INSTITUTO FEDERAL
Ceará

Campus
Crato

Semana de Informática 2020

Introdução ao Desenvolvimento Mobile com React Native

Harley Macêdo de Mello

harley.mello@ifce.edu.br

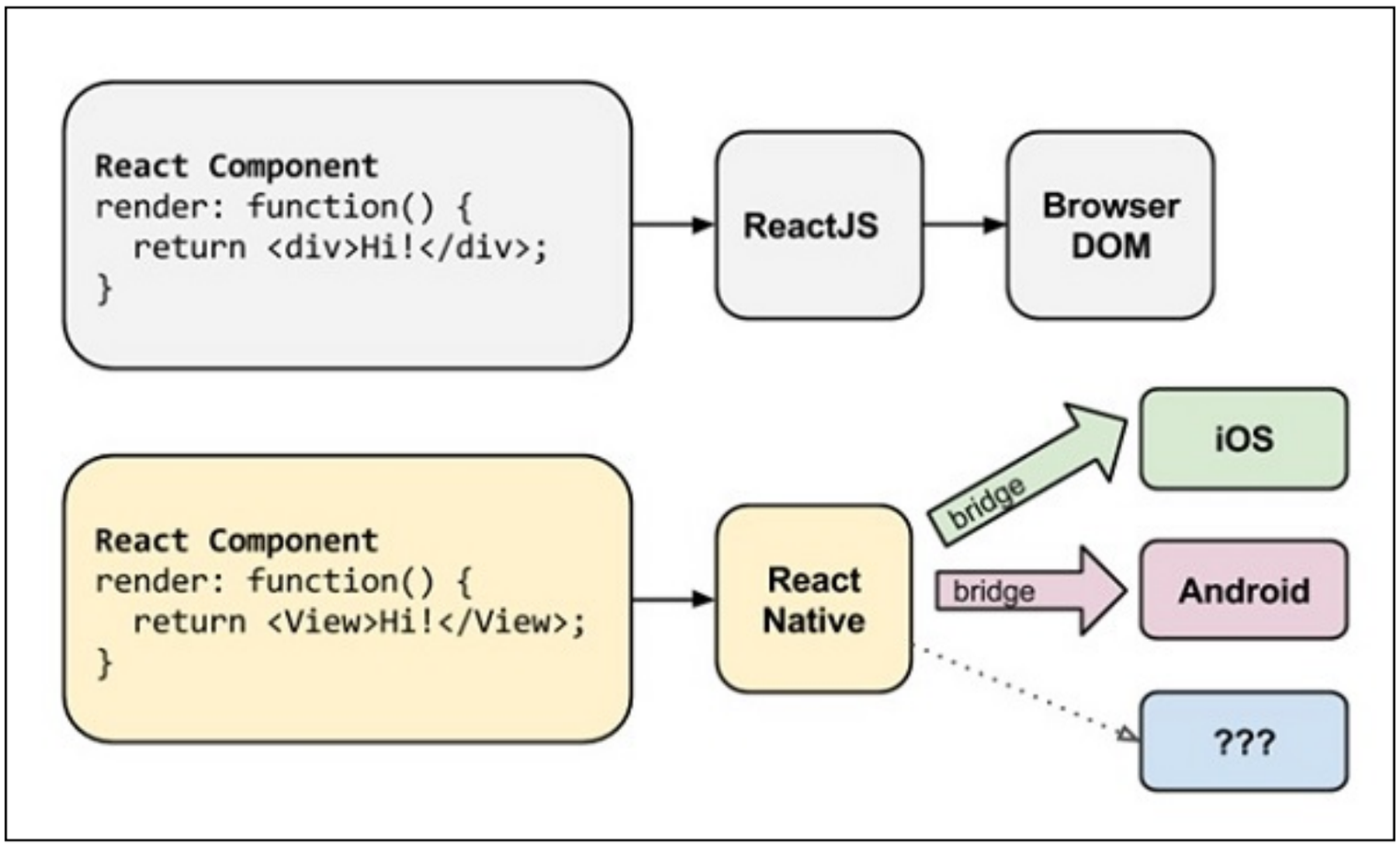
Roteiro

- Conceitos
- Ambiente de desenvolvimento
- Estrutura de App no React Native
- Componentes React Native
- API e JSON
- Props
- State
- Events
- Estilo de componentes
- Condicional e repetição
- Aplicações de exemplo

