

Será utilizado em todos os campos para alterar o valor do state

Quando houver alteração de texto o método será executado



REACT NATIVE

Um conceito interessante do ReactNative é o uso de Controlled Component

Para esse conceito, cria-se estados para cada componente

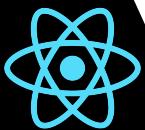
```
15  onChangeHandler(field, value) {  
16      this.setState({ [field]: value });  
17  }  
18  Conceito  
19  Computer  
20  Property  
21  
22  render() {  
23      return (  
24          <ScrollView style={ styles.container }>  
25              <FormRow>  
26                  <TextInput  
27                      style={ styles.input }  
28                      placeholder="user@email.com"  
29                      keyboardType="email-address"  
30                      value={ this.state.email }  
31                      onChangeText={ value => this.onChangeHandler('email', value) }  
32                  />  
33              </FormRow>
```

Criamos o método onChangeHandler com parâmetro de field e valor

Será utilizado em todos os campos para alterar o valor do state

Quando houver alteração de texto o método será executado

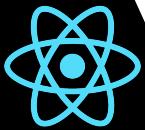
Botões e Ações



REACT NATIVE

A tela de login deve conter forma de Login e forme de Cadastrar

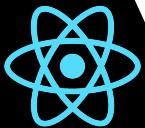
```
14
15    onChangeHandler(field, value) {
16        this.setState({ [field]: value });
17    }
18
19    login() {
20    }
21
22    solicitaCadastro() {
23    }
24
25    render() {
26        return (
27            <ScrollView>
28                <FormRow>
29                    <Text>Login</Text>
30                    <Input
31                        type='text'
32                        placeholder='Email'
33                        secureTextEntry={true}
34                        value={ this.state.senha }
35                        onChangeText={ value => this.onChangeHandler('senha', value) }
36                    />
37                </FormRow>
38
39                <FormRow>
40                    <Text>Senha</Text>
41                    <Input
42                        type='password'
43                        placeholder='Senha'
44                        secureTextEntry={true}
45                        value={ this.state.senha }
46                        onChangeText={ value => this.onChangeHandler('senha', value) }
47                    />
48                </FormRow>
49
50                <Button
51                    title='ENTRAR'
52                    color='#6542f4'
53                    onPress={()=> this.login() } />
54
55                <Button
56                    title='CADASTRE-SE'
57                    color='#a08af7'
58                    onPress={()=> this.solicitaCadastro() } />
59            </FormRow>
60        )
61    </ScrollView>
62    )
63
64 }
```



REACT NATIVE

A tela de login deve conter forma de Login e forme de Cadastrar

```
14
15     onChangeHandler(field, value) {
16         this.setState({ [field]: value });
17     }
18
19     login() {
20
21     }
22
23     solicitaCadastro() {
24
25     }
26
27     render() {
28         return (
29             <FormRow>
30                 <Text>Login</Text>
31                 <Input
32                     type='text'
33                     placeholder='Email'
34                     secureTextEntry={true}
35                     value={ this.state.senha }
36                     onChangeText={ value => this.onChangeHandler('senha', value) }
37                 />
38             </FormRow>
39
40             <FormRow>
41                 <Text>Senha</Text>
42                 <Input
43                     type='password'
44                     placeholder='Senha'
45                     secureTextEntry={true}
46                     value={ this.state.senha }
47                     onChangeText={ value => this.onChangeHandler('senha', value) }
48                 />
49             </FormRow>
50
51             <Button
52                 title='ENTRAR'
53                 color='#6542f4'
54                 onPress={()=> this.login() } />
55
56             <Button
57                 title='CADASTRE-SE'
58                 color='#a08af7'
59                 onPress={()=> this.solicitaCadastro() } />
60         </ScrollView>
61     )
62     )
63     )
64 }
```



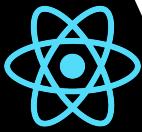
REACT NATIVE

A tela de login deve conter forma de Login e forme de Cadastrar

```
14
15  onChangeHandler(field, value) {
16      this.setState({ [field]: value });
17  }
18
19  login() {
20  }
21
22
23  solicitaCadastro() {
24  }
25
26
27  render() {
28      return (
29          <FormRow>
30              <Text>Login</Text>
31              <Input
32                  secureTextEntry={true}
33                  value={ this.state.senha }
34                  onChangeText={ value => this.onChangeHandler('senha', value) }
35              />
36          </FormRow>
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64      )
    )
```

The code snippet shows a React Native component with several sections highlighted by different colored boxes:

- A yellow box highlights the `login()` method.
- An orange box highlights the `solicitaCadastro()` method.
- A yellow box highlights the button for logging in, which has the title "ENTRAR".
- An orange box highlights the button for requesting a registration, which has the title "CADASTRE-SE".



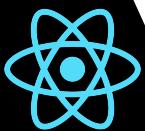
REACT NATIVE

A tela de login deve conter forma de Login e forme de Cadastrar

Estilizamos a área dos botões adicionando eles em uma View cada um

```
50      <View style={ styles.btn }>
51        <Button
52          title='ENTRAR'
53          color='#6542f4'
54          onPress={()=> this.login() }>
55        />
56      </View>
57
58      <View style={ styles.btn }>
59        <Button
60          title='CADASTRE-SE'
61          color='#a08af7'
62          onPress={()=> this.solicitaC...>
63        />
64      </View>
```

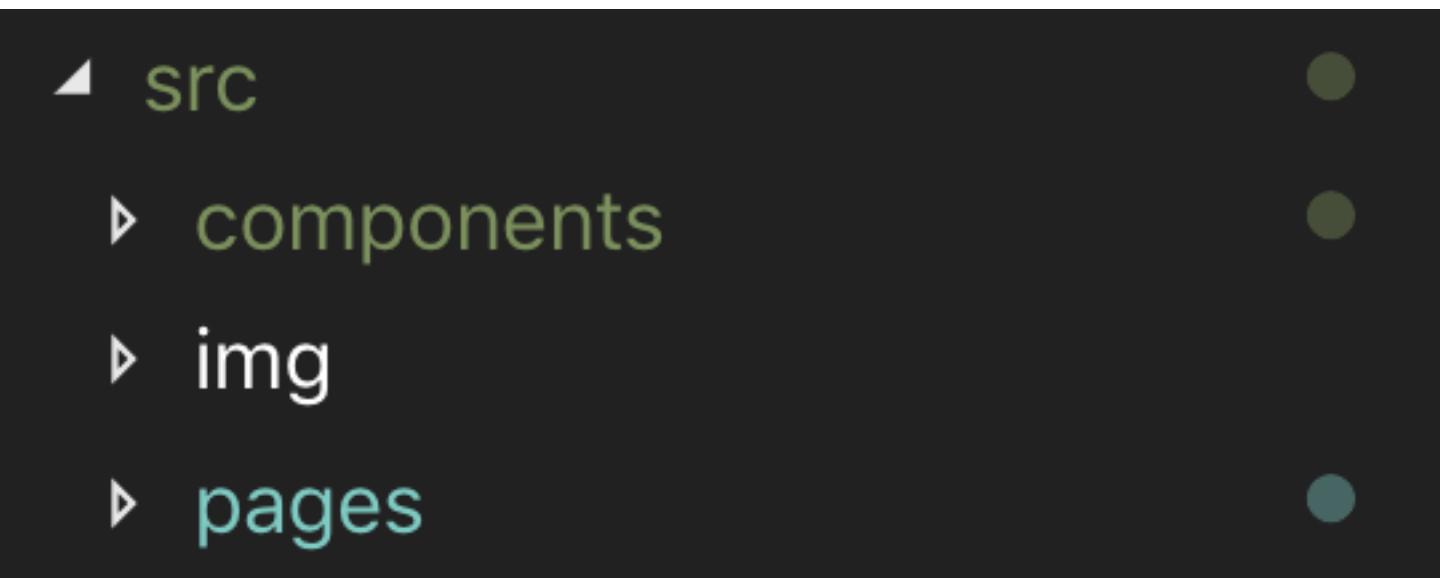
```
70      const styles = StyleSheet.create({
71        container: {
72          paddingRight: 10,
73          paddingLeft: 10,
74        },
75        input: {
76          paddingLeft: 5,
77          paddingRight: 5,
78        },
79        btn: {
80          paddingTop: 20,
81          fontSize: 11,
82        },
83      });
84    }
```

