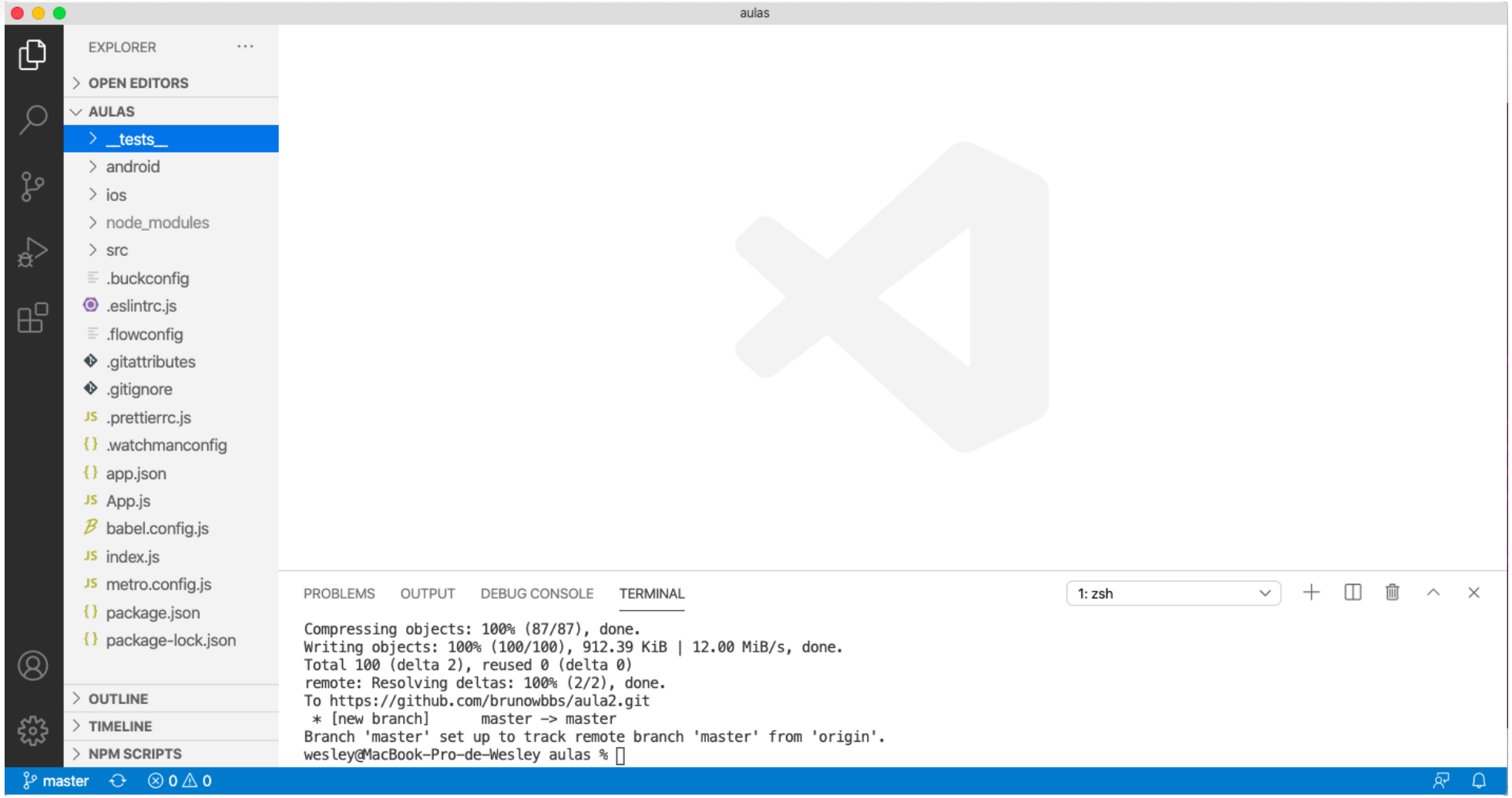
Unyleya
EDUCACIONAL

(2.1.1) Conceitos de React Native

Estrutura de um Projeto em React Native



Unyleya
EDUCACIONAL

Obrigado!

Unyleya EDUCACIONAL

Unyleya
EDUCACIONAL

(2.2.1) Class Component

Class Components

Os componentes em React Native que são declarados em formato de classe JavaScript são chamados de Class Components (ou Componentes tipo Classe) e assumem a seguinte sintaxe:

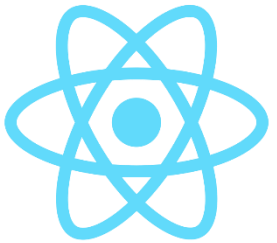
```
import React from 'react';
import { Text } from 'react-native';

class Cat extends React.Component {
  render() {
    return (
      <Text>Hello, I am your cat!</Text>
    );
  }
}

export default Cat;
```

- Um class component sempre deve herdar de React.Component (extends);
- Um class componente deve sempre invocar o método render(), que retorna um componente visual.

Class Components – setState( )

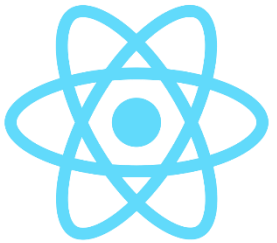
```
import React from 'react';
import {View, Text, Button } from 'react-native';

class Cat extends React.Component {

state = {
  valor: 0
}

maisUm = ()=>{
  this.setState({valor: this.state.valor + 1});
}

  render() {
    return (
      <View>
        <Text>{this.state.valor}</Text>
        <Button title="Adicionar" onPress={this.maisUm}/>
      </View>
    );
  }
}
export default Cat;
```

`setState()` agenda uma atualização para o objeto state de um componente. Quando o state muda, o componente responde renderizando novamente.



Class Components – Por que usar?

- Os Class Components é um tipo de componente muito verboso e possui uma complexidade extra;
- Versões anteriores a 0.59 do React Native, podia-se apenas utilizar componentes de classe;
- Podemos não implementar mais esse tipo de componente, mas no mercado ainda existem aplicações desenvolvidas com eles, e possivelmente teremos que fazer manutenções nesse tipo de sistema.
- Para reduzir a complexidade dos Componentes de classe, o time do React adicionou uma nova API de recursos chamados HOOKS.



Unyleya
EDUCACIONAL

Obrigado!