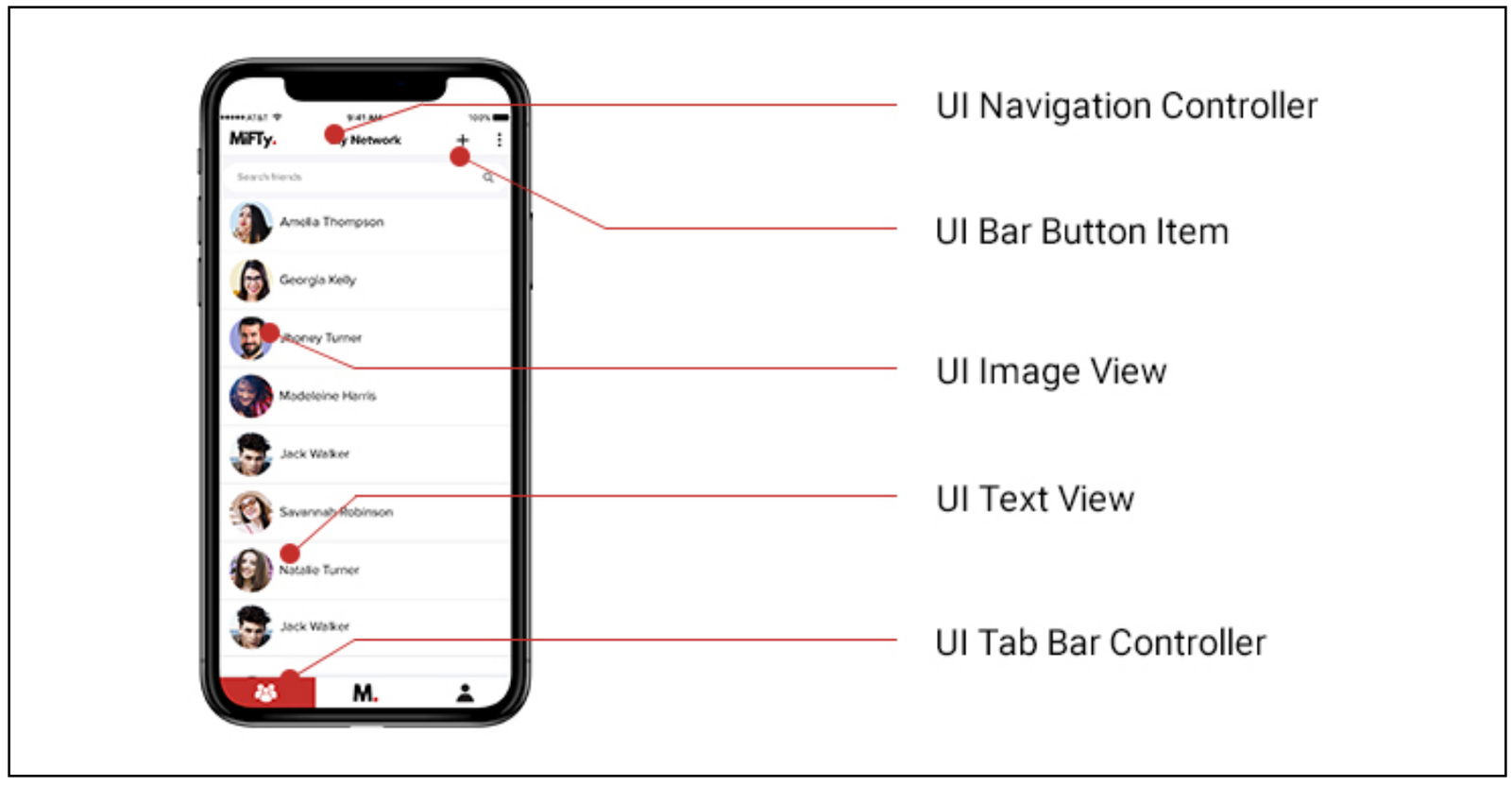
Conceitos

- Construir aplicações para plataformas mobile
- Mantido pelo Facebook
- Usa mesmos conceitos do Reactjs
- Gera uma aplicação nativa, não apenas simula
- Pouca diferença na programação para um S.O. e outro
- Pode agilizar o desenvolvimento

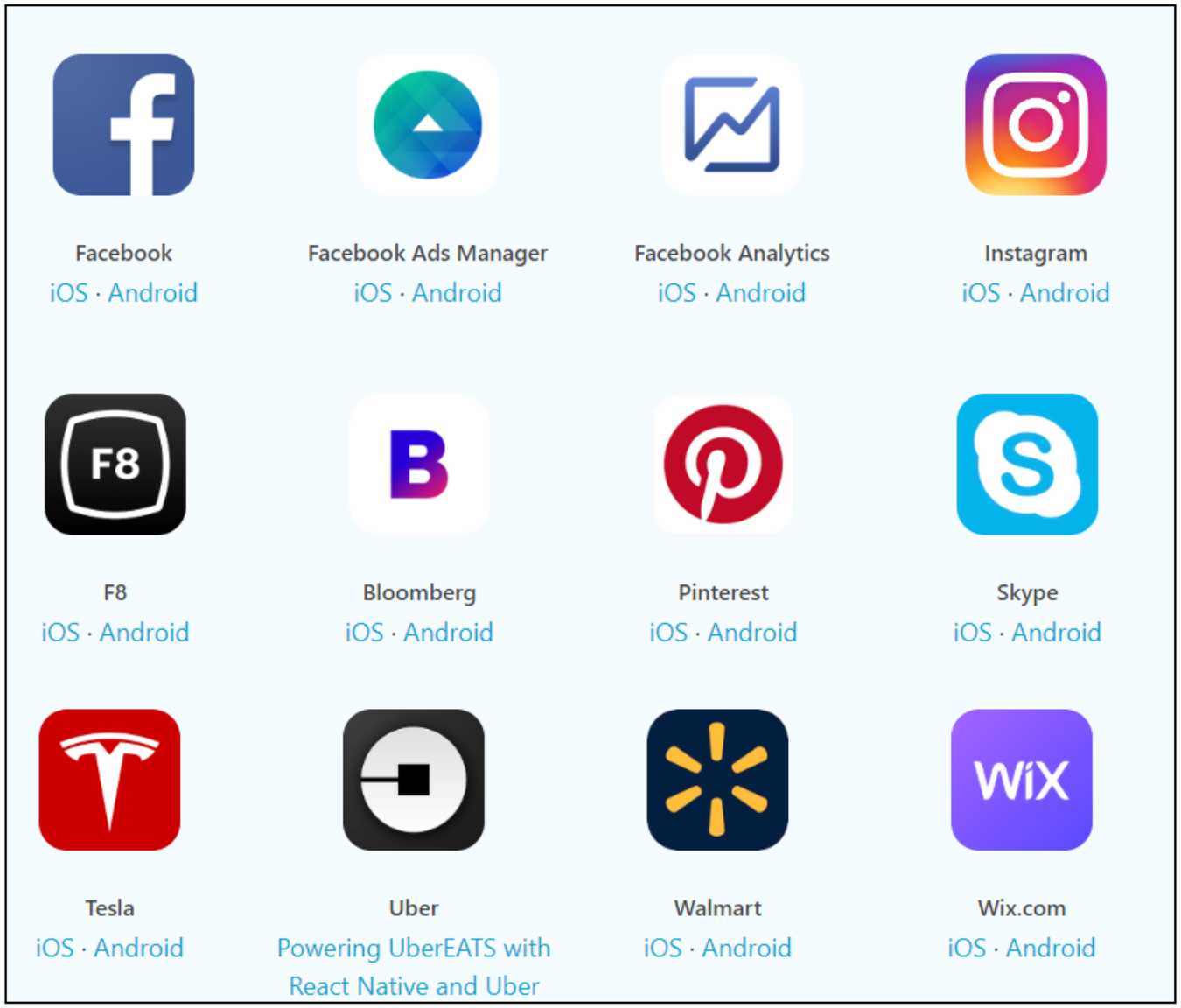
Conceitos


React Component
render: function() {
return Hi! ;
}
ReactJS
Browser DOM
React Component
render: function() {
return <View>Hi!</View>;
}
React Native
bridge
iOS
bridge
Android
???