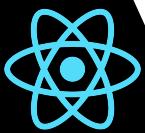


REACT NATIVE

Para a tela de login começar a ganhar forma de fato, vamos adicionar um logo ao projeto

Criamos a pasta "img" dentro de src

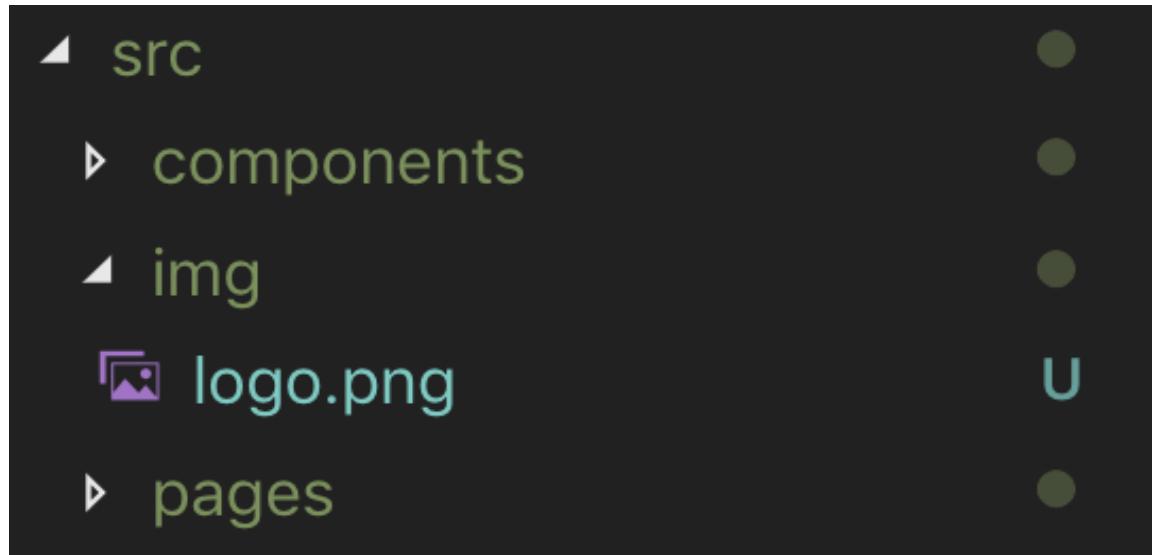


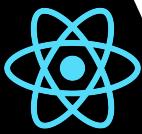


REACT NATIVE

Para a tela de login começar a ganhar forma de fato, vamos adicionar um logo ao projeto

Adicionamos o logo.png na pasta criada





REACT NATIVE

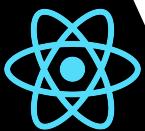
Para a tela de login começar a ganhar forma de fato, vamos adicionar um logo ao projeto

Vamos ajustar os imports do LoginPage.js e adicionar o uso do logo

JS LoginPage.js ×

```
1 import React from 'react';
2 import { TextInput, StyleSheet, View, ScrollView, Button, Image } from 'react-native';
3 import FormRow from '../components/FormRow';
```

```
30 <ScrollView style={ styles.container }>
31   <View style={styles.logoView}>
32     <Image
33       source={require('../img/logo.png')}
34         style={ styles.logo }
35     />
36   </View>
37 
```



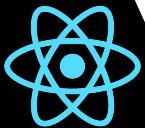
REACT NATIVE

Para a tela de login começar a ganhar forma de fato, vamos adicionar um logo ao projeto

Vamos ajustar os imports do LoginPage.js e adicionar o uso do logo

```
logo: {  
  aspectRatio: 1,  
  resizeMode: 'center',  
  width: 400,  
  height: 400,  
},  
logoView: {  
  justifyContent: "center",  
  alignItems: "center",  
},
```

```
new, ScrollView, Button, Image } from 'react-native';  
  </FormRow>;  
  
<ScrollView style={ styles.container }>  
  <View style={styles.logoView}>  
    <Image  
      source={require('../img/logo.png')}  
      style={ styles.logo }  
    />  
  </View>
```



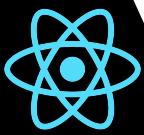
REACT NATIVE

Para a tela de login começar a ganhar forma de fato, vamos adicionar um logo ao projeto

Vamos ajustar os imports do LoginPage.js e adicionar o uso do logo

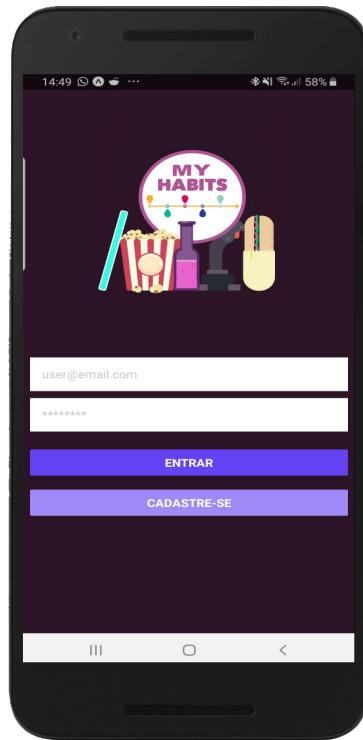
JS LoginPage.js ×

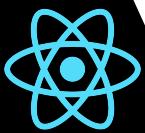
```
1 import React from 'react';
2 import { Text, View } from 'react-native';
3 import FormRow from './FormRow';
4
5 const styles = StyleSheet.create({
6   container: {
7     backgroundColor: '#2C1526',
8   },
9   input: {
10 }
```



REACT NATIVE

Para a tela de login começar a ganhar forma de fato, vamos adicionar um logo ao projeto

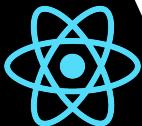




REACT NATIVE

Vamos configurar adicionar log nas ações

```
19     login() {  
20         console.log("Tentando efetuar login!");  
21     }  
22  
23     solicitaCadastro() {  
24         console.log("Solicitado cadastro!");  
25     }  
26 }
```

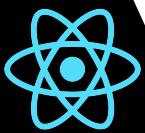


REACT NATIVE

Vamos configurar adicionar log nas ações

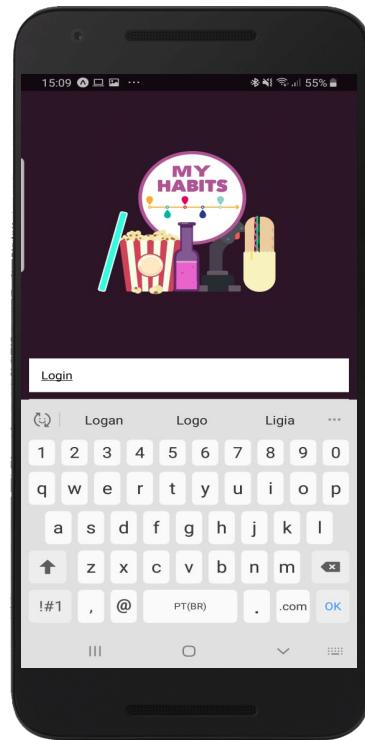
```
19    login() {  
20        console.log("Tentando efetuar login!");  
21    }  
22  
23    solicitaCadastro() {  
24        console.log("Solicitado cadastro!");  
25    }
```