Unyleya EDUCACIONAL

Unyleya
EDUCACIONAL

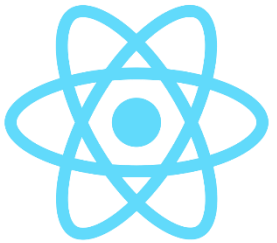
(2.2.2) Functional Components - HOOKS

Functional Components

Os componentes em React Native que são declarados em formato de função JavaScript são chamados de Functional Components (ou Componentes Funcionais) e assumem a seguinte sintaxe:

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

const Cat = () => {
  return <Text>Hello, I am cat!</Text>;
}

export default Cat;
```

- Um Componente Funcional é uma função Javascript que retorna um componente visual para ser exibido na tela.



Functional Components – useState( )

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

const Cat = () => {

  const [valor, setValor] = useState(0);

  const maisUm = ()=>{
    setValor(valor + 1);
  }

  return(
    <View>
      <Text>{valor}</Text>
      <Button title="Adicionar" onPress={maisUm}/>
    </View>
  );
}

export default Cat;
```

`useState()` é um Hook que te permite adicionar o state do React a um componente de função. Quando o state muda, o componente responde renderizando novamente.

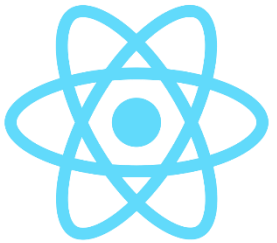
- Um Hook é uma função especial que te permite utilizar recursos do React.

Functional Components – useEffect( )

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

const Cat = () => {

  const [valor, setValor] = useState(0);

  useEffect(()=>{
    alert('executei');
  },[])

  const maisUm = ()=>{
    setValor(valor + 1);
  }

  return(
    <View>
      <Text>{valor}</Text>
      <Button title="Adicionar" onPress={maisUm}/>
    </View>
  );
}

export default Cat;
```

`useEffect()` permite executar efeitos colaterais em componentes funcionais.

```
useEffect(()=>{},[])
```



Unyleya
EDUCACIONAL

Obrigado!

Unyleya EDUCACIONAL

Unyleya
EDUCACIONAL

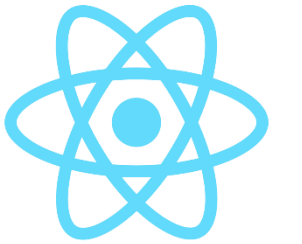
(2.3.1) JSX, Props e States

JSX

JSX é uma forma simplificada de criar elementos para serem utilizados em aplicações React. São bem similares com código HTML, e fornecem aos desenvolvedores uma forma mais intuitiva de criar os componentes de uma aplicação. O JSX não é interpretado pelo navegador, o React utiliza um transpilador para converter o código JSX.

```
<View>
  <Text/>
</View>
```

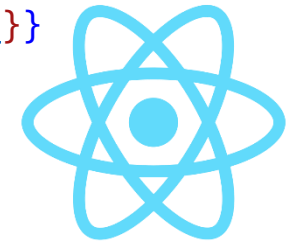
```
React.createElement(
  "div",
  React.createElement(
    "p",
    null,
  ),
);
```



Props

A maioria dos componentes podem ser personalizados quando criamos, com parâmetros diferentes. Esses parâmetros são chamados de props, abreviação de propriedades. Por exemplo, um componente básico do React Native é o `<Image/>`. Ao criar uma imagem, você pode usar uma *prop* chamada `source` para definir a imagem que será exibida.

```
import React from 'react';
import { Image } from 'react-native';

const Bananas = () => {
    return (
      <Image
        source={{uri:https://upload.wikimedia.org/wikipedia/commons/d/de/Bananavarieties.jpg}}
        style={{width: 193, height: 110}}/>
    );
}

export default Bananas;
```



States

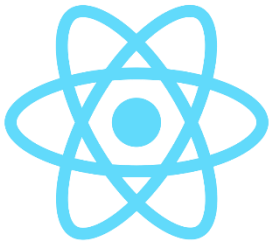
“States” é uma das partes mais importantes do React Native. Eles possuem o efeito de renderização instantânea toda vez que os seus valores são alterados. Eles que são responsáveis por causar o efeito dinâmico e reativo nas nossas aplicações. Um ponto importante é que os state são IMUTÁVEIS, não conseguimos alterar os seus valores, como se fossem variáveis normais.

ERRADO:

```
import React, {useState} from 'react’;

const [numero, setNumero] = useState(0);

numero = 10;
```

CERTO:

```
import React, {useState} from 'react’;

const [numero, setNumero] = useState(0);

setNumero(10);
```



Unyleya
EDUCACIONAL

Obrigado!

Unyleya EDUCACIONAL

Unyleya
EDUCACIONAL

(2.4.1) Componentes Básicos I

Text

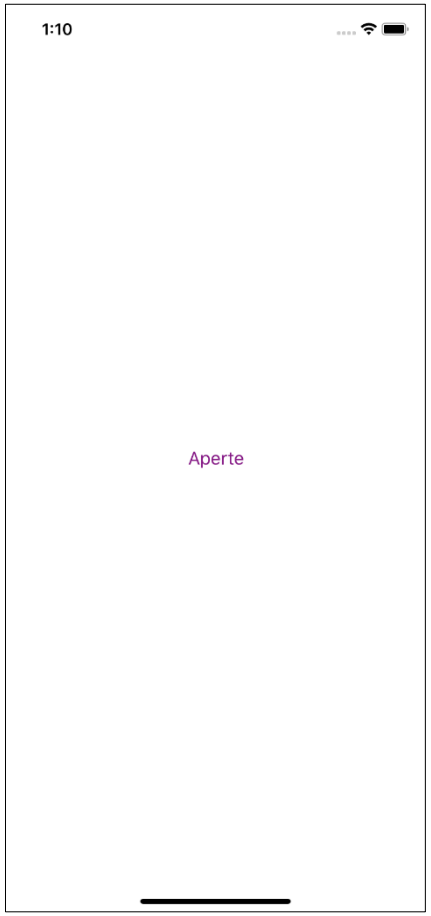
É um componente do React Native para exibição de textos.

```
import React from 'react';
import { Text } from 'react-native';
const App = () => {
  return (
    <Text>Hello World!</Text>
  )
}
export default App;
```

Button

É um componente do React Native que ao ser pressionado executa o comando a que está associado.

```
import React from 'react';
import { Button } from 'react-native';
const App = () => {
  return (
    <Button
      onPress={() => alert('Apertei!')}
      title="Aperte"
      color="#841584"
    />
  )
}
export default App;
```



TouchableOpacity

É um componente do React Native que ao ser pressionado executa o comando a que está associado. (Personalizável)

```
import React from 'react';
import { Text, TouchableOpacity } from 'react-native';
const App = () => {
  return (
    <TouchableOpacity
      onPress={() => alert('Apertei!')}
    >
      <Text >Aperte</Text>
    </TouchableOpacity>
  )
}
export default App;
```

Image

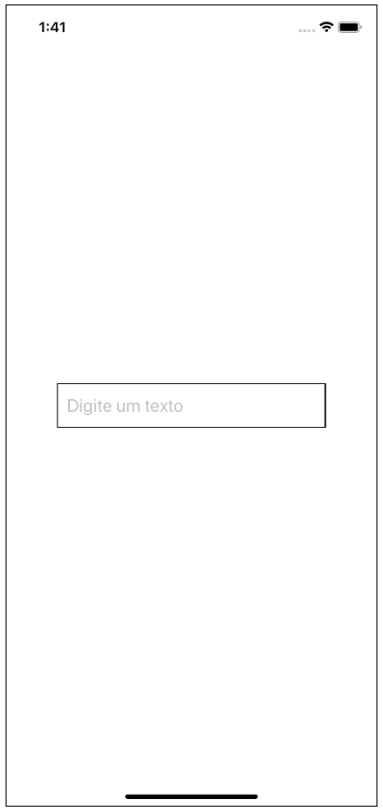
Exibe diferentes tipos de imagens, incluindo imagens de rede, recursos estáticos, imagens locais temporárias e imagens do disco local.

