Front-end: React

Aula 01 - React Native

Prof. MSc. Kelson Almeida

Google Classroom da turma…

Entre com o seu gmail (conta google) **pessoal**, **não entre com o email institucional.**

https://classroom.google.com/c/NzAzOTM4MjU1NzI0?cjc=6oebnc4

Agenda

- O que é React Native?
- Exercícios

O que é React Native?

- Uma biblioteca **JavaScript** criada pelo Facebook ***(Meta)*** para desenvolver aplicativos móveis nativos usando **React**.
- **Multiplataforma**, baseada em componentes, e usa JavaScript e JSX.

Vantagens do React Native

- **Desenvolvimento Multiplataforma:** Escreva um código e execute em iOS e Android.

- **Performance:** Performance próxima de um aplicativo nativo.

Vantagens do React Native

- **Comunidade e Suporte:** Grande comunidade e suporte robusto.
- **Hot Reloading:** Agiliza o processo de desenvolvimento.

Desvantagens?

- **Limitações de Componentes Nativos:** Nem todos os componentes estão disponíveis.
- **Performance para Casos Específicos:** Em certos casos, a performance pode ser inferior a um aplicativo totalmente nativo.
- **Curva de Aprendizado:** Requer conhecimento de React e JavaScript.

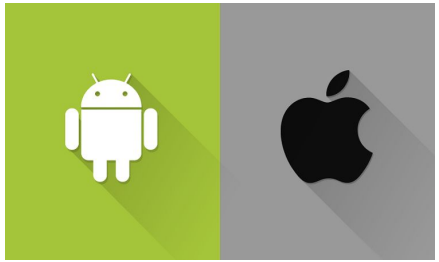
Desvantagens?

- **Limitações de Componentes Nativos:** Nem todos os componentes estão disponíveis.
- **Performance para Casos Específicos:** Em certos casos, a performance pode ser inferior a um aplicativo totalmente nativo.
- **Curva de Aprendizado:** Requer conhecimento de React e JavaScript.

Configuração do ambiente

- Instalação:
  - NodeJS e npm
  - VS Code
  - Expo go (Instalar no Android)

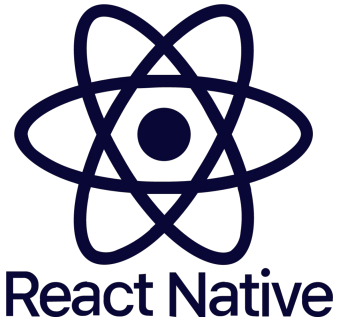
- Primeiro projeto:
  - **npx create-expo-app@latest MeuPrimeiroApp --template blank**



Componentes Funcionais

- **Componentes funcionais** são a base para construir interfaces no React Native, utilizando funções JavaScript para retornar elementos que representam a UI.

Componentes Funcionais



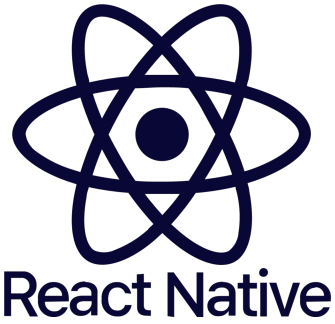
- **Vantagens:**
- Menos verboso que os componentes de classe.
- Melhor performance devido à menor sobrecarga.
- Facilita o uso de Hooks para adicionar funcionalidades como estado e ciclo de vida.
- Componentes funcionais promovem uma sintaxe mais limpa e concisa, especialmente com a introdução dos Hooks.

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

const HelloWorld = () => (
  <View>
    <Text>Hello, world!</Text>
  </View>
);
```

Entendendo JSX

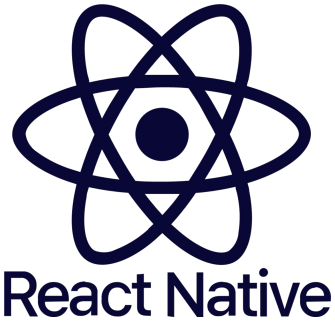
- **JSX** permite escrever a estrutura de componentes de UI dentro do JavaScript usando uma sintaxe que se assemelha ao HTML.
- JSX é transpilado para chamadas de função do React Native, permitindo a construção de UIs complexas de maneira declarativa.




```jsx
const App = () => (
  <View>
    <Text>Welcome to React Native!</Text>
  </View>
);
```

Introdução ao Hook useState

- **useState:** O hook useState permite adicionar estado local a um componente funcional.

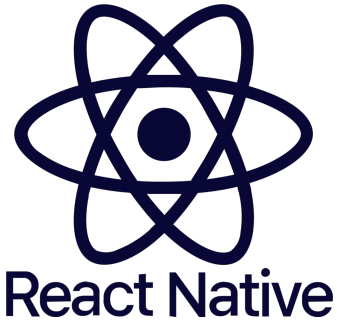



```jsx
import React, { useState } from 'react';
import { Button, Text, View } from 'react-native';

function Counter() {
  const [count, setCount] = useState(0);

  return (
    <View>
      <Text>Count: {count}</Text>
      <Button title="Increment" onPress={() => setCount(count + 1)} />
    </View>
  );
}
```

Passando e Recebendo Props



- **Props** são a forma de passar dados de um componente pai para um componente filho no React Native.
- props são usados para customizar componentes e promover a reutilização.

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

const Greeting = ({ name }) => (
  <View>
    <Text>Hello, {name}!</Text>
  </View>
);

const App = () => (
  <View>
    <Greeting name="React Native" />
  </View>
);
```

Prática: Construindo um Formulário Simples

- Utilizar componentes, estado, e props para construir um formulário simples com um campo de entrada e exibição do texto.

jsx Copy code

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

function SimpleForm() {
  const [text, setText] = useState('');

  return (
    <View>
      <TextInput
        value={text}
        onChangeText={setText}
        placeholder="Type here"
      />
      <Button title="Submit" onPress={() => alert(`Submitted: ${text}`)}
      <Text>You typed: {text}</Text>
    </View>
  );
}
```

Vamos praticar? [1] [com o prof]

- **Objetivo:** Criar um componente funcional que aceita um nome como prop e exibe uma mensagem de saudação.

Vamos praticar? [2] [com o prof]

- **Objetivo:** Implementar um componente Counter que exibe um número (inicialmente 0) e dois botões para incrementar e decrementar o valor. Utilize o hook useState.

Vamos praticar? [3] [com o prof]

- **Objetivo:** Criar um componente que inclui um <TextInput> para entrada de texto e um <Text> para exibir o texto digitado. Use useState para armazenar e atualizar o valor do texto.

Vamos praticar? [3] [com o prof]

- **Objetivo:** Criar um componente que inclui um <TextInput> para entrada de texto e um <Text> para exibir o texto digitado. Use useState para armazenar e atualizar o valor do texto.

Vamos praticar? [4] [com o prof]

- **Objetivo:** Desenvolver um componente que renderiza uma lista de itens (<Text>) a partir de um array de strings. Utilize .map() para criar os elementos da lista dinamicamente.