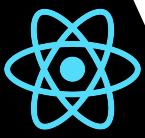
```
[15:02:05] Tentando efetuar login!  
[15:02:07] Solicitado cadastro!  
[15:02:08] Solicitado cadastro!  
[15:02:08] Tentando efetuar login!  
[15:02:09] Solicitado cadastro!  
[15:02:09] Tentando efetuar login!  
[15:02:10] Solicitado cadastro!  
[15:02:10] Tentando efetuar login!  
[15:02:10] Tentando efetuar login!  
[15:02:12] Tentando efetuar login!  
[15:02:12] Tentando efetuar login!  
[15:02:12] Solicitado cadastro!  
[15:02:12] Solicitado cadastro!  
[15:02:13] Tentando efetuar login!  
[15:02:13] Solicitado cadastro!  
[15:02:13] Tentando efetuar login!  
[15:02:13] Tentando efetuar login!  
[15:02:15] Solicitado cadastro!  
[15:02:15] Solicitado cadastro!  
[15:02:15] Solicitado cadastro!  
[15:02:15] Tentando efetuar login!  
[15:02:15] Solicitado cadastro!  
[15:02:15] Tentando efetuar login!
```



REACT NATIVE

Temos um problema, quando abrimos o teclado, a tela fica congelada

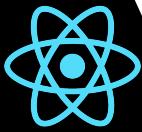




REACT NATIVE

Precisamos de algo assim...





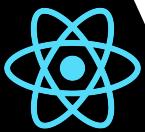
REACT NATIVE

Precisamos de algo assim... KeyboardAvoidingView

JS LoginPage.js ✘

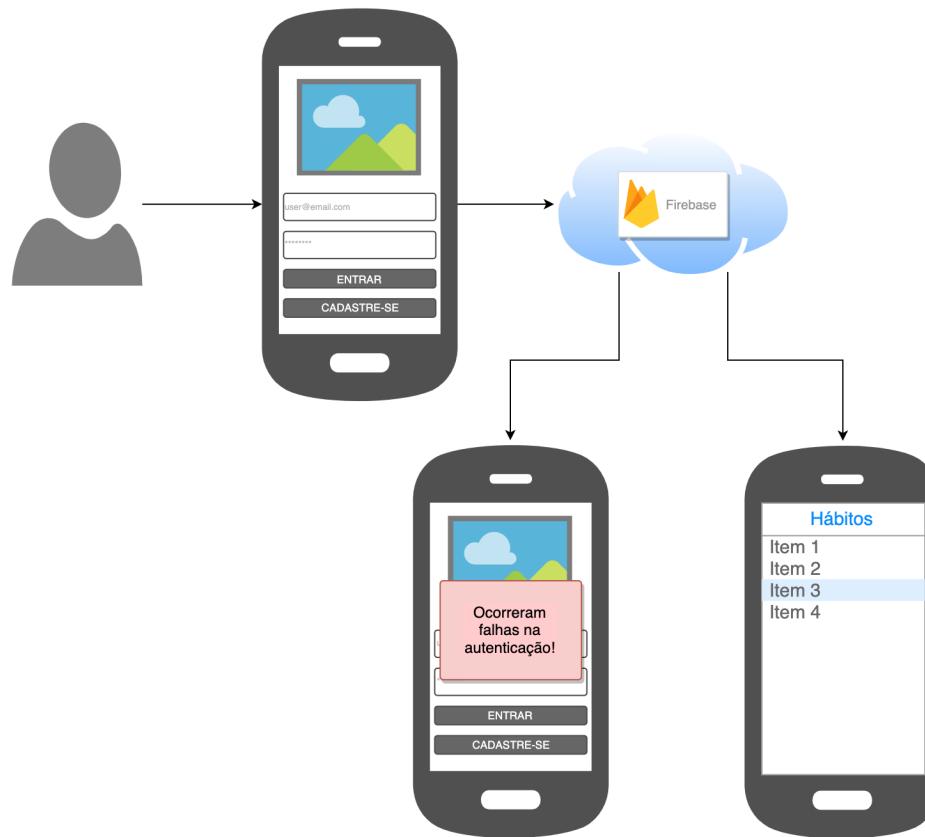
```
1 import React from 'react';
2 import { TextInput, StyleSheet, View, ScrollView, Button, Image, KeyboardAvoidingView } from 'react-native';
3 import FormRow from '../components/FormRow';
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27 render() {
28     return (
29         <KeyboardAvoidingView behavior="padding" enabled style={{flex: 1}}>
30             <ScrollView style={ styles.container }>...
31             </KeyboardAvoidingView>
32         )
33     }
34 }
```

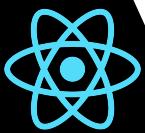
Comunicando com Firebase



REACT NATIVE

Fluxo de autenticação





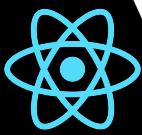
REACT NATIVE

Fluxo de autenticação

Para isso, vamos trabalhar com o Firebase

<https://firebase.google.com>

The screenshot shows the official Firebase website. At the top, there is a navigation bar with the Firebase logo, a 'Mais' dropdown, a search bar labeled 'Pesquisa', a 'Language' dropdown, a 'Ir para o console' button, and a 'Fazer login' button. The main visual features a blue background with a stylized illustration of two people (a man and a woman) sitting cross-legged on a surface, interacting with a laptop. To the left, large white text reads 'Firebase helps mobile and web app teams succeed'. At the bottom, there are two buttons: 'Primeiros passos' and 'Watch the video'.



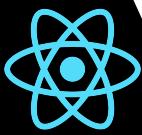
REACT NATIVE

Fluxo de autenticação

Para isso, vamos trabalhar com o Firebase

Vamos em "Primeiros passos"

A screenshot of the Firebase homepage. At the top, there is a navigation bar with icons for Firebase, Google Analytics, Google Cloud Storage, and Google Cloud Functions, followed by links for 'Mais', 'Pesquisa', 'Language', 'Ir para o console', and 'Fazer login'. The main content area features a large, semi-transparent background image of a person sitting at a desk with a laptop, looking up thoughtfully. Overlaid on this image is the text 'Firebase helps mobile and web app teams succeed'. At the bottom left, there is a white button with a blue border containing the text 'Primeiros passos'. To its right, there is another button with a blue border and a play icon, labeled 'Watch the video'. The overall theme is professional and user-centered.



REACT NATIVE

Fluxo de autenticação

Olá! Este é o Firebase

Ferramentas do Google para desenvolver ótimos aplicativos, interagir com seus usuários e ganhar mais por meio dos anúncios para dispositivos móveis.