Conceitos


MiFTy.
Search friends
Amelia Thompson
Georgia Kelly
Rooney Turner
Madeleine Harris
Jack Walker
Savannah Robinson
Natalie Turner
Jack Walker
M.
UI Navigation Controller
UI Bar Button Item
UI Image View
UI Text View
UI Tab Bar Controller

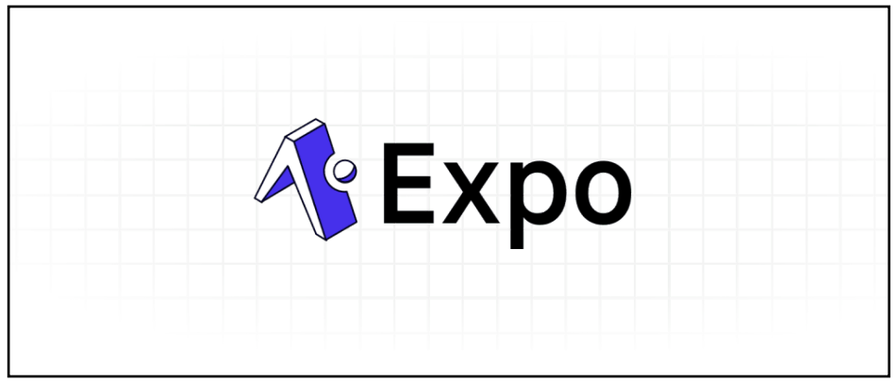
Conceitos

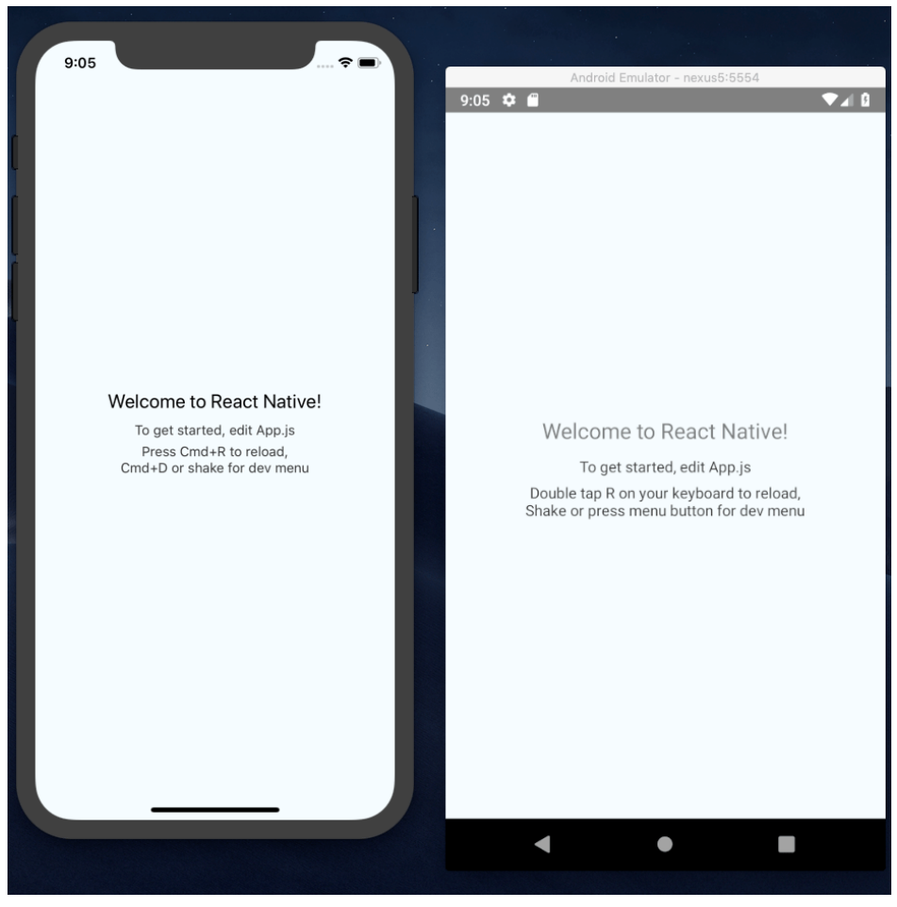

Facebook
iOS · Android
Facebook Ads Manager
iOS · Android
Facebook Analytics
iOS · Android
Instagram
iOS · Android
F8
F8
iOS · Android
Bloomberg
iOS · Android
Pinterest
iOS · Android
Skype
iOS · Android
Tesla
iOS · Android
Uber
Powering UberEATS with React Native and Uber
Walmart
iOS · Android
Wix.com
iOS · Android

Ambiente de desenvolvimento

- Ambiente Android
  - Node e NPM
  - JDK 8 e Android Studio
  - Python 2
  - Simulador de dispositivo Android
- Ambiente Expo
  - Node e NPM
  - Pacote expo-cli
  - Smartphone ou simulador
  - Instalar o app 'expo' na loja de seu sistema móvel
- Totalmente online
  - Acessar snack.expo.io

