



# Desenvolvimento para Dispositivos Móveis

## React Native

parte 1

# Aula anterior

## Ambiente

Android Studio (utilizaremos o expo-cli), NodeJS (Version Manager)

Você terá acesso ao comando **npm**, que será o gerenciador de pacotes Node (você também pode utilizar o **Yarn** se preferir).

Para a instalação do expo-cli, utilize o comando abaixo:

```
npm install -g expo-cli
```

```
expo init aula01 --npm
```

```
cd aula01
```

```
expo start      # ou apenas npm start
```

# Aula anterior

npm start

```
> aula01@1.0.0 start
> expo start

Starting project at C:\Users\User\Documents\laravel\aula01
Starting Metro Bundler



> Metro waiting on exp://192.168.0.36:19000
> Scan the QR code above with Expo Go (Android) or the Camera app (iOS)

> Press a | open Android
> Press w | open web

> Press j | open debugger
> Press r | reload app
> Press m | toggle menu

> Press ? | show all commands

Logs for your project will appear below. Press Ctrl+C to exit.
```



## Aula anterior

Interface Web

Emulador Android Studio

Direto no Device



## Para Hoje

Estrutura de arquivos

Navegação básica

Primeiros componentes

Operações lógicas

Teste e execução

## Como funciona a arquitetura

Quando criamos uma aplicação React Native



Antes a comunicação era indireta (json)



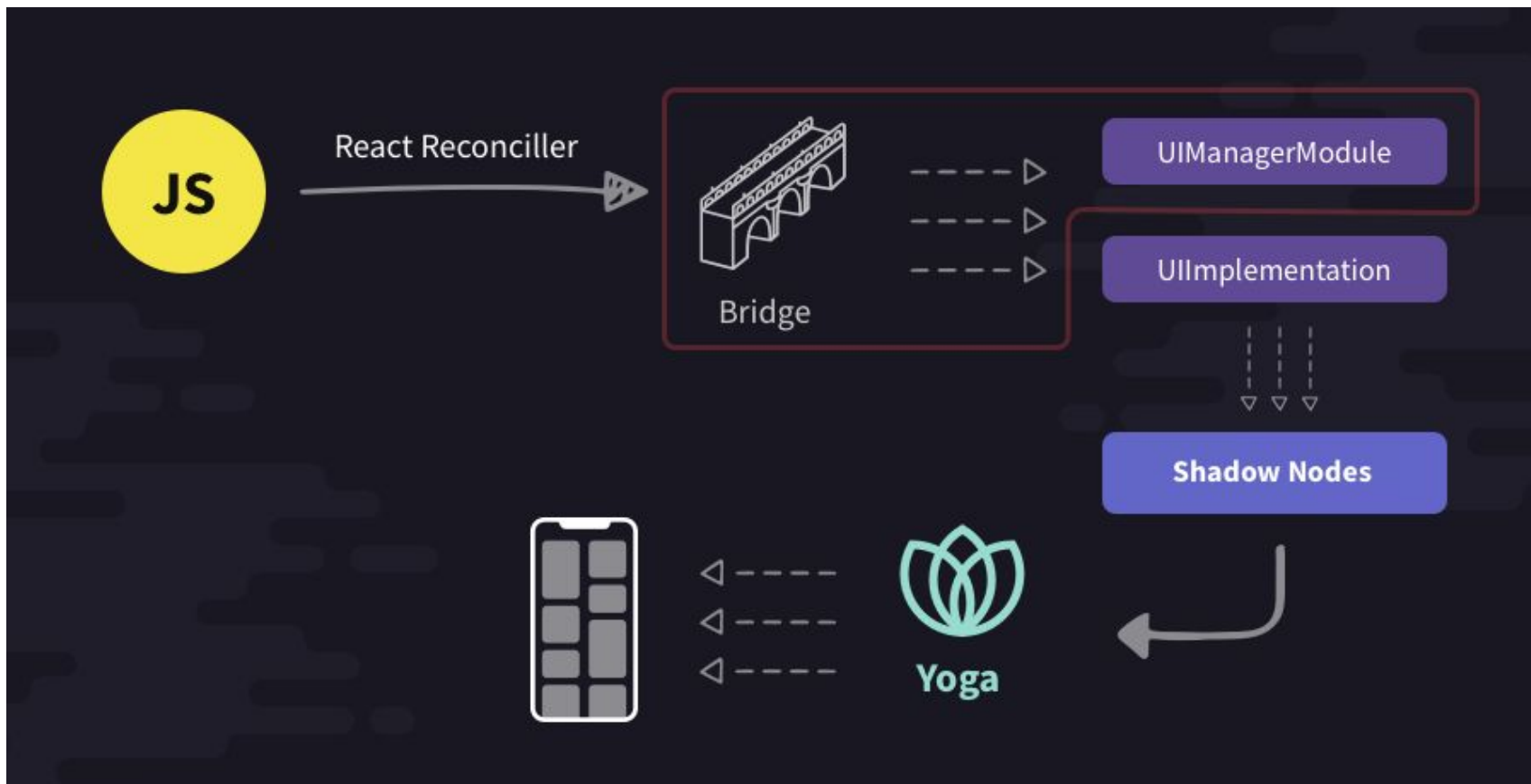
## JavaScript Interface (JSI)