[? Saiba mais](#) [≡ Documentação](#) [□ Suporte](#)

Adicionamos um novo projeto

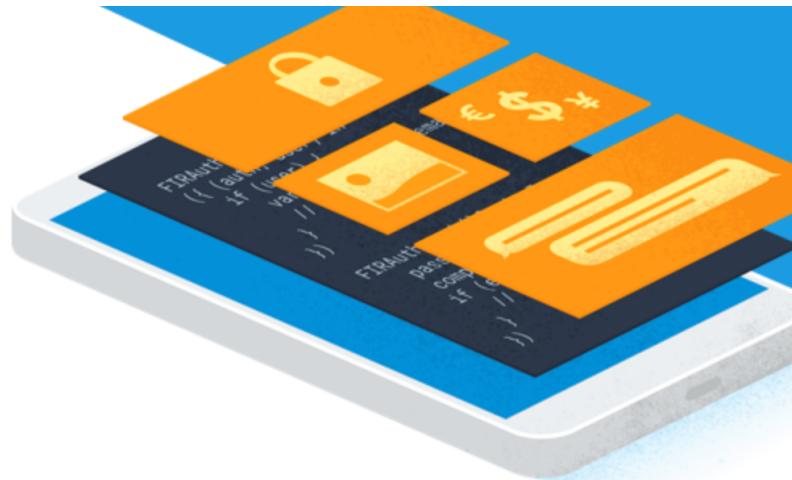
Seus projetos que usam o Firebase

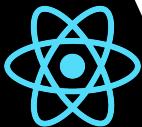


Adicionar projeto



Explorar projeto de demonstração





REACT NATIVE

Fluxo de autenticação

Para isso, vamos trabalhar com o Firebase

O nome do projeto coloquem
como MyHabitTimeline<Sobrenome>

Isso fará seu projeto ser único
Ou dará erro!

Adicionar um projeto X

Nome do projeto + +

MyHabitTimeline ▼

Dica: os projetos abrangem apps de várias plataformas (?)

Código do projeto (?)

myhabittimeline (?)

Locais (?)

Estados Unidos (Analytics) (?)

nam5 (us-central) (Cloud Firestore) (?)

Usar as configurações padrão para compartilhar os dados do Google Analytics para Firebase

✓ Compartilhar seus dados do Analytics com todos os recursos do Firebase

✓ Compartilhe seus dados do Analytics com o Google para melhorar os produtos e serviços da empresa

✓ Compartilhe seus dados do Analytics com o Google para ativar o suporte técnico

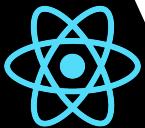
✓ Compartilhe seus dados do Analytics com o Google para ativar o Comparativo de mercado

✓ Compartilhe seus dados do Analytics com os especialistas em contas do Google

Aceito os [termos do controlador](#). Isso é necessário ao compartilhar dados do Analytics para melhorar os produtos e serviços do Google. [Saiba mais](#)

Confirmo que estou usando os serviços do Firebase no meu app e concordo com os aplicáveis [atualizados](#).

Cancelar Criar projeto



REACT NATIVE

Fluxo de autenticação

Para isso, vamos trabalhar com o Firebase

O nome do projeto coloquem
como MyHabitTimeline<Sobrenome>

Adicionar um projeto

Nome do projeto

MyHabitTimeline

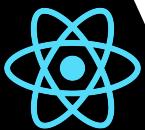
Android + iOS + </>

Dica: os projetos abrangem apps de várias plataformas

Concluindo...

MyHabitTimelineMadalozzo

Criar projeto



REACT NATIVE

Fluxo de autenticação

Para isso, vamos trabalhar com o Firebase

Projeto MyHabitTimeline<Sobrenome>
criado

Adicionar um projeto

Nome do projeto

MyHabitTimeline

Android + iOS + </>

Dica: os projetos abrangem apps de várias plataformas

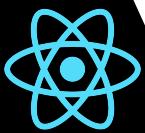
MyHabitTimelineMadalozzo

Seu novo projeto está pronto

Continuar

Criar projeto

Cancelar

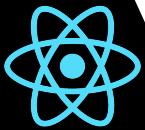


REACT NATIVE

Fluxo de autenticação

Para isso, vamos trabalhar com o Firebase

O projeto está pronto para usar...



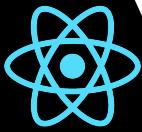
REACT NATIVE

Fluxo de autenticação

Para isso, vamos trabalhar com o Firebase

O projeto está pronto para usar...

The screenshot shows the Firebase console interface. On the left, there's a sidebar with navigation links: Project Overview, Desenvolver (Authentication, Database, Storage, ...), Qualidade (Crashlytics, Performance, Test Lab), Analytics (Dashboard, Events, Conversions, ...), and Ampliar (Predictions, A/B Testing, Cloud Mes...). The main area is titled "Comece adicionando o Firebase ao seu aplicativo" and features sections for iOS, Android, and a button to "Adicione um app para começar". Below this, a callout box says "Armazene e sincronize dados de app em milissegundos" and shows two cards: "Auth" (Autenticar e gerenciar usuários) and "Cloud Firestore" (A próxima geração do Realtime Database).



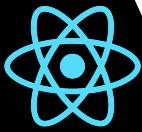
REACT NATIVE

Fluxo de autenticação

Para isso, vamos trabalhar com o Firebase

Vamos entrar na tela de configuração de autenticação

The screenshot shows the Firebase console's main interface. On the left, there's a sidebar with sections for Project Overview, Desenvolver (Developer), Qualidade (Quality), Analytics, and Ampliar (Expand). The main content area is titled "Comece adicionando o Firebase ao seu aplicativo" (Start by adding Firebase to your app) and features a section for "Auth" with the sub-section "Autenticar e gerenciar usuários" (Authenticate and manage users). Below this, there are sections for "Cloud Firestore" and "Cloud Functions". A large illustration of two people using mobile devices is visible in the background.



REACT NATIVE

Fluxo de autenticação

Para isso, vamos trabalhar com o Firebase

Vamos entrar na tela de configuração de autenticação

The screenshot shows the Firebase console's authentication interface. At the top, there's a navigation bar with 'Project Overview' and other tabs like 'Desenvolvedor', 'Qualidade', and 'Analytics'. Below the navigation, a section titled 'Comece adicionando o Firebase ao seu aplicativo' provides instructions for integrating the SDK. A large search bar at the top allows users to search for users by email or phone number. Below it, a table lists user information with columns for 'Identificador', 'Provedores', 'Criado em', 'Conectado', and 'UID do usuário'. A blue button labeled 'Configurar método de login' is visible at the bottom. On the right side of the interface, there are two floating cards: one showing a person icon with the text 'Autenticar e gerenciar usuários a partir de uma variedade de provedores sem código do lado do servidor' and another showing a magnifying glass icon with the text 'Ver os documentos'.

Comece adicionando o Firebase ao seu aplicativo

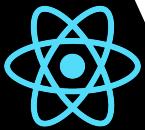
Pesquise por endereço de e-mail, número de telefone ou UID do usuário

Identificador Provedores Criado em Conectado UID do usuário ↑

Autenticar e gerenciar usuários a partir de uma variedade de provedores sem código do lado do servidor

Saiba mais Ver os documentos

Configurar método de login



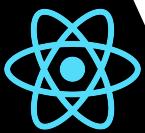
REACT NATIVE

Fluxo de autenticação

Para isso, vamos trabalhar com o F

Para este projeto, vamos habilitar apenas login com email e senha

Provedores de login		
Provedor		Status
✉ E-mail/senha		Desativado
📞 Smartphone		Desativado
⚡ Google		Desativado
🎮 Play Games		Desativado
👾 Game Center	Beta	Desativado
🌐 Facebook		Desativado
🐦 Twitter		Desativado
🐙 GitHub		Desativado
Yahoo		Desativado
Microsoft		Desativado
👤 Anônimo		Desativado