Ambiente de desenvolvimento

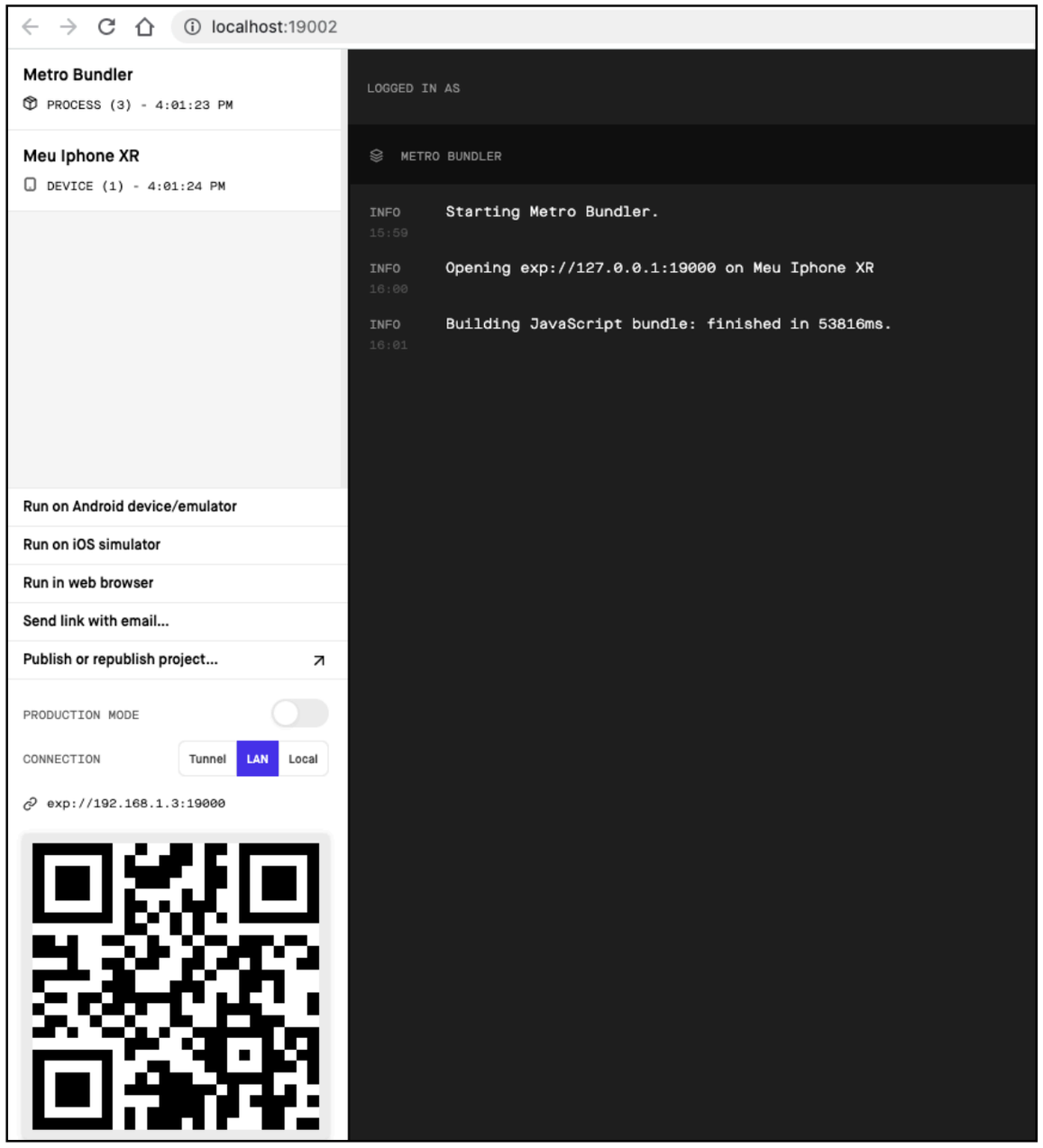


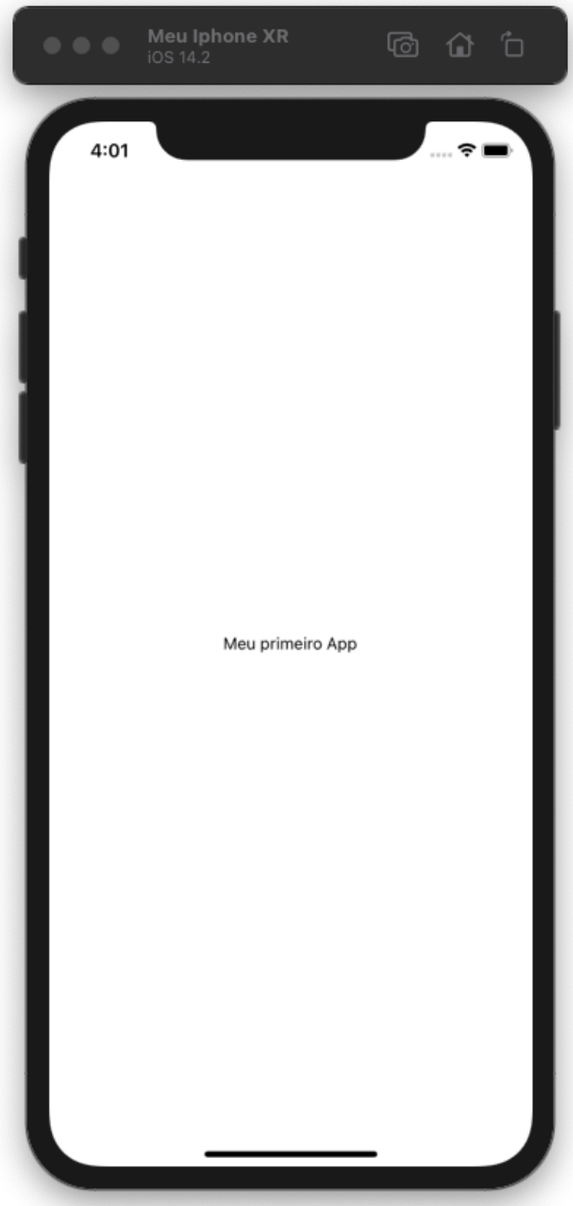


Estrutura de App no React Native

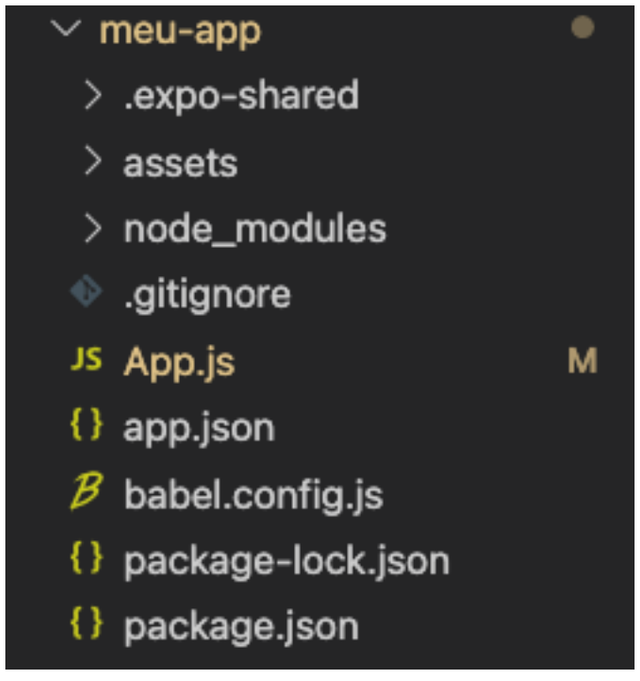
- Instalar o pacote expo-cli
  - npm install -g expo-cli
- Executar o criador de App
  - expo init meu-app
- Entrar na pasta criada
  - cd meu-app
- Iniciar app criado
  - expo start