JSI permitiu a comunicação direta de um recurso C++ através de referência

Como na DOM da browser

# Fabric

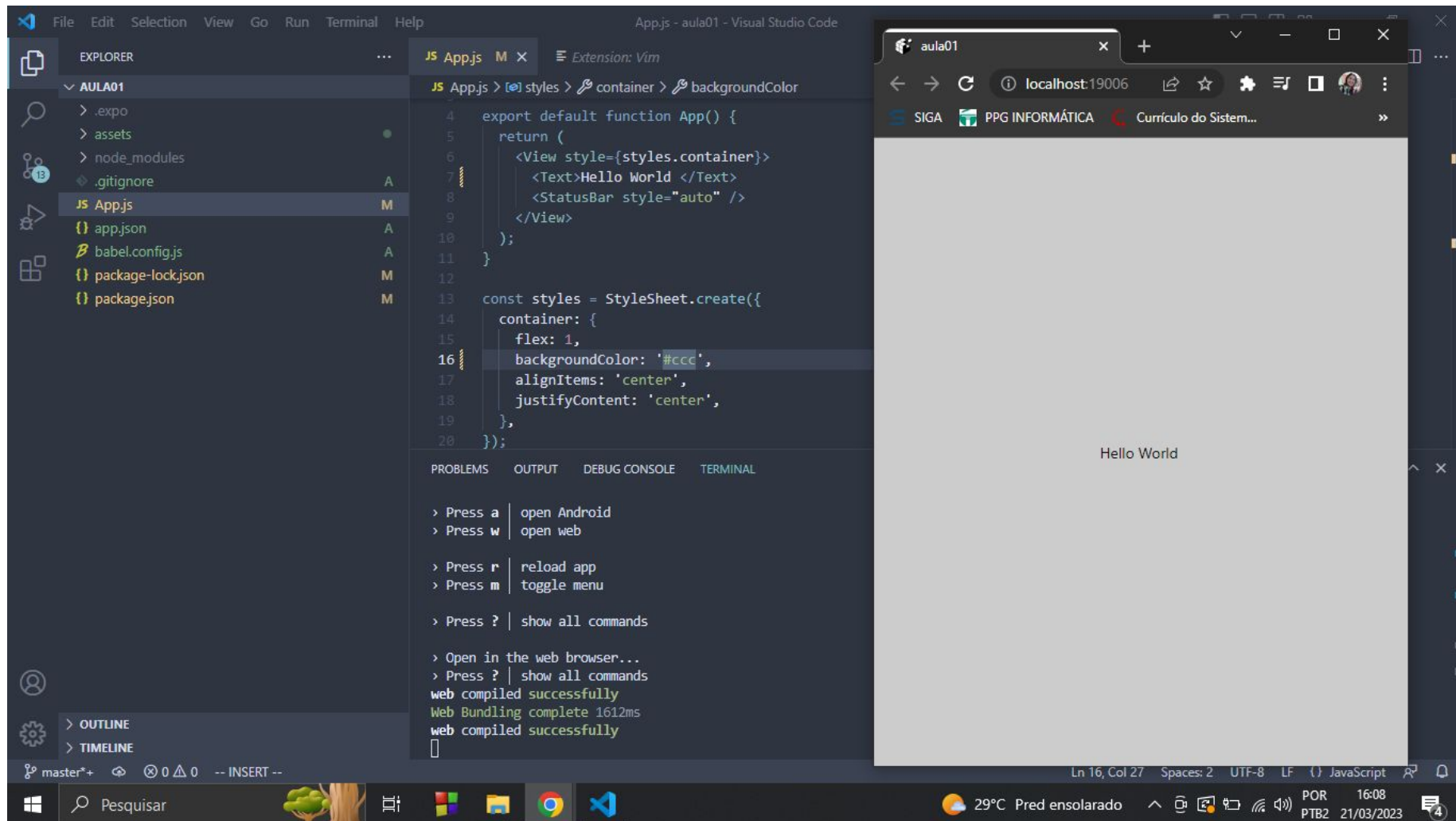
Fabric e JSI (2018): comunicação assíncrona e serial