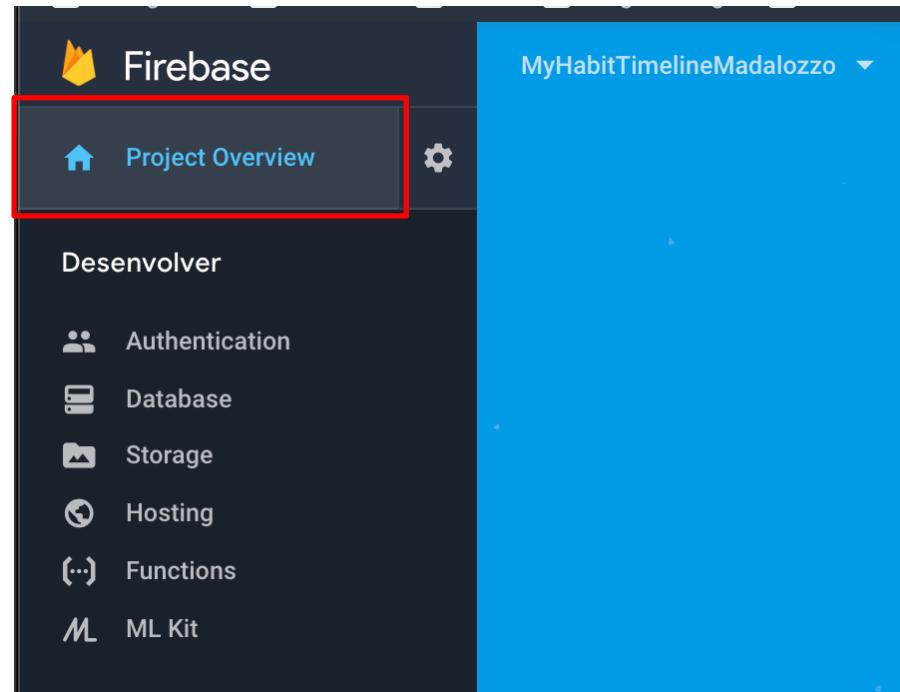


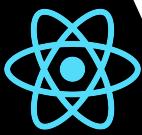
REACT NATIVE

Fluxo de autenticação

Para isso, vamos trabalhar com o Firebase

Agora, devemos configurar o App na plataforma





REACT NATIVE

Fluxo de autenticação

Para isso, vamos trabalhar com o Firebase

Agora, devemos configurar o App na plataforma

MyHabitTimelineMadalozzo

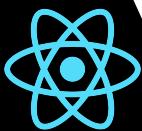
Plano Spark

Comece adicionando o Firebase ao seu aplicativo

Nosso SDK principal tem acesso à maioria dos recursos do Firebase e inclui o Firebase Analytics de apps para iOS e Android

iOS | Android | </> |

Adicione um app para começar



REACT NATIVE

Fluxo de autenticação

Para isso, vamos trabalhar com o Firebase

Agora, devemos configurar o App na plataforma

Para configurar temos que escolher qual o estilo da aplicação

- iOS
- Android
- Web

MyHabitTimelineMadalozzo

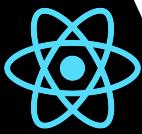
Plano Spark

Comece adicionando o Firebase ao seu aplicativo

Nosso SDK principal tem acesso à maioria dos recursos do Firebase e inclui o Firebase Analytics de apps para iOS e Android

iOS | Android | </> |

Adicione um app para começar



REACT NATIVE

Fluxo de autenticação

Para isso, vamos trabalhar com o Firebase

Agora, devemos configurar o App na plataforma

Para configurar temos que escolher qual o estilo da aplicação

- iOS
- Android
- Web

MyHabitTimelineMadalozzo

Plano Spark

Comece adicionando o Firebase ao seu aplicativo

Nosso SDK principal tem acesso à maioria dos recursos do Firebase e inclui o Firebase Analytics de apps para iOS e Android

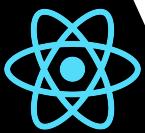
iOS

</>

Android

Adicione um app para começar

Angular



REACT NATIVE

Fluxo de autenticação

Para isso, vamos trabalhar com o Firebase

X Adicionar o Firebase ao seu app da Web

1 Registrar app

Apelido do app ②

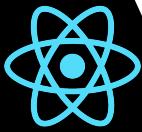
MyHabitTimeline

Também configure o **Firebase Hosting** para este app. [Saiba mais](#) ↗

A configuração do Hosting também pode ser feita depois. Comece a usar a qualquer momento sem pagar nada.

Registrar app

2 Adicionar SDK do Firebase



REACT NATIVE

Fluxo de autenticação

Para isso, vamos trabalhar com o Firebase

Adicionar o Firebase ao seu app da Web

Registrar app

Adicionar SDK do Firebase

Copie e cole esses scripts na parte inferior da tag <body>, mas antes de usar qualquer serviço do Firebase:

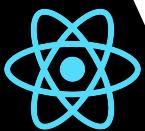
```
<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/6.0.4.firebaseio-app.js"></script>

<!-- TODO: Add SDKs for Firebase products that you want to use
      https://firebase.google.com/docs/web/setup#config-web-app -->

<script>
  // Your web app's Firebase configuration
  var firebaseConfig = {
    apiKey: "AIzaSyCNitxD7TRFrZvgU4v8YQWBhWYFsX28Aoc",
    authDomain: "myhabittimeelinemadalozzo.firebaseioapp.com",
    databaseURL: "https://myhabittimeelinemadalozzo.firebaseioio.com",
    projectId: "myhabittimeelinemadalozzo",
    storageBucket: "myhabittimeelinemadalozzo.appspot.com",
    messagingSenderId: "675793945197",
    appId: "1:675793945197:web:74e72611188ff215"
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
</script>
```

Saiba mais sobre o Firebase para Web: [Primeiros passos](#), [Referência da API Web SDK](#), [Amostras](#)

Continuar no console



REACT NATIVE

Fluxo de autenticação

Para isso, vamos trabalhar com o Firebase

```
// Your web app's Firebase configuration
var firebaseConfig = {
  apiKey: "AIzaSyCNitxD7TRFrZvgU4v8YQWBhWYFsX28Aoc",
  authDomain: "myhabitatemelinemadalozzo.firebaseio.com",
  databaseURL: "https://myhabitatemelinemadalozzo.firebaseio.com",
  projectId: "myhabitatemelinemadalozzo",
  storageBucket: "myhabitatemelinemadalozzo.appspot.com",
  messagingSenderId: "675793945197",
  appId: "1:675793945197:web:74e72611188ff215"
};
// Initialize Firebase
firebase.initializeApp(firebaseConfig);
```

× Adicionar o Firebase ao seu app da Web

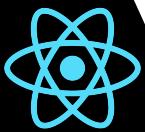
Registrar app

:serviço do Firebase:

:ed first -->
op.js"></script>

1",

[Amostras](#)



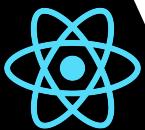
REACT NATIVE

Fluxo de autenticação

Para isso, vamos trabalhar com o Firebase

A configuração de acesso ao firebase deve ser adicionada ao método `componentDidMount()`

```
16     componentDidMount() {
17         var firebaseConfig = {
18             apiKey: "AIzaSyCNitxD7TRFrZvgU4v8YQWBhWYFsX28Aoc",
19             authDomain: "myhabittimelinemadalozzo.firebaseio.com",
20             databaseURL: "https://myhabittimelinemadalozzo.firebaseio.com",
21             projectId: "myhabittimelinemadalozzo",
22             storageBucket: "myhabittimelinemadalozzo.appspot.com",
23             messagingSenderId: "675793945197",
24             appId: "1:675793945197:web:74e72611188ff215"
25         };
26
27         firebase.initializeApp(firebaseConfig);
28     }
```



REACT NATIVE

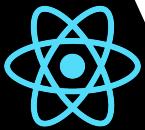
Fluxo de autenticação

Para isso, vamos trabalhar com o Firebase

A configuração de acesso ao firebase deve ser adicionada ao método `componentDidMount()`