Estrutura de App no React Native





Estrutura de App no React Native



App.js

meu-app > App.js > ...

```
1  import React from 'react'
2  import { StyleSheet, Text, View } from 'react-native'
3
4  export default function App() {
5    return (
6      <View style={styles.container}>
7        <Text>Meu primeiro App</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 });
```

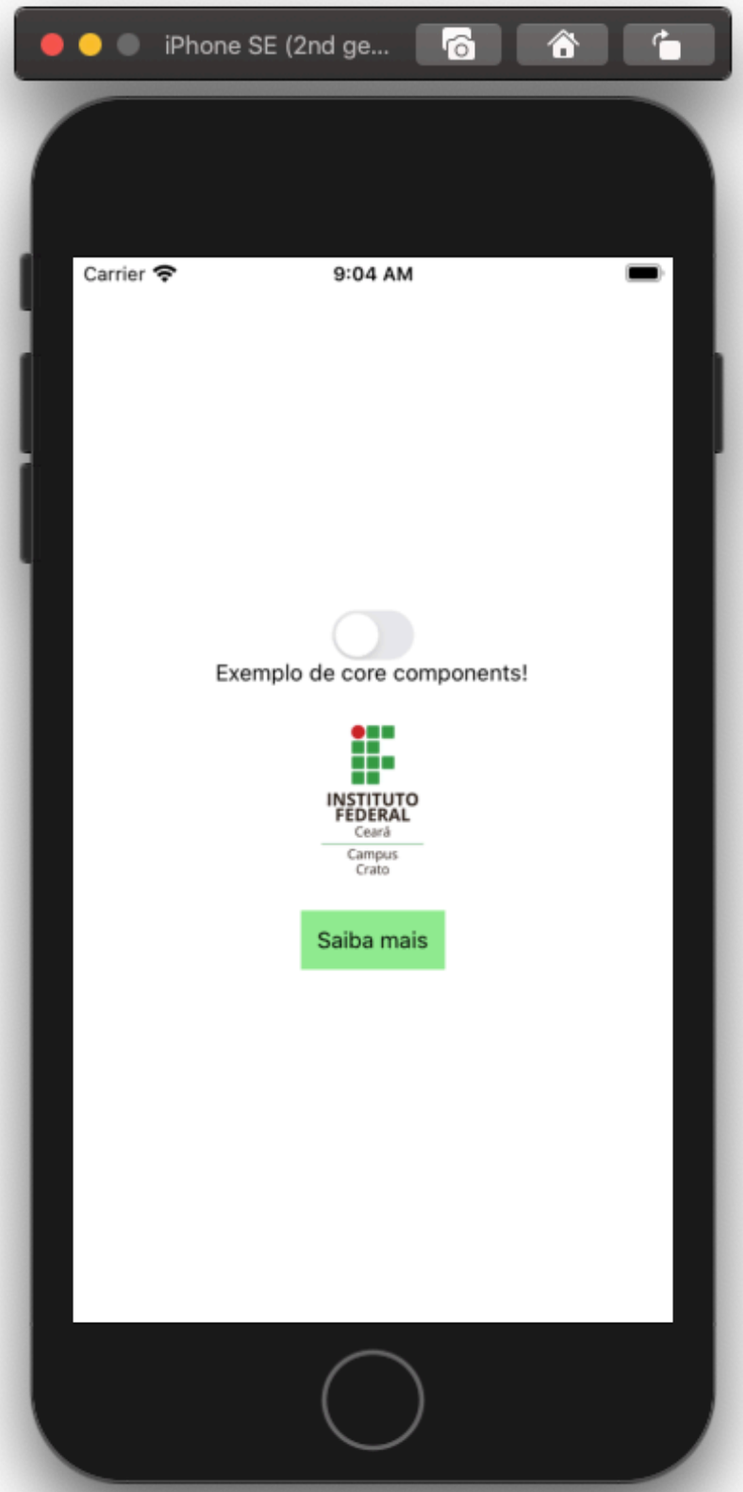
Estrutura de App no React Native

- Arquivo 'app.js' é o componente inicial
- Arquivo 'package.json' contêm configurações da aplicação
- 'imports' contêm as bibliotecas que serão utilizadas no componente
- 'export default function' é a função que permite reuso do componente
- 'styles' contêm os estilos utilizados

Componentes React Native

- São classes ou funções que retornam um conteúdo JSX
- Conteúdo retornado precisa ser unificado
- Podem ter estado e comportamento
- Componentes 'core' como o View e o ListView
- Representam componentes nativos de cada S.O.
- Componentes próprios podem agregar nativos

Componentes React Native



Componentes React Native

- View
  - Container para o conteúdo da tela
  - <View> … </View>
- SafeAreaView
  - Considera apenas área livre
  - <SafeAreaView> … </SafeAreaView

Componentes React Native

- Text
  - Texto genérico
  - Reconhece click
  - <Text> … </Text>
- TextInput
  - Entrada de texto
  - Evento onChangeText
  - <TextInput> … </TextInput>

Componentes React Native

- Button
  - Botão padrão
  - Evento onPress
  - <Button> … </Button>
- TouchableOpacity
  - Container para textos que serão clicados
  - Aparência de botão
  - <TouchableOpacity> … </TouchableOpacity>

Componentes React Native

- Image
  - Imagem padrão
  - Atributo source
  - <Image />
- Switch
  - Alternância ou liga/desliga
  - Evento onValueChange
  - <Switch />