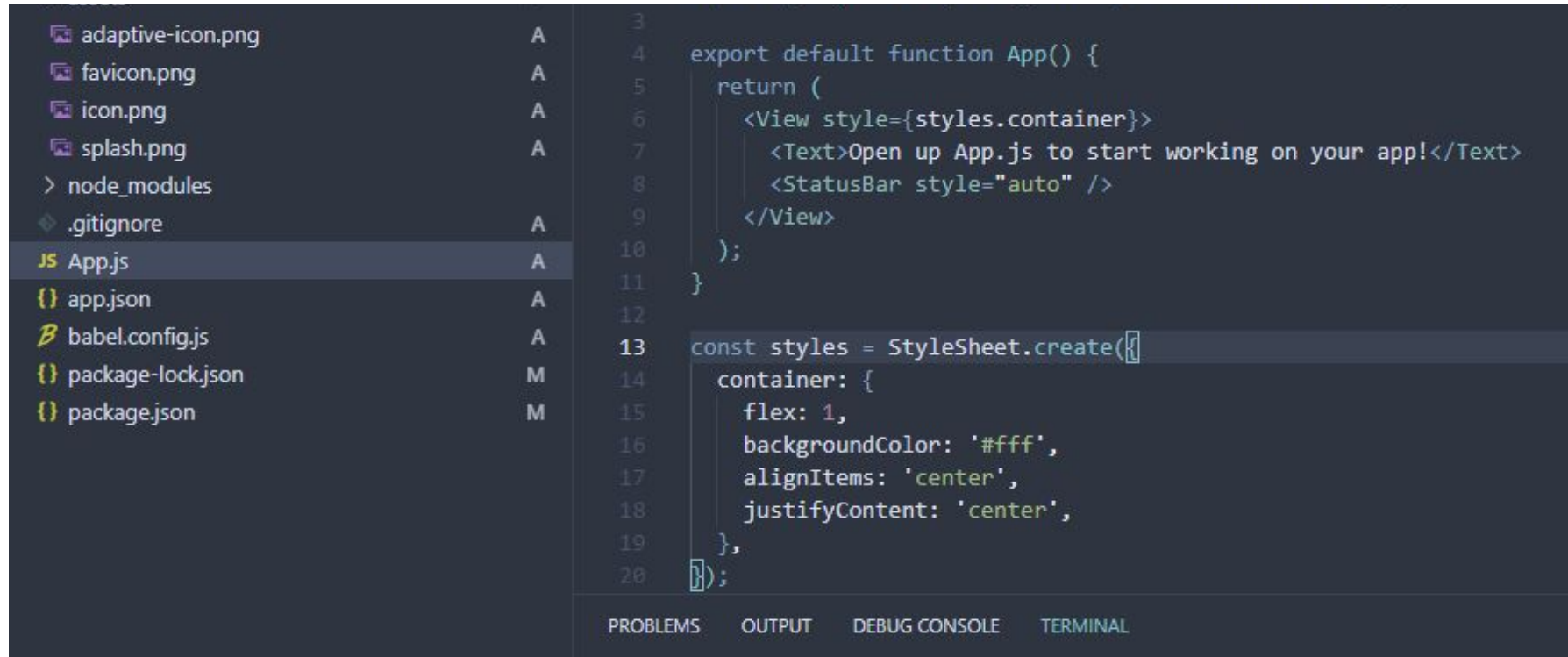
# Ambiente

## interface app inicial



# RN - Estrutura

## Arquivo App.js define os componentes



```
3
4 export default function App() {
5   return (
6     <View style={styles.container}>
7       <Text>Open up App.js to start working on your app!</Text>
8       <StatusBar style="auto" />
9     </View>
10  );
11 }
12
13 const styles = StyleSheet.create({
14   container: {
15     flex: 1,
16     backgroundColor: '#fff',
17     alignItems: 'center',
18     justifyContent: 'center',
19   },
20 });
```



# Componentes

## Componentes Básicos

- View
- Image
- Text
- Text Input
- Stylesheet

## Componentes UI

- Button
- Switch
- Stylesheet



# Componentes

Criando os primeiros componentes.