Para que o firebase funcione, devemos adicioná-lo ao projeto

```
→ MyHabitTimeline git:(master) ✘ yarn add firebase
```



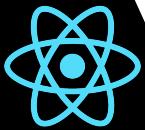
REACT NATIVE

Fluxo de autenticação

Programando a função de acesso

JS LoginPage.js ×

```
35
36     login() {
37         ...
38         this.setState({ isLoading: true, message: '' });
39         const { email, senha } = this.state;
40
41         return firebase
42             .auth()
43             .signInWithEmailAndPassword(email, senha)
44             .then(user => {
45                 console.log("Acesso permitido");
46             })
47             .catch(error => {
48                 this.setState({
49                     message: "Ocorreram erros ao tentar acessar",
50                     isLoading: false
51                 });
52             })
53     }
54 }
```



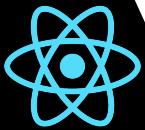
REACT NATIVE

Fluxo de autenticação

Programando a função de acesso

Configuramos os states e
desestruturamos email e senha

```
JS LoginPage.js ✘
35
36     login() {
37         ...
38         this.setState({ isLoading: true, message: '' });
39         const { email, senha } = this.state;
40
41         return firebase
42             .auth()
43             .signInWithEmailAndPassword(email, senha)
44             .then(user => {
45                 console.log("Acesso permitido");
46             })
47             .catch(error => {
48                 this.setState({
49                     message: "Ocorreram erros ao tentar acessar",
50                     isLoading: false
51                 });
52             })
53     }
54 }
```



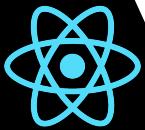
REACT NATIVE

Fluxo de autenticação

Programando a função de acesso

Retornamos a tentativa de conexão com o Firebase

```
JS LoginPage.js ✘
35
36     login() {
37         ...
38         this.setState({ isLoading: true, message: '' });
39         const { email, senha } = this.state;
40
41         return firebase
42             .auth()
43             .signInWithEmailAndPassword(email, senha)
44             .then(user => {
45                 console.log("Acesso permitido");
46             })
47             .catch(error => {
48                 this.setState({
49                     message: "Ocorreram erros ao tentar acessar",
50                     isLoading: false
51                 });
52             })
53     }
54 }
```



REACT NATIVE

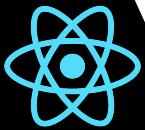
Fluxo de autenticação

Programando a função de acesso

Retornamos a tentativa de conexão com o Firebase

Método de autenticação com Email e senha

```
JS LoginPage.js ×
35
36     login() {
37         ...
38         this.setState({ isLoading: true, message: '' });
39         const { email, senha } = this.state;
40
41         return firebase
42             .auth()
43             .signInWithEmailAndPassword(email, senha)
44             .then(user => {
45                 console.log("Acesso permitido");
46             })
47             .catch(error => {
48                 this.setState({
49                     message: "Ocorreram erros ao tentar acessar",
50                     isLoading: false
51                 });
52             })
53     }
54 }
```



REACT NATIVE

Fluxo de autenticação

Programando a função de acesso

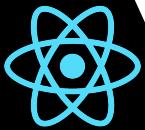
Retornamos a tentativa de conexão com o Firebase

Método de autenticação com Email e senha

Promisse para se caso o acesso foi aprovado

JS LoginPage.js

```
35
36     login() {
37         ...
38         this.setState({ isLoading: true, message: '' });
39         const { email, senha } = this.state;
40
41         return firebase
42             .auth()
43             .signInWithEmailAndPassword(email, senha)
44             .then(user => {
45                 console.log("Acesso permitido");
46             })
47             .catch(error => {
48                 this.setState({
49                     message: "Ocorreram erros ao tentar acessar",
50                     isLoading: false
51                 });
52             })
53     }
54 }
```



REACT NATIVE

Fluxo de autenticação

Programando a função de acesso

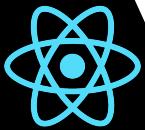
Retornamos a tentativa de conexão com o Firebase

Método de autenticação com Email e senha

Promisse para se caso o acesso foi aprovado

Promisse para se caso o acesso NÃO foi aprovado

```
JS LoginPage.js ×
35
36     login() {
37         ...
38         this.setState({ isLoading: true, message: '' });
39         const { email, senha } = this.state;
40
41         return firebase
42             .auth()
43             .signInWithEmailAndPassword(email, senha)
44             .then(user => {
45                 console.log("Acesso permitido");
46             })
47             .catch(error => {
48                 this.setState({
49                     message: "Ocorreram erros ao tentar acessar",
50                     isLoading: false
51                 });
52             })
53     }
54 }
```

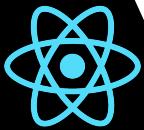


REACT NATIVE

Fluxo de autenticação

Programando a função para controle de acesso

```
JS LoginPage.js ✘
36
37     acessarApp() {
38         this.setState({ isLoading: false });
39         this.props.navigation.replace('Habitos');
40     }
41
42     login() {
43         ...
44         this.setState({ isLoading: true, message: '' });
45         const { email, senha } = this.state;
46
47         return firebase
48             .auth()
49             .signInWithEmailAndPassword(email, senha)
50             .then(user => {
51                 this.acessarApp();
52             })
53             .catch(error => {
54                 this.setState({
55                     message: "Ocorreram erros ao tentar acessar",
56                     isLoading: false
57                 });
58             })
59     }
60 }
```



REACT NATIVE

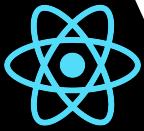
Fluxo de autenticação

Programando a função para controle de acesso

Programamos o acessarApp() e executarmos caso o login tenha sido aprovado

```
JS LoginPage.js x
37     acessarApp() {
38         this.setState({ isLoading: false });
39         this.props.navigation.replace('Habitos');
40     }
41
42     login() {
43         this.setState({ isLoading: true, message: '' });
44         const { email, senha } = this.state;
45
46         return firebase
47             .auth()
48             .signInWithEmailAndPassword(email, senha)
49             .then(user => {
50                 this.acessarApp();
51             })
52             .catch(error => {
53                 this.setState({
54                     message: "Ocorreram erros ao tentar acessar",
55                     isLoading: false
56                 });
57             })
58     }

```



REACT NATIVE

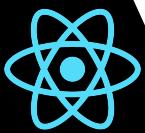
Fluxo de autenticação

Programando a função para controle de acesso

Para navegar usamos o replace para carregar página configurada na pilha da aplicação

```
JS LoginPage.js x
37     acessarApp() {
38         this.setState({ isLoading: false });
39         this.props.navigation.replace('Habitos');
40     }
41
42     login() {
43         this.setState({ isLoading: true, message: '' });
44         const { email, senha } = this.state;
45
46         return firebase
47             .auth()
48             .signInWithEmailAndPassword(email, senha)
49             .then(user => {
50                 this.acessarApp();
51             })
52             .catch(error => {
53                 this.setState({
54                     message: "Ocorreram erros ao tentar acessar",
55                     isLoading: false
56                 });
57             })
58     }