```
import React from 'react';
import { Image } from 'react-native';
const App = () => {
  return (
    <Image
      source={{ uri:'https://image.freepik.com/smartphone.jpg' }}
      style={{ width: 150, height: 150 }}
      resizeMode="contain"
    />

    <Image
      source={require('../src/logo.png')}
      style={{ width: 100, height: 100 }}
      resizeMode="cover"
    />
  )
}
export default App;
```

TextInput

É um componente React para exibição de textos.

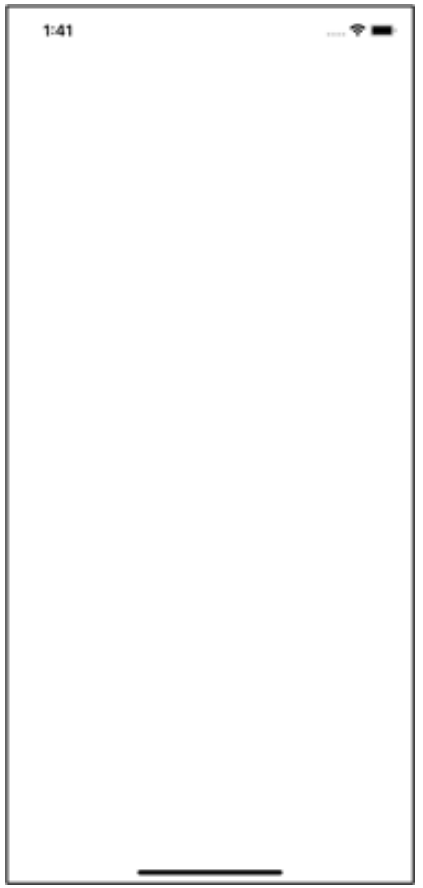
```
import React from 'react';
import { TextInput } from 'react-native';
const App = () => {
  return (
    <TextInput
      placeholder="Digite um texto"
      onChangeText={(value) => alert(value)
      } />
  )
}
export default App;
```



View

É o componente mais fundamental para criação de Interfaces.
Uma View que agrupa layouts.

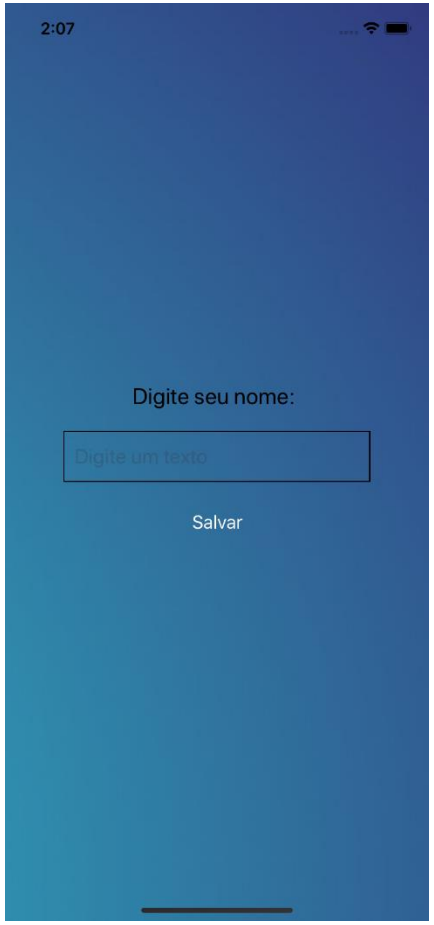
```
import React from 'react';
import { TextInput } from 'react-native';
const App = () => {

  return (
    <View>

    </View>
  )
}
export default App;
```



ImageBackground

É um componente View com uma imagem de fundo.

```
import React from 'react';
import { Text, Button,TextInput, ImageBackground } from 'react-
native';
const App = () => {
  return (
    <ImageBackground
     source={{ uri: 'https://images.unsplash.com/img.png' }} >
        <Text >Digite seu nome: </Text>
        <TextInput
          placeholder="Digite um texto"
          onChangeText={(value) => alert(value)} />
       <Button
          title="Salvar"
          onPress={() => alert('OK')} color="#fff" />
    </ImageBackground>
  )
}
export default App;
```



StatusBar

É um componente para controlar a barra de status do aplicativo.

```
import React from 'react';
import { StatusBar } from 'react-native';
const App = () => {
  return (
    <StatusBar
          hidden={false}
          barStyle="dark-content"
          backgroundColor="#fff" />
  )
}
export default App;
```

Unyleya
EDUCACIONAL

Obrigado!

Unyleya EDUCACIONAL

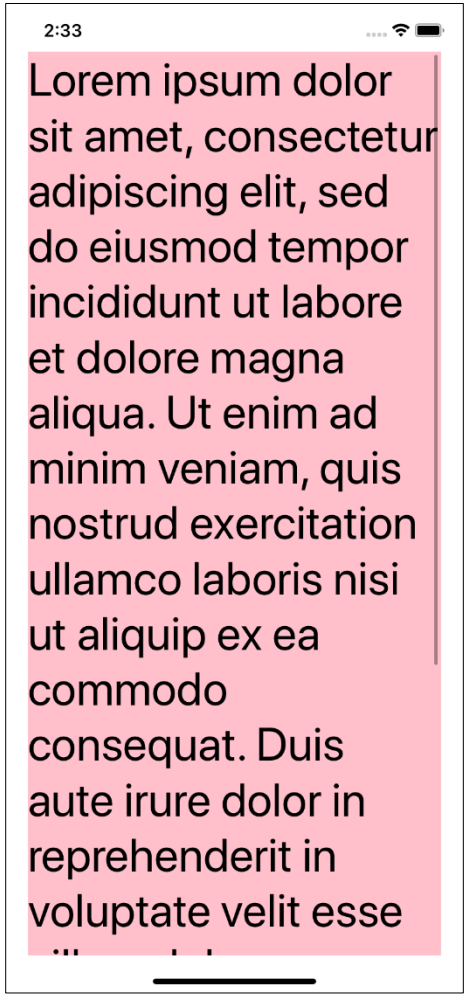
Unyleya
EDUCACIONAL

(2.4.2) Componentes Básicos II

ScrollView

É um componente React Native para rolagem de conteúdo. Deve ter uma altura limitada para funcionar.

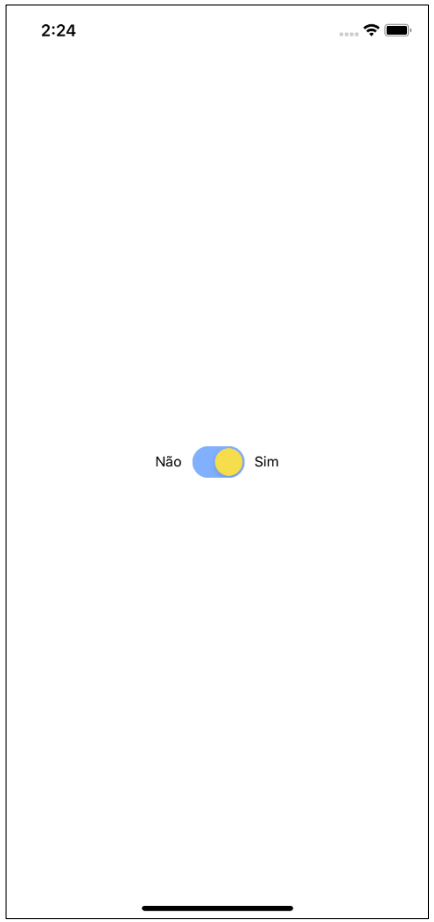
```
import React from 'react';
import { Text, ScrollView } from 'react-native';