Crie a pasta `src/components/title` no `./` do app  
`index.js`

```
import React from "react"
```

```
import {View, Text} from "react-native"
```

# Componentes

Criamos a função que vai definir os componentes:

```
export default function Title(){  
  return(  
    <View>  
      <Text>Titulo App</Text>  
    </View>  
  );  
}
```



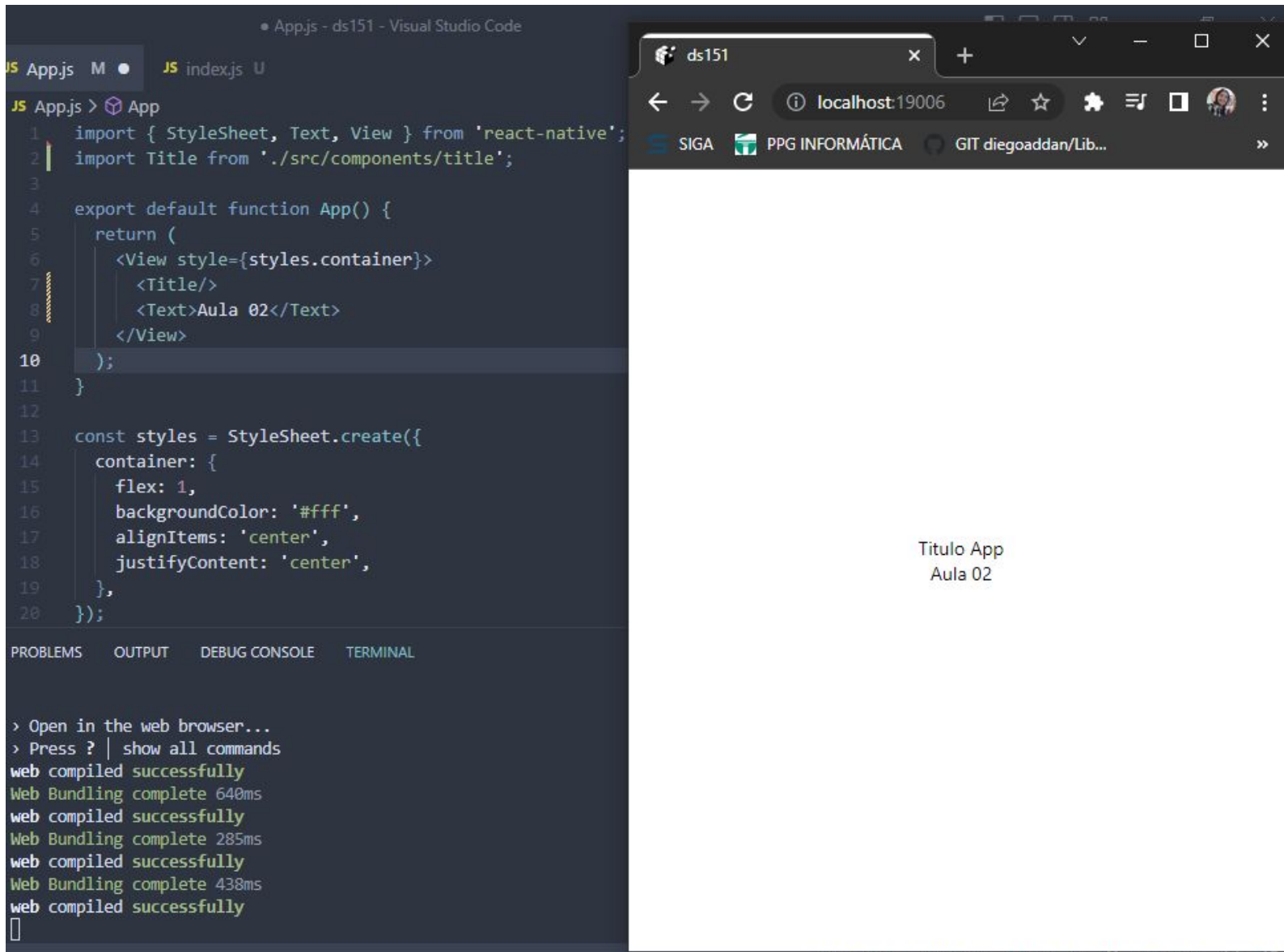
# Componentes

Desta forma facilita manutenção e atualizações

no App.js importamos o componente

```
import Title from './src/components/title';
```