```



REACT NATIVE

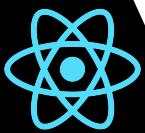
Fluxo de autenticação

Programando a função para controle de acesso

Para navegar usamos o replace para carregar página configurada na pilha da aplicação

```
JS App.js x
1 const AppNavigator = createStackNavigator(
2   {
3     'Login': {
4       screen: LoginPage,
5       navigationOptions: {
6         header: null,
7       }
8     },
9   },
10  'Habitos': {
11    screen: HabitosPage,
12    navigationOptions: {
13      title: 'Hábitos',
14      headerTitleStyle: {
```

```
JS LoginPage.js x
30
31   ...
32   ...
33   ...
34   ...
35   ...
36   ...
37   acessarApp() {
38     this.setState({ isLoading: false });
39     this.props.navigation.replace('Habitos');
40   }
41
42   login() {
43     ...
44     this.setState({ isLoading: true, message: '' });
45     const { email, senha } = this.state;
46
47     return firebase
48       .auth()
49       .signInWithEmailAndPassword(email, senha)
50       .then(user => {
51         this.acessarApp();
52       })
53       .catch(error => {
54         this.setState({
55           message: "Ocorreram erros ao tentar acessar",
56           isLoading: false
57         });
58     })
59   }
60 }
```

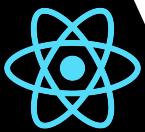


REACT NATIVE

Fluxo de autenticação

Tratando os erros de login

<https://firebase.google.com/docs/reference/js/firebase.auth.Error>



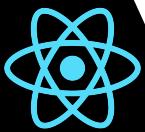
REACT NATIVE

Fluxo de autenticação

Tratando os erros de login

JS LoginPage.js ✘

```
42     getMsgByErrorCode(errorCode) {  
43         switch (errorCode){  
44             case "auth/wrong-password":  
45                 return "Senha incorreta!";  
46             case "auth/invalid-email":  
47                 return "E-mail inválido!";  
48             case "auth/user-not-found":  
49                 return "Usuário não encontrado!";  
50             case "auth/user-disabled":  
51                 return "Usuário desativado!";  
52             case "auth/email-already-in-use":  
53                 return "Usuário já está em uso!";  
54             case "auth/operation-not-allowed":  
55                 return "Operação não permitida!";  
56             case "auth/weak-password":  
57                 return "Senha muito fraca!";  
58             default:  
59                 return "Erro desconhecido!";  
60         }  
61     }
```



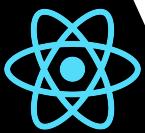
REACT NATIVE

Fluxo de autenticação

Tratando os erros de login

```
JS LoginPage.js ×

62
63     login() {
64         this.setState({ isLoading: true, message: '' });
65         const { email, senha } = this.state;
66
67         return firebase
68             .auth()
69             .signInWithEmailAndPassword(email, senha)
70             .then(user => {
71                 this.acessarApp();
72             })
73             .catch(error => {
74                 this.setState({
75                     message: this.getMsgByErrorCode(error.code),
76                     isLoading: false
77                 });
78             })
79     }
}
```



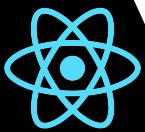
REACT NATIVE

Fluxo de autenticação

Agora, vamos criar um método para renderizar os botões (entrar e cadastrar) ou um ícone de loading

```
JS LoginPage.js ×  
86     renderButton() {  
87         if (this.state.isLoading)  
88             return <ActivityIndicator size="large" style={ styles.loading } />;  
89  
90         return (  
91             <View>  
92  
93                 <View style={ styles.btn }>  
94                     <Button  
95                         title='ENTRAR'  
96                         color='#6542f4'  
97                         onPress={()=> this.login() }  
98                     />  
99                 </View>  
100  
101                 <View style={ styles.btn }>  
102                     <Button  
103                         title='CADASTRE-SE'  
104                         color='#a08af7'  
105                         onPress={()=> this.solicitaCadastro() }  
106                     />  
107                 </View>  
108             </View>  
109         )  
110     }
```

Para o loading vamos usar o componente `ActivityIndicator`



REACT NATIVE

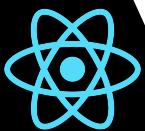
Fluxo de autenticação

Agora, vamos criar um método para renderizar os botões (entrar e cadastrar) ou um ícone de loading

```
JS LoginPage.js ×  
86     renderButton() {  
87         if (this.state.isLoading)  
88             return <ActivityIndicator size="large" style={ styles.loading } />;  
89  
90         return (  
91             <View>  
92  
93                 <View style={ styles.btn }>  
94                     <Button  
95                         title='ENTRAR'  
96                         color='#6542f4'  
97                         onPress={()=> this.login() }  
98                     />  
99                 </View>  
100  
101                 <View style={ styles.btn }>  
102                     <Button  
103                         title='CADASTRE-SE'  
104                         color='#a08af7'  
105                         onPress={()=> this.solicitaCadastro() }  
106                     />  
107                 </View>  
108             </View>  
109         )  
110     }
```

Para o loading vamos usar o componente `ActivityIndicator`

Caso o state `isLoading` estiver TRUE carrega o ícone de loading, senão apresenta os botões



REACT NATIVE

Fluxo de autenticação

Agora, vamos criar um método para renderizar os botões (entrar e cadastrar) ou um ícone de loading

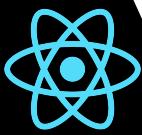
No método render() vamos retirar os dois botões e colocarmos a chamada para o método criado

```
136
137
138
139
140
141
142
143
144
145
```

```
    placeholder="*****"
    secureTextEntry
    value={ this.state.senha }
    onChangeText={ value => this.onChangeHandler('senha', value) }
/>
</FormRow>

{ this.renderButton() }

</ScrollView>
```



REACT NATIVE

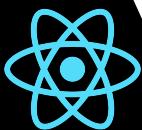
Fluxo de autenticação

Agora, vamos criar o método que renderiza a mensagem de erro em tela

```
JS LoginPage.js ×

111    renderMessage() {
112        const { message } = this.state;
113        if (!message)
114            return null;
115
116        Alert.alert(
117            "Erro!",
118            message.toString(),
119            [
120                {
121                    text: 'OK',
122                    onPress: () => { this.setState({ message: '' }); }
123                }
124            ]
125        );
126    }
```

O alerta só será efetuado se existir mensagem configurada no state



REACT NATIVE

Fluxo de autenticação

Agora, vamos criar o método que renderiza a mensagem de erro em tela

JS LoginPage.js ×

```
111 renderMessage() {  
112     const { message } = this.state;  
113     if (!message)  
114         return null;  
115  
116     Alert.alert(  
117         "Erro!",  
118         message.toString(),  
119         [{  
120             text: 'OK',  
121             onPress: () => { this.setState({ message: '' }); }  
122         }]  
123     );  
124 }  
125 }
```

O alerta só será efetuado se existir mensagem configurada no state

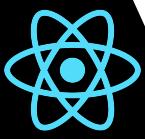
JS LoginPage.js ×

```
1 import React from 'react';  
2 import { TextInput, StyleSheet, View, ScrollView,  
3         Button, Image, KeyboardAvoidingView,  
4         ActivityIndicator, Alert } from 'react-native';  
5 import FormRow from '../components/FormRow';  
6 import firebase from 'firebase';
```



JS LoginPage.js ×

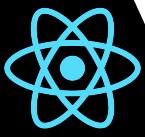
```
8  export default class LoginPage extends React.Component {  
9      constructor(props) {  
10         super(props);  
11  
12         this.state = {  
13             email: '',  
14             senha: '',  
15             isLoading: false,  
16             message: "",  
17         };  
18     }  
19 }
```