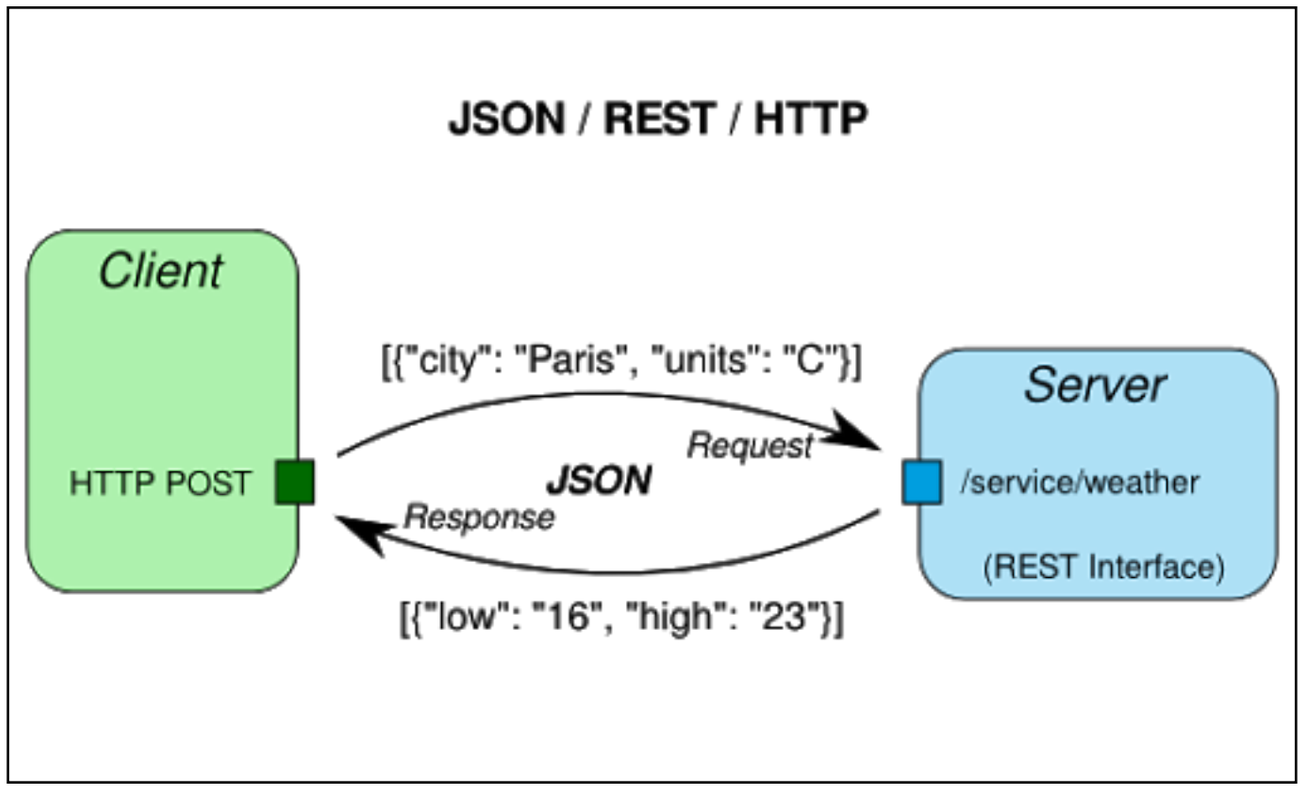
Componentes React Native

- Componentes próprios retornam algum JSX
- Podem agregar outros componentes próprios
- Podem agregar componentes 'core'
- Semelhantes aos componentes próprios no Reactjs
  - <Busca> … </Busca>
  - <Cartao> … </Cartao>

API e JSON

- API
  - É uma forma muito usada para comunicação de sistemas
  - Fornecem informações através de consultas
  - Podem ser grátis ou pagas
  - Podem requerer uma chave de acesso
- JSON
  - Estrutura de arquivo mais utilizado em APIs
  - Formatação da informação com {'chave': 'valor'}

API e JSON


JSON / REST / HTTP
Client
HTTP POST
[{"city": "Paris", "units": "C"}]
Request
JSON
Response
[{"low": "16", "high": "23"}]
Server
/service/weather
(REST Interface)

API e JSON

doc1.json

Users > harleymacedo > Desktop > {} doc1.json > ...

```
{
    "nome": "Harley",
    "sobrenome": "Macedo",
    "cidade": "Crato",
    "habilitado": true,
    "peso": 72
}
```