const App = () => {
  return (

    <ScrollView >
      <Text>
        Lorem ipsum dolor sit amet,
        consectetur adipiscing elit, in
        culpa qui officia deserunt mollit
        anim id est laborum.
      </Text>
    </ScrollView>

  );
}

export default App;
```



Switch

Renderiza uma entrada booleana.

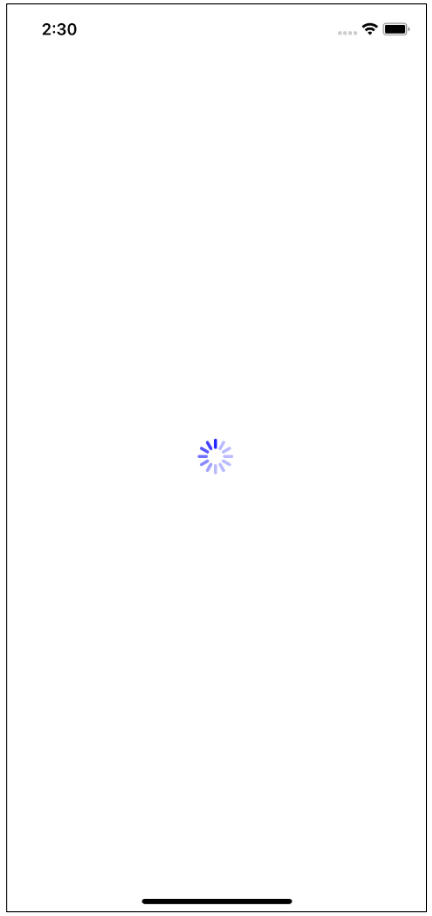
```
import React, { useState } from "react";
import { View, Text, Switch } from "react-native";
const App = () => {
  const [isEnabled, setIsEnabled] = useState(false);
  const toggleSwitch = () => setIsEnabled(!enable);

  return (
    <View >
      <Text>Não</Text>
      <Switch
        onValueChange={toggleSwitch}
        value={isEnabled}
      />
      <Text>Sim</Text>
    </View>
  );
}
export default App;
```



ActivityIndicator

Exibe um indicador de carregamento circular.

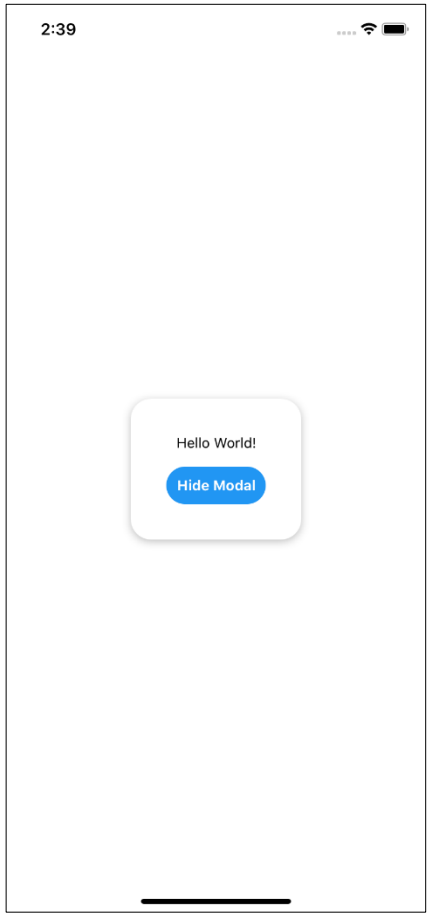
```
import React from "react";
import { ActivityIndicator } from "react-native";
const App = () => (
  <ActivityIndicator size="large" color="#00ff00" />
);

export default App;
```



Modal

O componente Modal é uma maneira básica de apresentar o conteúdo acima de uma exibição anexa.

```
import React, { useState } from "react";
import { Modal, View } from "react-native";

const App = () => {
  const [modalVisible, setModalVisible] = useState(false);
  return (
    <View >
      <Modal
        animationType="slide"
        transparent={true}
        visible={modalVisible}
      >
        <View >
        //Acrescente os compontes do modal aqui
        </View>
      </Modal>
    </View>
  );
};
export default App;
```



Unyleya
EDUCACIONAL

Obrigado!

Unyleya EDUCACIONAL

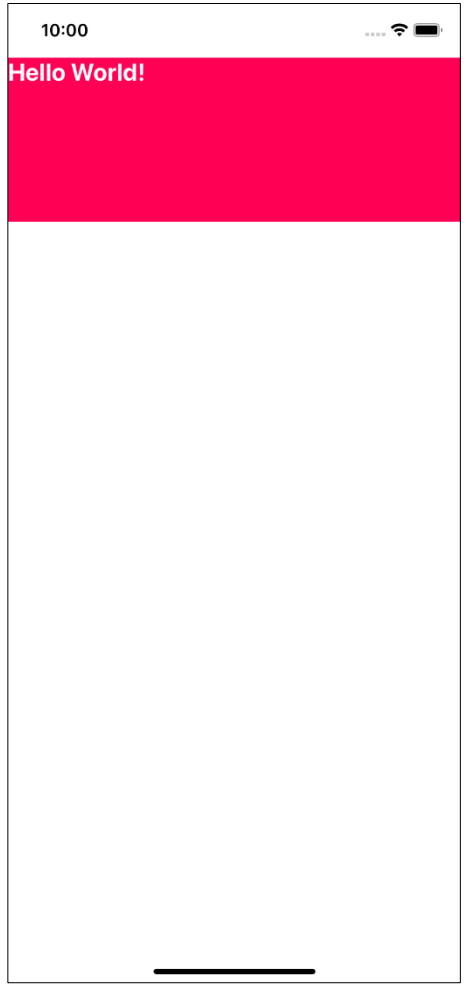
Unyleya
EDUCACIONAL

(2.5.1) Styles

Styles

Para conseguirmos utilizar o **CSS** no React Native, é necessário chamar uma nova propriedade, que seria o StyleSheet.