# Componentes



The image shows a development environment with Visual Studio Code on the left and a web browser on the right. The VS Code editor displays the `App.js` file, which imports components from `react-native` and a local `title` component. It defines an `App` function that returns a `View` containing a `Title` component and a `Text` component with the text "Aula 02". Below the code, the terminal shows repeated messages of "web compiled successfully" and "Web Bundling complete". The web browser on the right, titled "ds151", shows the rendered output of the app: "Titulo App" and "Aula 02".

```
App.js > App
1 import { StyleSheet, Text, View } from 'react-native';
2 import Title from './src/components/title';
3
4 export default function App() {
5   return (
6     <View style={styles.container}>
7       <Title/>
8       <Text>Aula 02</Text>
9     </View>
10  );
11 }
12
13 const styles = StyleSheet.create({
14   container: {
15     flex: 1,
16     backgroundColor: '#fff',
17     alignItems: 'center',
18     justifyContent: 'center',
19   },
20 });
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

> Open in the web browser...  
> Press ? | show all commands  
web compiled successfully  
Web Bundling complete 640ms  
web compiled successfully  
Web Bundling complete 285ms  
web compiled successfully  
Web Bundling complete 438ms  
web compiled successfully

ds151  
localhost:19006  
SIGA PPG INFORMÁTICA GIT diegoaddan/Lib...

Titulo App  
Aula 02



## Componentes

Crie a pasta `src/components/main` no `./` do  
app  
`index.js`

Adicione a tag `</form>` na função

Crie a pasta `src/components/form` no `./` do  
app  
`index.js`



# Componentes

## Main/index.js



```
2  import { View } from "react-native"
3
4  export default function Title(){
5    return(
6      <View>
7        <Form/>
8      </View>
9    );
10 }
```

# Componentes

Podemos inserir elementos no formulário através de views

```
export default function Form(){  
  return(  
    //conteudo do formulario  
    <View>  
      <View></View>  
      <View></View>  
      <View></View>  
    </View>  
  );  
}
```

# Componentes

Criaremos no form um **label** e um **input**

```
return(  
  //conteudo do formulario  
  <View>  
    <View>  
      <Text>Altura</Text>  
      <TextInput></TextInput>  
      <Text>Peso</Text>  
      <TextInput></TextInput>  
    </View>  
  </View>  
)
```

# Componentes

Podemos adicionar atributos para as tags como **placeholder** e **tipo de entrada**

```
<Text>Peso</Text>
```

```
<TextInput
```

```
placeholder="Ex. 70.10"
```

```
keyboardType="numeric"
```

```
/>
```



## Componentes

Importamos o **Main** e o **Form** no **App.js**

```
Import Main from './src/components/Main/index'
```

O form é importado indiretamente na Landing Page através da **Main**

```
Import Form from '../Form/'
```

# Componentes

E na função principal chamamos o componente através da função:

```
export default function App() {  
  return (  
    <View style={styles.container}>  
      <Title/>  
      <Main/>  
    </View>  
  );  
}
```