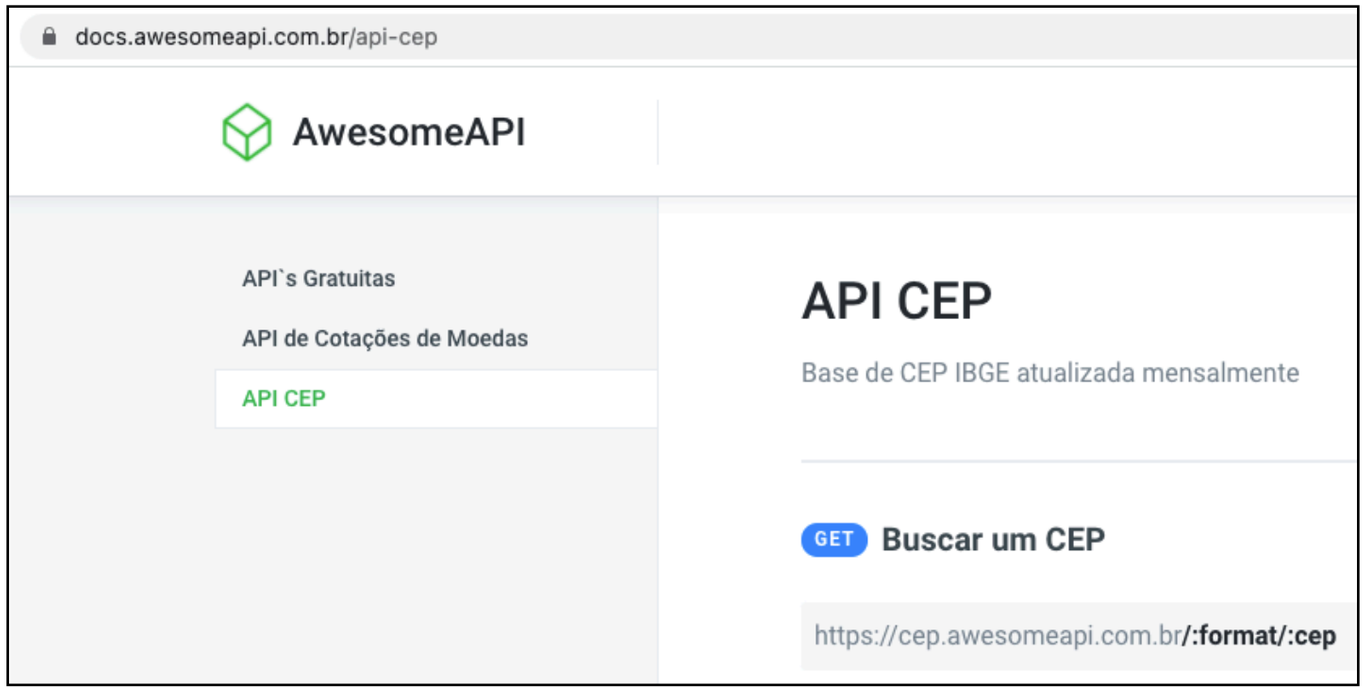
doc1.json

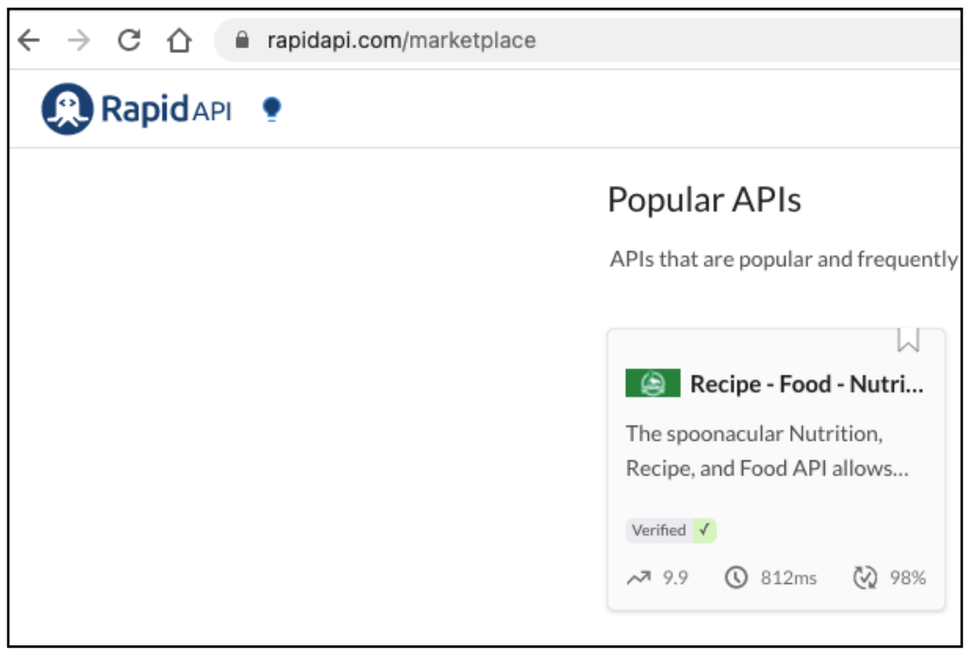
Users > harleymacedo > Desktop > {} doc1.json > ...

```
[
    {
        "nome": "Harley",
        "habilitado": true,
        "peso": 72
    },

    {
        "nome": "Ana",
        "habilitado": true,
        "peso": 62
    }
]
```

