Professor Rodrigo Bossini
https://sites.google.com/site/professorrodrigobossini/

Introdução a React & React Native

1 Introdução

React é uma biblioteca Javascript criada pelo Facebook. Seu primeiro uso se deu em 2011 e sua primeira release pública foi lançada em 2013. Sua finalidade é a construção de interfaces gráficas, com foco particular para Single Page Applications (SPAs). Uma SPA, como o nome sugere, possui uma única página. Em geral, essa página tem uma área principal e controles e links que permitem a interação do usuário. Conforme ele interage, o conteúdo da área principal é atualizado, sem que a página tenha que ser completamente recarregada, evitando a navegação entre múltiplas páginas, submissões de formulários que podem fazer a tela "piscar" e assim por diante, o que pode causar interrupções na experiência do usuário.

Suas características principais são:

- A construção de componentes visuais é **declarativa**. Para cada estado possível da aplicação, o desenvolvedor especifica uma view simples. Conforme os dados são alterados (ou seja, o estado da aplicação), somente os componentes apropriados são atualizados graficamente.

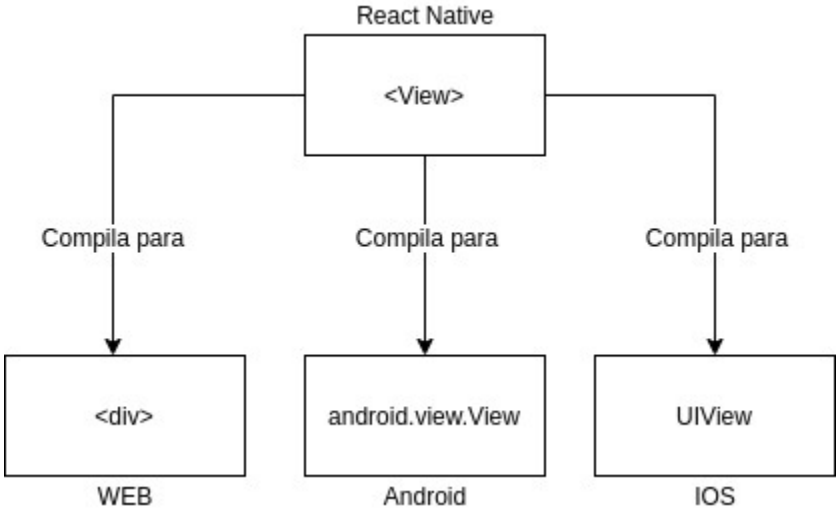
- O desenvolvimento é baseado em **componentes**. Cada componente lida com seu próprio estado e é responsável por especificar sua forma de exibição. Trata-se de um paradigma que promove naturalmente a **reusabilidade** de código.

- Pode ser utilizado em qualquer ambiente Javascript, inclusive renderizado no lado do servidor usando o Node, por exemplo.

Enquanto **React** é uma biblioteca apropriada para o desenvolvimento de aplicações WEB, **React Native** é um framework que viabiliza o uso do React para o desenvolvimento de aplicações para dispositivos móveis. Em particular, **React Native** possui uma coleção de componentes **React** que são **compilados para código nativo**.

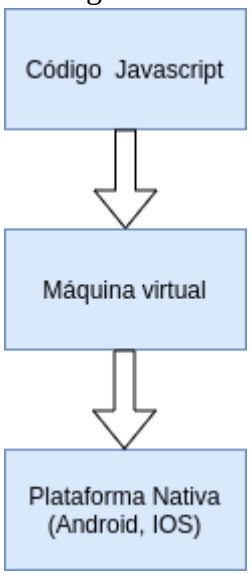
Uma aplicação desenvolvida com React Native pode ser destinada às principais plataformas existentes: Android e IOS, o que leva o nome de **desenvolvimento híbrido**. Como o nome sugere, embora o desenvolvimento ocorra em Javascript, a aplicação final utiliza componentes nativos, o que quer dizer que a aplicação resultante utiliza as mesmas APIs que uma aplicação desenvolvida nativamente utilizaria. A Figura 1.1 ilustra uma das principais características do React Native: ele possui componentes que são usados de forma genérica e que depois podem ser compilados para componentes apropriados, dependendo da plataforma que for ser utilizada.

Figura 1.1 – Um View do React Native pode ser compilado para diferentes componentes nativos



Por outro lado, **o código Javascript não é compilado para código nativo.** Há uma espécie de máquina virtual capaz de interpretá-lo e interagir com o sistema operacional adequadamente. Veja a Figura 1.2.

Figura 1.2



2 Criando aplicações React Native: Expo CLI versus React Native CLI

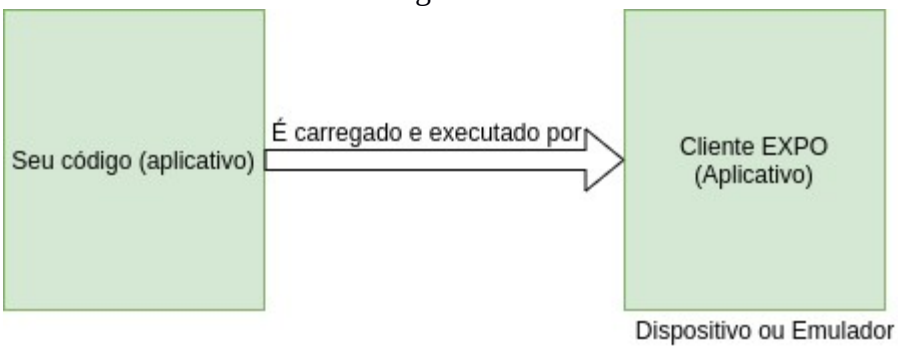
Aplicações React Native podem ser criadas de algumas formas diferentes. Duas delas são por meio do **Expo CLI** e do **React Native CLI**. A Tabela 2.1 mostra alguns itens a serem considerados.