```
import React, { useState } from "react";
import { Text, View, StyleSheet } from "react-native";
const App = () => {
  return (
    <View style={styles.container}>
      <Text style={styles.texto}>Hello World!</Text>
    </View>
  );
}
const styles = StyleSheet.create({
  container: {
    marginTop: 50,
    height: 150,
    backgroundColor: "#ff0055"
  },
  texto: {
    fontSize: 22,
    color: '#ffffff',
    fontWeight: 'bold'
  }
});
export default App;
```



Unyleya
EDUCACIONAL

Obrigado!

Unyleya EDUCACIONAL

Unyleya
EDUCACIONAL

(2.5.2) Flexbox e Alinhamentos

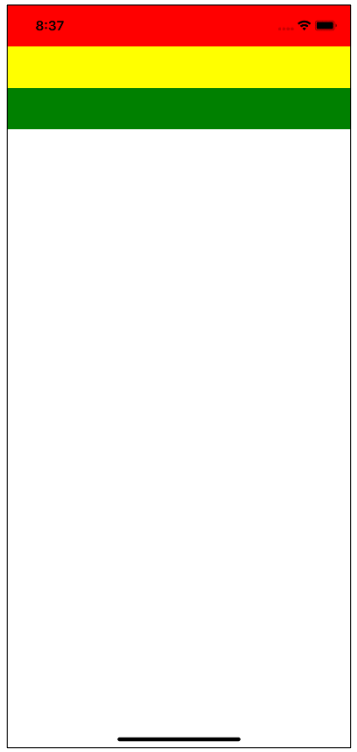
Flexbox e Alinhamentos

O FlexBox é a forma de organizar os componentes em tela no React-Native. Ele é projetado para prover um layout consistente para diferentes tamanhos de tela, ou seja: **responsivo.** Você normalmente utilizará uma combinação de flexDirection, alignItems, e justifyContent para se chegar ao layout correto.

Flexbox e Alinhamentos - flexDirection

Adicionar `flexDirection` a um style do componente determina o **eixo primário** de seu layout para os descendentes serem organizados horizontalmente (`row`) ou verticalmente (`column`). O default é `column`.

column

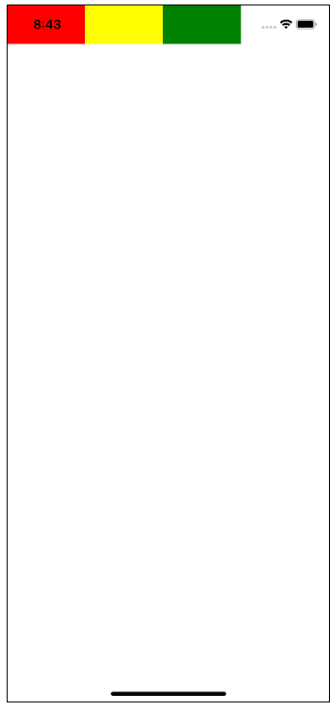



```
import React from 'react';
import { View, Text, Button } from 'react-native';
const App = () => {
  return (
    <View style={{ flexDirection: 'column' }} >
      <View style={{ backgroundColor: 'red', height: 50 }} />
      <View style={{ backgroundColor: 'yellow', height: 50 }} />
      <View style={{ backgroundColor: 'green', height: 50 }} />
    </View>
  );
}
export default App;
```

Flexbox e Alinhamentos - flexDirection

Adicionar `flexDirection` a um style do componente determina o **eixo primário** de seu layout para os descendentes serem organizados horizontalmente (`row`) ou verticalmente (`column`). O default é `column`.

row

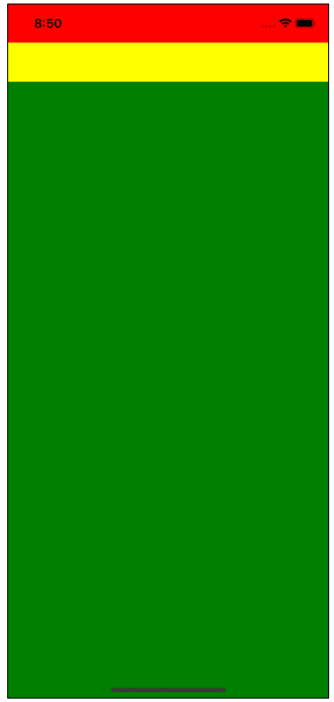



```
import React from 'react';
import { View, Text, Button } from 'react-native';
const App = () => {
  return (
    <View style={{ flexDirection: 'row' }} >
      <View style={{ backgroundColor: 'red', height: 50 }} />
      <View style={{ backgroundColor: 'yellow', height: 50 }} />
      <View style={{ backgroundColor: 'green', height: 50 }} />
    </View >
  );
}
export default App;
```

Flexbox e Alinhamentos - flex

Flex faz que os componentes tenham um nível de privilégio equivalente na divisão de espaçamento durante a tela:

column

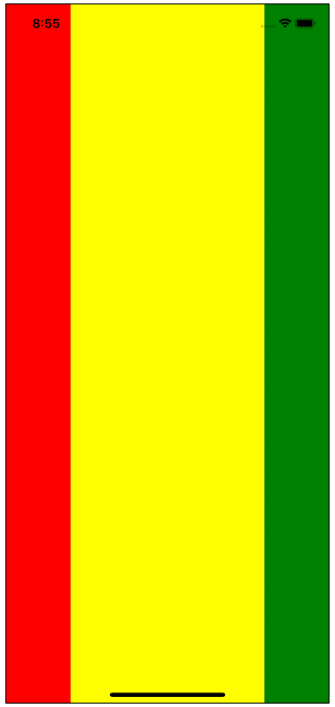



```
import React from 'react';
import { View, Text, Button } from 'react-native';
const App = () => {
  return (
    <View style={{ flexDirection: 'column', flex: 1, backgroundColor: 'pink' }} >
      <View style={{ backgroundColor: 'red', height: 50 }} />
      <View style={{ backgroundColor: 'yellow', height: 50 }} />
      <View style={{ backgroundColor: 'green', height: 50, flex: 1 }} />
    </View>
  );
}
export default App;
```

Flexbox e Alinhamentos - flex

Flex faz que os componentes tenham um nível de privilégio equivalente na divisão de espaçamento durante a tela:

row

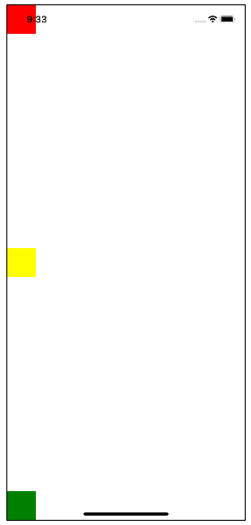