API e JSON

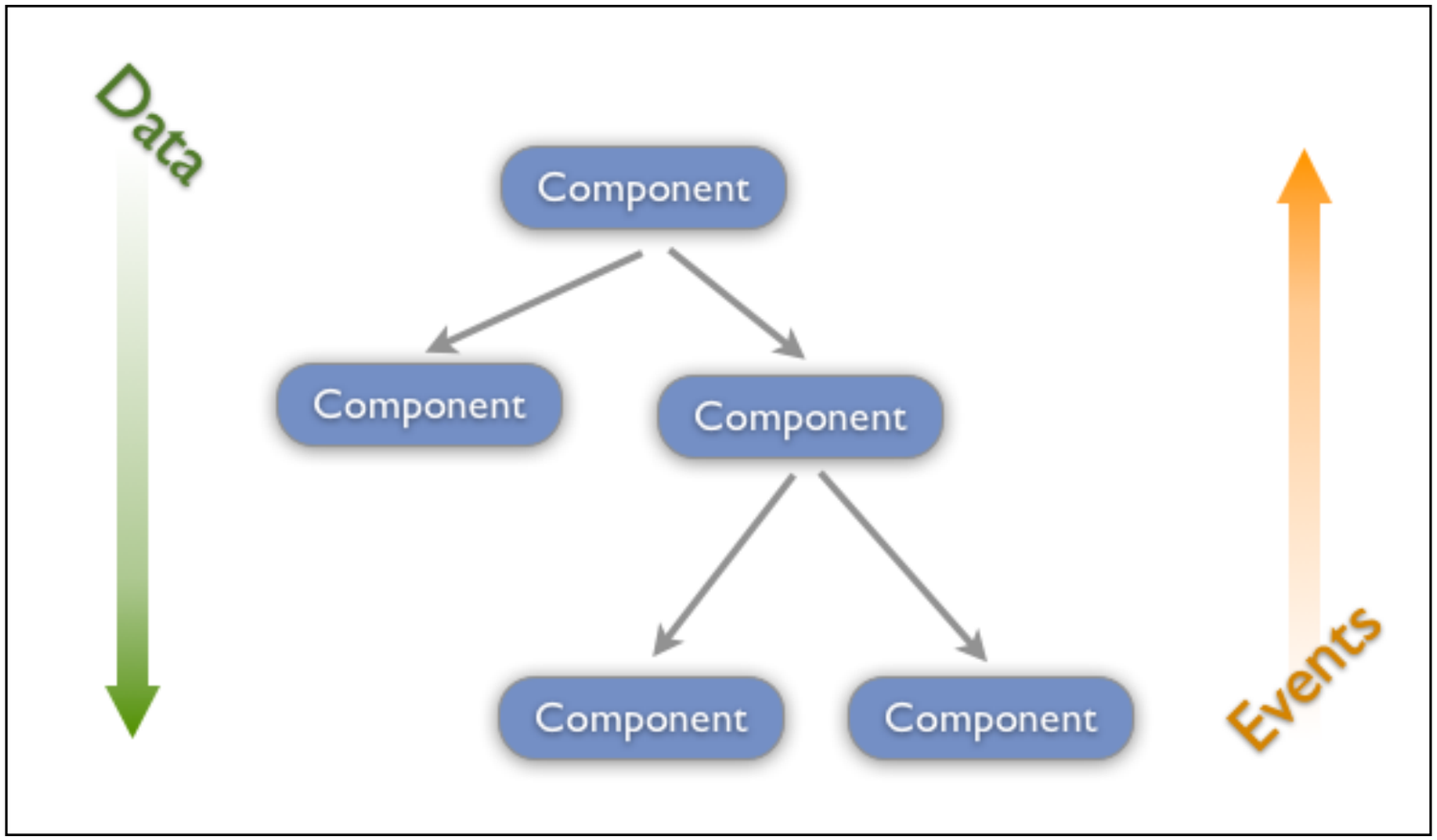

docs.awesomeapi.com.br/api-cep
AwesomeAPI
API`s Gratuitas
API de Cotações de Moedas
API CEP
API CEP
Base de CEP IBGE atualizada mensalmente
GET
Buscar um CEP
https://cep.awesomeapi.com.br/:format/:cep


rapidapi.com/marketplace
RapidAPI
Popular APIs
APIs that are popular and frequently
Recipe - Food - Nutri...
The spoonacular Nutrition, Recipe, and Food API allows...
Verified
9.9
812ms
98%

Props

- Forma usada para passar parâmetros para um componente
- Passados por meio de atributos na própria 'tag'
- As props podem ser integradas ao estado do componente
- As props são apenas leitura

Props


Data
Component
Component
Component
Component
Component
Events

Props

App.js

```
meu-app > App.js > ...
1  import React from 'react'
2  import { StyleSheet, Text, View } from 'react-native'
3  import MeuComponente from './MeuComponente'
4
5  export default function App() {
6    return (
7      <View style={styles.container}>
8        <Text>Meu primeiro App</Text>
9
10       <MeuComponente id='A01' />
11       <MeuComponente id='A02' />
12       <MeuComponente id='A03' />
13     </View>
14   )
15 }
16
```

MeuComponente.js

```
meu-app > MeuComponente.js > ...
1  import React from 'react'
2  import {View, Text} from 'react-native'
3
4  export default function MeuComponente (props) {
5      return (
6          <View>
7              <Text>Meu componente novo: {props.id} </Text>
8          </View>
9      )
10 }
```

State

- Armazenar valores que pertecem ao componente
- Quando o State muda, o componente é renderizado novamente
- Inicializado no construtor
- Usar a função this.setState({ }) para alterar o state
- Uma forma mais indicada é utilizado o 'useState'
- Inicializando atributos e modificando com a função set

State

App.js | MeuComponente.js

meu-app > MeuComponente.js > MeuComponente

```
1   import React, {useState} from 'react'
2   import {View, Text} from 'react-native'
3
4   export default function MeuComponente (props) {
5
6       const [modoNoturno, setModoNoturno] = useState('Desligado')
7
8   return (
9       <View>
10          <Text>Meu componente novo: {props.id} </Text>
11          <Text>O modo noturno está: {modoNoturno} </Text>
12      </View>
13  )
```

State

JS App.js JS MeuComponente.js

meu-app > JS MeuComponente.js > ...

```
1   import React, {useState} from 'react'
2   import {View, Text} from 'react-native'
3
4   export default function MeuComponente (props) {
5
6       const [modoNoturno, setModoNoturno] = useState('Desligado')
7
8       const mudarEstado = () => {
9           setTimeout( () => {setModoNoturno('Ligado')}, 3000 )
10      }
11
12      mudarEstado()
13
14      return (
15          <View>
16              <Text>Meu componente novo: {props.id} </Text>
17              <Text>O modo noturno está: {modoNoturno} </Text>
18          </View>
19      )
20  }
```

Events

- Programar ações de resposta
- Usar o camelCase para nomes de eventos
- Principais eventos
  - onPress
  - onChangeText
  - onValueChange
- O objeto que disparou o evento pode passar parâmetros

Estilo de componentes

- Aplicar o mesmo formato de CSS
- Usando camelCase para as propriedades
- Usar {{ algumAtributo: valor }} para estilos inline
- CSS no mesmo arquivo pode ser definido com constante
- StyleSheet função 'create' para armazenar estilos

Condicional

- Renderizar um JSX com base em condição
- Criar uma função que verifica condição
- Função criada retorna um JSX
- Pode-se usar o operador ternário
  - condicao ? retorno_se_true : retorno_se_false

Condicional

```
4   export default function MeuComponente (props) {
5
6       const [modoNoturno, setModoNoturno] = useState(false)
7
8       const mudarEstado = () => {
9           setTimeout( () => {setModoNoturno('Ligado')}, 3000 )
10      }
11
12      mudarEstado()
13
14      return (
15          <View>
16              <Text>Meu componente novo: {props.id} </Text>
17              <Text>O modo noturno está: {modoNoturno} </Text>
18
19              {modoNoturno ? <Text>O modo noturno economisa bateria</Text> : <Text>Modo noturno não ativo</Text>}
20          </View>
21      )
22  }
```

Repetição

- Renderizar cada item de um array
- A função map pode ser usada para iterar pelos itens
- Em cada iteração pode-se retornar um JSX
- Uma chave única pode ser passada aos componentes filhos

Repetição

```
return (
    <SafeAreaView>
        <View style={styles.viewBusca}>
            <TextInput placeholder='Digite o gênero do jogo' style={styles.input1} onChangeText={mudarTexto} />
            <Button title='Buscar' onPress={buscar} />

            <ScrollView style={styles.scrollView1}>
                {jogos.map( (item, key) => {
                    return (<Card titulo={item.title} descricao={item.short_description} />)
                } )}
            </ScrollView>
        </View>
    </SafeAreaView>
)
```

Aplicação de busca