REACT NATIVE

Fluxo de autenticação

Vamos programar o cadastro de um novo usuário

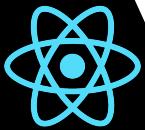


REACT NATIVE

Fluxo de autenticação

Criamos o método cadastrar()

```
JS LoginPage.js ×
81
82     cadastrar() {
83
84     }
85
86     solicitaCadastro() {
87         const { email, senha } = this.state;
88         if (!email || !senha) {
89             Alert.alert(
90                 "Cadastramento!",
91                 "Para se cadastrar informe e-mail e senha"
92             );
93             return null;
94         }
95         Alert.alert(
96             "Cadastramento!",
97             "Deseja cadastrar seu usuário com os dados informados?",
98             [
99                 {
100                     text: "CANCELAR",
101                     style: 'cancel' // apenas para estilizar botão do IOS
102                 },
103                 {
104                     text: "CADASTRAR",
105                     onPress: () => { this.cadastrar() }
106                 },
107             ],
108         );
109     }
110 }
```



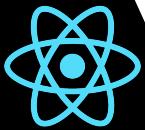
REACT NATIVE

Fluxo de autenticação

Criamos o método cadastrar()

Alteramos o solicitaCadastro()

```
JS LoginPage.js ×
81
82     cadastrar() {
83
84 }
85
86     solicitaCadastro() {
87         const { email, senha } = this.state;
88         if (!email || !senha) {
89             Alert.alert(
90                 "Cadastramento!",
91                 "Para se cadastrar informe e-mail e senha"
92             );
93             return null;
94         }
95         Alert.alert(
96             "Cadastramento!",
97             "Deseja cadastrar seu usuário com os dados informados?",
98             [
99                 {
100                     text: "CANCELAR",
101                     style: 'cancel' // apenas para estilizar botão do IOS
102                 },
103                 {
104                     text: "CADASTRAR",
105                     onPress: () => { this.cadastrar() }
106                 },
107             ],
108         );
109     }
110 }
```



REACT NATIVE

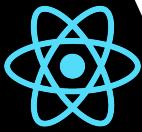
Fluxo de autenticação

Criamos o método cadastrar()

Alteramos o solicitaCadastro()

Se o e-mail e a senha não
forem informados na tela inicial ←
então, dispara alerta solicitando

```
JS LoginPage.js ×
81
82     cadastrar() {
83
84 }
85
86     solicitaCadastro() {
87         const { email, senha } = this.state;
88         if (!email || !senha) {
89             Alert.alert(
90                 "Cadastramento!",
91                 "Para se cadastrar informe e-mail e senha"
92             );
93             return null;
94         }
95         Alert.alert(
96             "Cadastramento!",
97             "Deseja cadastrar seu usuário com os dados informados?",
98             [
99                 {
100                     text: "CANCELAR",
101                     style: 'cancel' // apenas para estilizar botão do IOS
102                 },
103                 {
104                     text: "CADASTRAR",
105                     onPress: () => { this.cadastrar() }
106                 },
107             ],
108         );
109     }
110 }
```



REACT NATIVE

Fluxo de autenticação

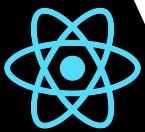
Criamos o método cadastrar()

Alteramos o solicitaCadastro()

Se o e-mail e a senha não
forem informados na tela inicial ←
então, dispara alerta solicitando

Solicita confirmação
de cadastro

```
JS LoginPage.js ×
82     cadastrar() {
83
84   }
85
86   solicitaCadastro() {
87     const { email, senha } = this.state;
88     if (!email || !senha) {
89       Alert.alert(
90         "Cadastramento!",
91         "Para se cadastrar informe e-mail e senha"
92       );
93       return null;
94     }
95     Alert.alert(
96       "Cadastramento!",
97       "Deseja cadastrar seu usuário com os dados informados?",
98       [
99         {
100           text: "CANCELAR",
101           style: 'cancel' // apenas para estilizar botão do IOS
102         },
103         {
104           text: "CADASTRAR",
105           onPress: () => { this.cadastrar() }
106         },
107       ],
108     );
109   }
110 }
```



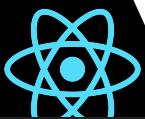
REACT NATIVE

Fluxo de autenticação

No método cadastrar() vamos solicitar o cadastro de novo usuário no firebase utilizando a função createUser

```
JS LoginPage.js ×

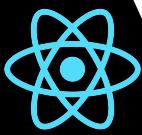
81
82     cadastrar() {
83         ...
84         const { email, senha } = this.state;
85
86         return firebase
87             .auth()
88             .createUserWithEmailAndPassword(email, senha)
89             .then(user => {
90                 this.acessarApp();
91             })
92             .catch(error => {
93                 this.setState({
94                     message: this.getMsgByErrorCode(error.code),
95                     isLoading: false
96                 });
97             })
98     }
99 }
```



REACT NATIVE

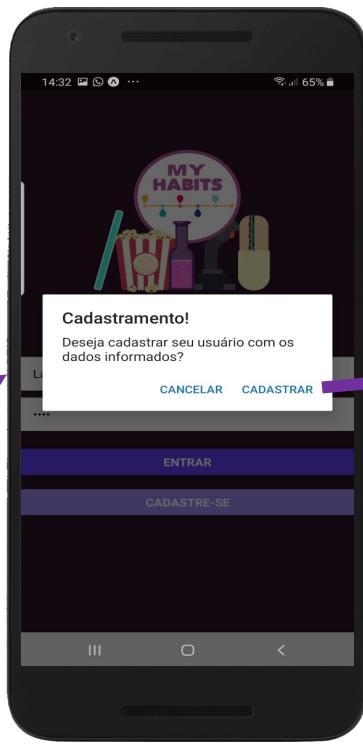
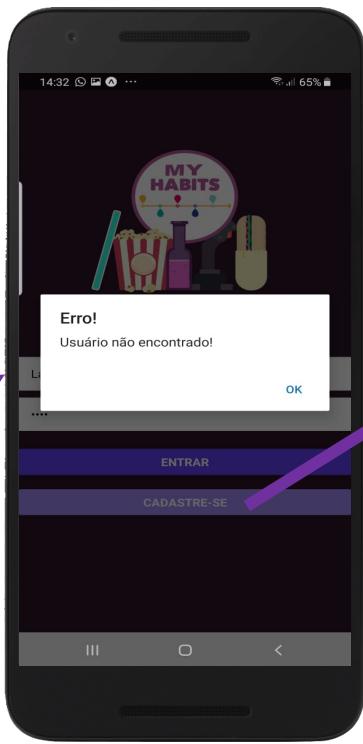
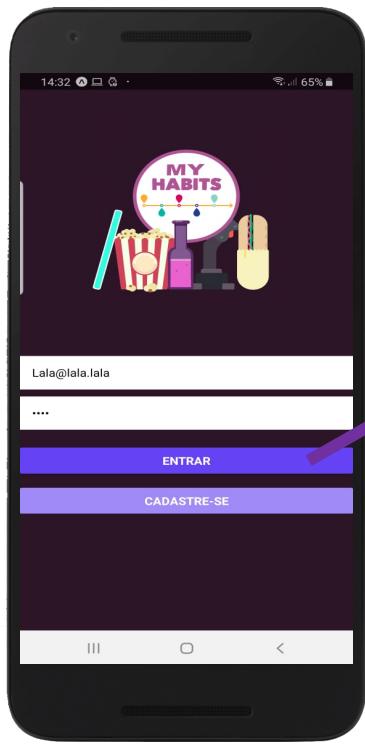
JS LoginPage.js ×

```
189
190
191
192
193
194
195
196
197
      value={ this.state.senha
      onChangeText={ value => t
      />
    </FormRow>
      { this.renderButton() }
      { this.renderMessage() }
    </ScrollView>
      })
    }
```



REACT NATIVE

Fluxo de Autenticação com Firebase



Hábitos
0 Correr
1 Musculação
2 Caminhar
3 Fast Food
4 Junk Food
5 Futebol
6 Seriado
7 Video Game