```
5 export default function Main(){  
6   return(  
7     <View>  
8       <Form/>  
9     </View>  
10  );  
11 }
```

# Componentes

## Importamos o Main no App.js

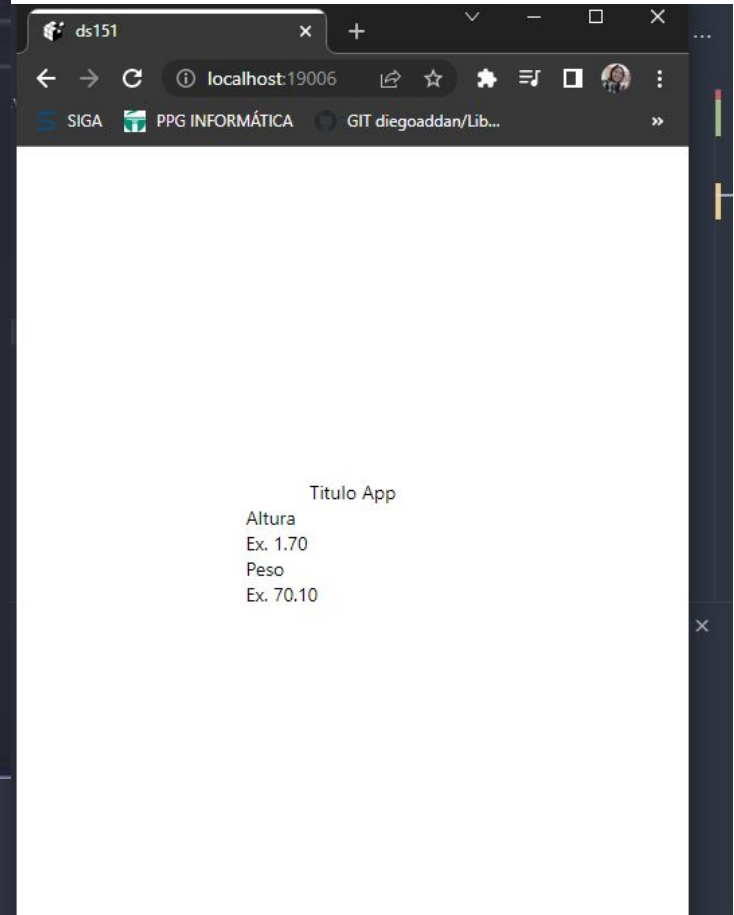
```
import React from "react"
import { View, Text, TextInput } from "react-native"

export default function Form(){
  return(
    <View>
      <View>

        <Text>Altura</Text>
        <TextInput
          placeholder="Ex. 1.75"
          keyboardType="numeric"
        />

        <Text>Peso</Text>
        <TextInput
```

```
web compiled successfully
Web Bundling complete 400ms
web compiled successfully
Web Bundling complete 265ms
web compiled successfully
[]
```



# Componentes

## Incluimos no Form o elemento Button

```
import {Text, TextInput, Button } from "react-native-web";  
  
    <Button title="Calcular IMC"/>
```

Titulo App

Altura

Ex. 1.70

Peso

Ex. 70.10

CALCULAR IMC





## Componentes

Precisamos incluir uma função “onPress”  
que adiciona comportamento ao componente

Além disso criamos um componente para o  
resultado do cálculo

# Componentes

Crie a pasta `src/components/Result/` no `./` do app  
`index.js`

```
export default function ResultImc(props){  
  return(  
    <View>  
      <Text>{props.resultImc}</Text>  
      <Text>{props.messageResultImc}</Text>  
    </View>  
  );  
}
```



## Componentes

No form criaremos os parâmetros que estarão sendo enviados pela função de estado (abaixo do botão)

```
<ResultImc messageResultImc={messageImc}  
  resultImc={imc}></ResultImc>
```

Vamos utilizar o **useState**

```
import React, {useState} from "react"
```

Precisaremos de alguns parametros

# Componentes

## Criamos os parametros e estados iniciais

```
const [height, setHeight] = useState(null)
```

```
const [weight, setWeight] = useState(null)
```

```
const [messageImc, setMessageImc] = useState("preencha os dados")
```

```
const [imc, setImc] = useState(null)
```

```
const [textButton, setTextButton] = useState("Calcular")
```

agora criamos a função de cálculo:  
 $\text{altura} * \text{altura} / \text{peso}$

# Componentes

```
function imcCalculator(){  
    return setImc((weight/(height*height)).toFixed(2))  
}
```

```
function validationImc(){  
    if(weight != null && height != null){  
        imcCalculator()  
        setMessageImc("seu IMC eh igual: ")  
        setTextButton("Calcular Novamente")  
        return  
    }  
    setImc(null)  
    setTextButton("Calcular")  
    setMessageImc("preencha os dados")  
}
```

# Componentes

Adiciona o comportamento no botão com  
onPress e estados nos Inputs

```
<Text>Altura</Text>
<TextInput
  onChangeText={setHeight}
  value={height}
  placeholder="Ex. 1.75"
  keyboardType="numeric"
/>

<Text>Peso</Text>
<TextInput
  onChangeText={setWeight}
  value={weight}
  placeholder="Ex. 75.365"
  keyboardType="numeric"
```

# Componentes

Adiciona o comportamento no botão com  
onPress

```
<Button  
  onPress={() => validationImc()}  
  title="Calcular IMC"  
>
```

Titulo App

Altura  
1.80

Peso  
80

**CALCULAR IMC**

24.69

seu IMC eh igual:



Concluindo...

Estrutura principal do boilerplate

Componentes

Principais recursos

Exercício