```
import React from 'react';
import { View, Text, Button } from 'react-native';
const App = () => {
  return (
    <View style={{ flexDirection: 'row', flex: 1, backgroundColor: 'pink' }} >
      <View style={{ backgroundColor: 'red', flex: 1 }} />
      <View style={{ backgroundColor: 'yellow', flex: 3 }} />
      <View style={{ backgroundColor: 'green', flex: 1 }} />
    </View>
  );
}
export default App;
```

Flexbox e Alinhamentos - justifyContent

Adicionar justifyContent a um style do componente determina a distribuição dos **descendentes** ao **longo do eixo primário**. Faz com que os descendentes sejam distribuídos pelo começo, centro, fim ou espaçados uniformemente. As opções disponíveis são respectivamente flex-start(começo), center(centro), flex-end(fim), space-around(espaço uniforme geral), e space-between(espaço entre os componentes herdeiros).

column

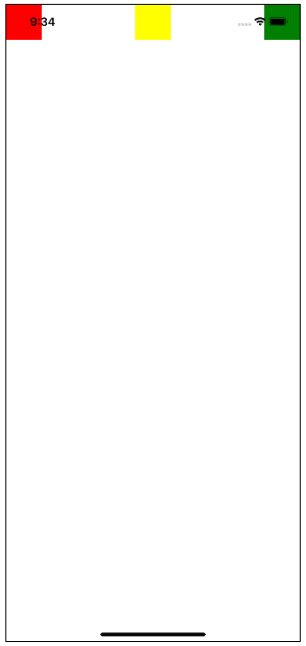


```
import React from 'react';
import { View, Text, Button } from 'react-native';
const App = () => {
  return (
    <View style={{ flexDirection: 'column', flex: 1, justifyContent: "space-between" }} >
      <View style={{ backgroundColor: 'red', height: 50, width: 50 }} />
      <View style={{ backgroundColor: 'yellow', height: 50, width: 50 }} />
      <View style={{ backgroundColor: 'green', height: 50, width: 50 }} />
    </View>
  );
}
export default App;
```

Flexbox e Alinhamentos - justifyContent

Adicionar justifyContent a um style do componente determina a distribuição dos **descendentes** ao **longo do eixo primário**. Faz com que os descendentes sejam distribuídos pelo começo, centro, fim ou espaçados uniformemente. As opções disponíveis são respectivamente flex-start(começo), center(centro), flex-end(fim), space-around(espaço uniforme geral), e space-between(espaço entre os componentes herdeiros).

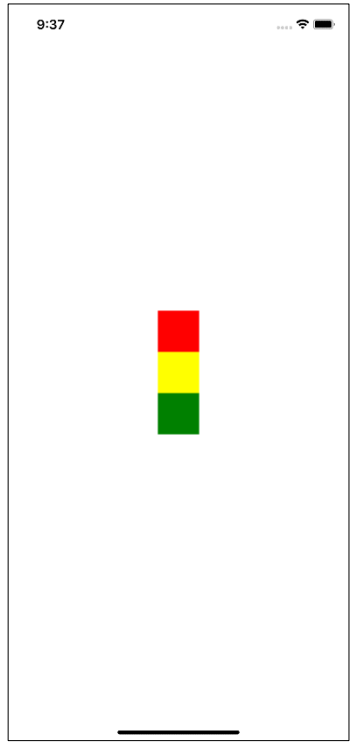
row




```
import React from 'react';
import { View, Text, Button } from 'react-native';
const App = () => {
  return (
    <View style={{ flexDirection: 'column', flex: 1, justifyContent: "space-between" }} >
      <View style={{ backgroundColor: 'red', height: 50, width: 50 }} />
      <View style={{ backgroundColor: 'yellow', height: 50, width: 50 }} />
      <View style={{ backgroundColor: 'green', height: 50, width: 50 }} />
    </View>
  );
}
export default App;
```

Flexbox e Alinhamentos - alignItems

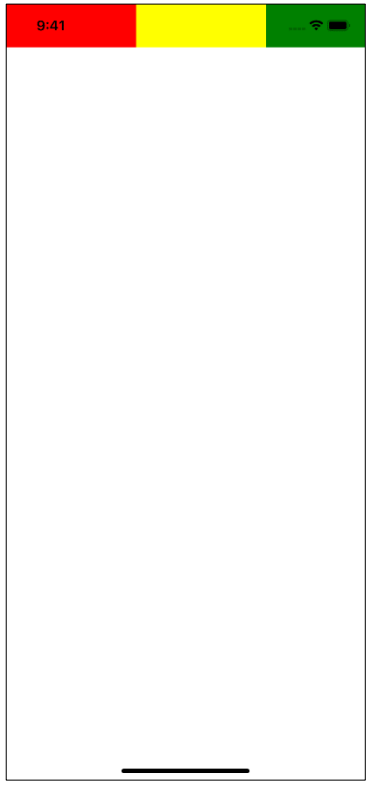
Adicionar alignItems a um style do componente determina o alinhamento dos descendentes ao longo do **eixo secundário**(se o eixo primário for row, então o secundário é column, e vice-versa). É para os descendentes serem alinhados pelo começo, centro, final, ou alargados para ocupar tudo. As opções disponíveis são respectivamente flex-start(começo), center(centro), flex-end(fim), e stretch(alargado).




```
import React from 'react';
import { View, Text, Button } from 'react-native';
const App = () => {
  return (
    <View style={{ flexDirection: 'column', flex: 1, justifyContent: "space-between" }} >
      <View style={{ backgroundColor: 'red', height: 50, width: 50 }} />
      <View style={{ backgroundColor: 'yellow', height: 50, width: 50 }} />
      <View style={{ backgroundColor: 'green', height: 50, width: 50 }} />
    </View>
  );
}
export default App;
```

Flexbox e Alinhamentos - flexWrap

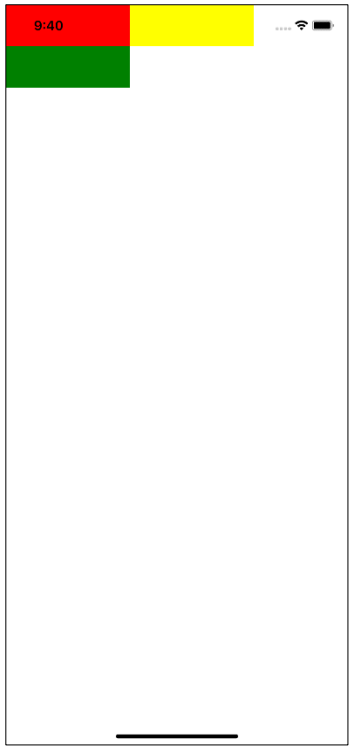
Digamos que precisemos realinhar itens baseando-se na quantidade de elementos de uma tela e cada componente tenha um valor fixo cujo 3 já ultrapasse a dimensão do dispositivo do usuário:




```
import React from 'react';
import { View, Text, Button } from 'react-native';
const App = () => {
  return (
    <View style={{ flexDirection: 'row', flex: 1, flexWrap: 'nowrap' }} >
      <View style={{ backgroundColor: 'red', height: 50, width: 150 }} />
      <View style={{ backgroundColor: 'yellow', height: 50, width: 150 }} />
      <View style={{ backgroundColor: 'green', height: 50, width: 150 }} />
    </View >
  );
}
export default App;
```

Flexbox e Alinhamentos - flexWrap

Digamos que precisemos realinhar itens baseando-se na quantidade de elementos de uma tela e cada componente tenha um valor fixo cujo 3 já ultrapasse a dimensão do dispositivo do usuário:




```
import React from 'react';
import { View, Text, Button } from 'react-native';
const App = () => {
  return (
    <View style={{ flexDirection: 'row', flex: 1, flexWrap: 'wrap' }} >
      <View style={{ backgroundColor: 'red', height: 50, width: 150 }} />
      <View style={{ backgroundColor: 'yellow', height: 50, width: 150 }} />
      <View style={{ backgroundColor: 'green', height: 50, width: 150 }} />
    </View >
  );
}
export default App;
```

Unyleya
EDUCACIONAL

Obrigado!

Unyleya EDUCACIONAL

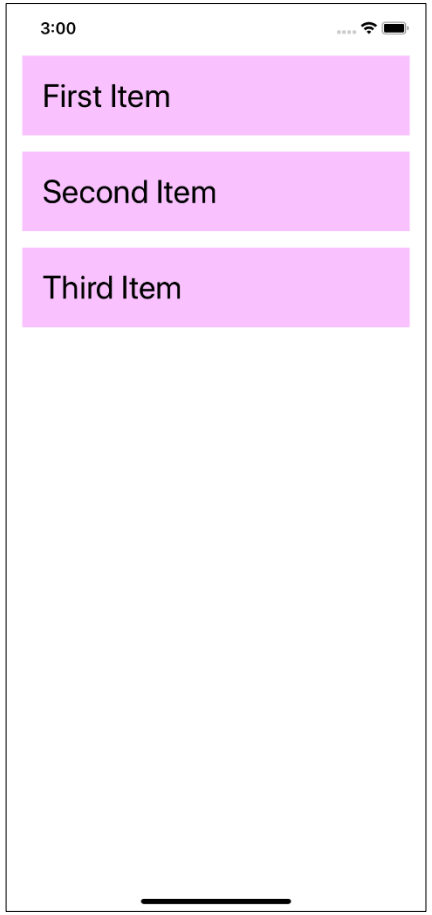
Unyleya
EDUCACIONAL

(2.6.1) Componente de Listagem

FlatList

É um componente de alto desempenho para renderizar listas simples e básicas para uma aplicação.

```
import React, { useState } from "react";
import { FlatList} from "react-native";
const DATA = [
  {id: "bd7acbeaba",title: "First Item"},
  {id: "3ac68afc", title: "Second Item"},
  {id: "58694a0f", title: "Third Item“ },
];
const App = () => {
  return (
    <FlatList
      data={DATA}
      renderItem={({ item }) => (
        <Text>{item.title}</Text>
      )}
      keyExtractor={(item) => item.id}
    />
  );
};
export default App;
```



Unyleya
EDUCACIONAL

Obrigado!

Unyleya EDUCACIONAL

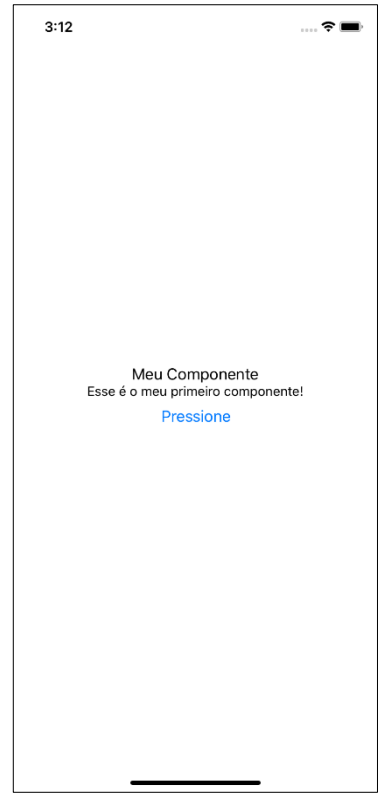
Unyleya
EDUCACIONAL

(2.7.1) Componentes Personalizados

Componentes Personalizados

É um componente de alto desempenho para renderizar listas simples e básicas para uma aplicação.

```
import React from 'react';
import { View, Text, Button } from 'react-native';
const aulas = () => {
  return <MeuComponente />;
}
export default aulas;
const MeuComponente = () => {
  return (
    <View>
      <Text>Meu Componente</Text>
      <Text>Esse é o meu primeiro componente!</Text>
      <Button title="Pressione" onPress={() => alert('OK')} />
    </View>
  )
}
```



Unyleya
EDUCACIONAL

Obrigado!

Unyleya EDUCACIONAL

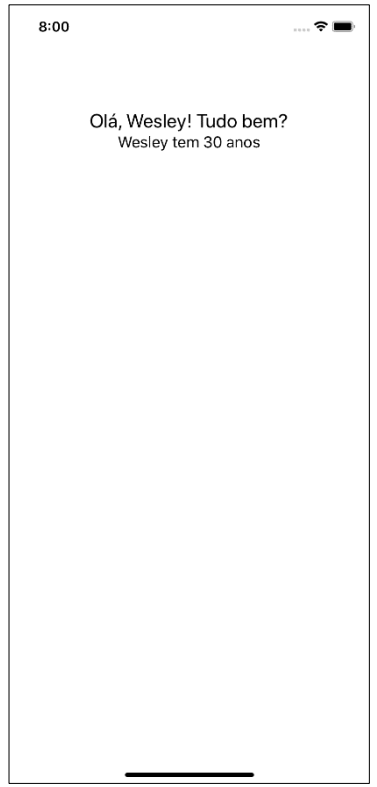
Unyleya
EDUCACIONAL

(2.8.1) Props entre componentes

Passando Props entre componentes

As Props em React Native são as propriedades que podemos passar de um componente para outro podendo ser utilizada internamente, seja para exibir ou aplicar alguma lógica própria do componente.

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

const aulas = () => {
  return <MeuComponente nome="Wesley" nascimento={1990} />;
}
export default aulas;

const MeuComponente = (props) => {
  return (
    <View >
      <Text>Olá, {props.nome}! Tudo bem?</Text>
      <Text>Wesley tem {2020 - props.nascimento} anos</Text>
    </View>
  )
}
```



Unyleya
EDUCACIONAL

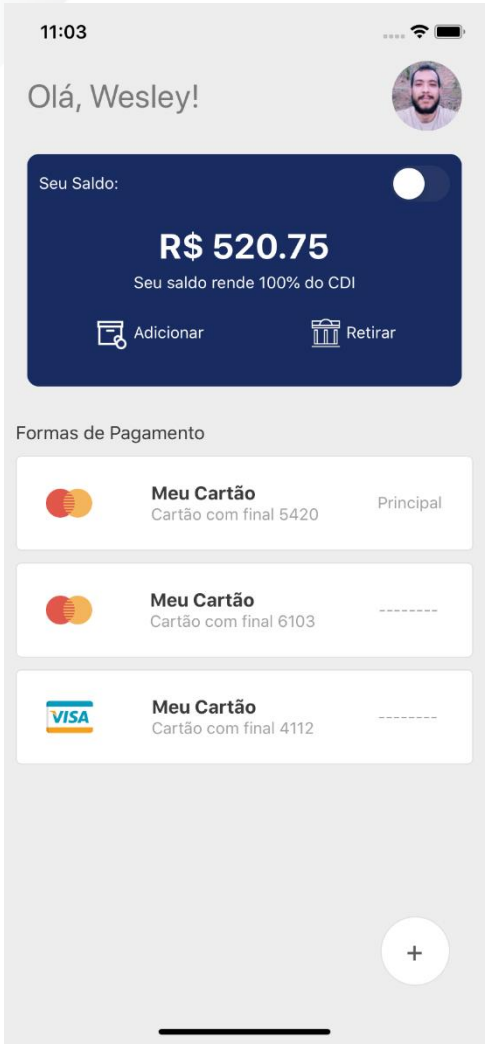
Obrigado!

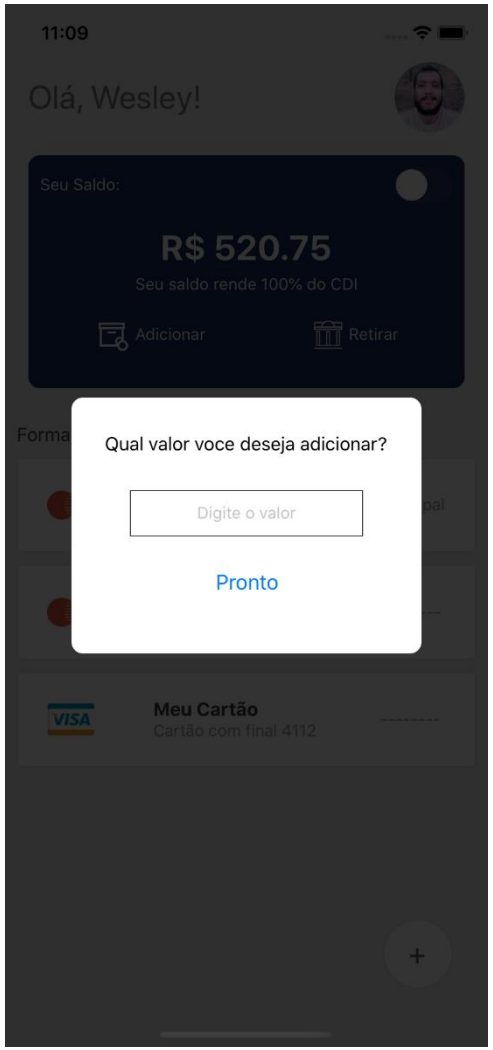
Unyleya EDUCACIONAL

Unyleya
EDUCACIONAL

(2.9.1) Criando aplicações na prática – Parte I

Projeto – Carteira digital





Regras:

- Crie uma interface para uma carteira digital.
- O aplicativo deve apresentar um saldo para o usuário.
- O usuário pode alterar esse saldo, portanto deve possuir uma opção para remover ou adicionar valores. (Sugestão: Utilize um modal para receber os valores que serão calculados do saldo).

Unyleya
EDUCACIONAL

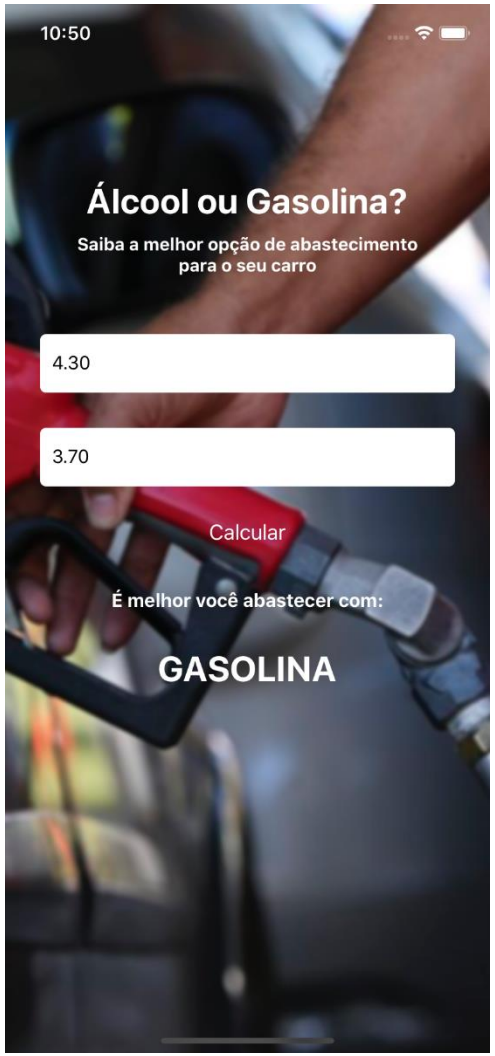
Obrigado!

Unyleya EDUCACIONAL

Unyleya
EDUCACIONAL

(2.9.3) Criando aplicações na prática – Parte III

Projeto – Álcool ou Gasolina?



Regras:

- **Multiplique o valor da gasolina no posto de combustível por 0,7.**
- Se o resultado for maior que o valor do álcool, vale abastecer com álcool.
- Se o resultado for menor que valor do álcool, abasteça com gasolina.

Exemplo: Se a gasolina custa 2,40 num posto e o álcool custa 1,40. Multiplicando 2,40 x 0,7 temos o resultado 1,68, que é maior que 1,40. Neste caso é melhor abastecer com álcool. Se o valor do álcool estivesse acima de 1,68, valeria mais a pena abastecer com gasolina neste posto.

Unyleya
EDUCACIONAL

Obrigado!

Unyleya EDUCACIONAL