Tabela 2.1

Expo CLI	**React Native CLI**
É um projeto de código fonte aberto de terceiros. Grátis.	Mantido oficialmente pelo time React Native e pela comunidade.
Ambiente de Desenvolvimento de Aplicações com maior nível de abstração	Ambiente de desenvolvimento “rústico”. Para começar a desenvolver, é preciso instalar o Android Studio, XCode etc.
Muitos utilitários para lidar com componentes. Acesso a itens específicos como a câmera simplificado etc.	Pouquíssimos utilitários como ocorre com o Expo.
Expo de certa forma é uma camada entre a sua aplicação e a plataforma nativa e o desenvolvedor fica limitado a esse ambiente.	Flexibilidade para integrar-se com qualquer código nativo.

Neste material iremos utilizar o Expo. Ele funciona como mostra a Figura 2.1.

Figura 2.1



Visite o Link 2.1 para saber mais sobre o Expo.

Link 2.1
https://expo.io/

3 Instalação do Expo e primeiro projeto

3.1 O primeiro passo é instalar o NodeJS. Ele inclui o Node Package Manager (npm), que será usado para instalar o Expo.

Uma vez instalado o Node, abra um terminal e execute o comando

npm install expo-cli --global

3.2 A seguir, crie um diretório em que serão salvos seus projetos. Seu nome pode ser algo como **react-native-workspace** ou algo similar.

3.3 Abra um terminal e navegue até o seu diretório com o comando

cd seu_diretorio

3.4 Para criar o primeiro projeto, use o comando

expo init primeiro-projeto

3.5 Escolha o template **Blank**. Por um lado, ele inclui o ambiente Expo (diferente da opção Bare Minimum) e, por outro, é simples o suficiente para começarmos.

3.6 Repare que foi criada uma pasta em seu diretório com o nome do projeto. Use o comando a seguir para navegar até ela

cd primeiro-projeto

3.7 Para colocar o ambiente em execução use o comando

npm start

3.8 Note que um servidor foi colocado em execução e uma aba do navegador foi utilizada para fazer uma requisição. Ela exibe o ambiente Expo. Inspecione as opções que ele oferece.

3.9 Note que há uma opção chamada **Run on Android Device/Emulator**. Basta clicar nela para que uma instância do emulador (caso você tenha uma instalada) seja colocada em execução. A aplicativo Expo será baixado e colocado em execução.

3.10 Note também que há um QR Code. Caso deseje, você pode baixar o Expo Client em seu dispositivo real e fazer a leitura do QR Code a fim de executar a aplicação nele.

3.11 Abra um novo terminal (não feche aquele que está executando o servidor) e digite o seguinte comando para abrir uma instância do VS Code associada ao diretório do projeto. O

terminal deve estar vinculado ao diretório antes de executar o comando (navegue com cd, caso necessário).

code .

3.12 Abra o arquivo **App.js**. Note que ali é definido um componente React simples, por meio de uma função Javascript, chamado **App**. A função devolve uma expressão JSX que contém um **View** que, por sua vez, possui um **Text** aninhado. Além disso, há a definição de um grupo de configurações CSS que é aplicada à View desse componente.

3.13 Vamos adicionar um botão logo abaixo do texto a fim de alterar seu conteúdo. Veja a Listagem 3.13.1.

Listagem 3.13.1

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

export default function App() {
return (
<View style={styles.container}>
<Text>Open up App.js to start working on your app!</Text>
<Button title="Alterar texto"/>
</View>
);
}
```

3.14 Note que o texto exibido faz parte do **estado** do componente. Como ele é definido como uma função, até a versão 16.8 do React, precisaríamos convertê-lo para uma classe. Porém, a partir desta versão há operadores que permitem a definição de estado mesmo para funções. O primeiro passo é importar o operador **setState**. A seguir, vamos definir um vetor que armazena uma variável com o texto atual e uma função que será colocada em execução para alterar o texto. O vetor é devolvido pelo Hook setState. Veja a Listagem 3.14.1.

Listagem 3.14.1

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

export default function App() {
const [outputText, setOutputText] = useState('Texto inicial');
return (
<View style={styles.container}>
<Text>Open up App.js to start working on your app!</Text>
<Button title="Alterar texto"/>
</View>
);
}
```

3.15 Desejamos exibir o que está armazenado na variável outputText. Ela é uma expressão Javascript e, como estamos em um contexto JSX, é preciso englobá-la com chaves. Veja a Listagem 3.15.1.

Listagem 3.15.1

```
<Text>{outputText}</Text>
```

3.16 O texto será alterado quando o botão for clicado. O clique deve ser tratado com a propriedade **onPress** do botão. Associe a ela uma função que chama a função devolvida pelo operador useState. Veja a Listagem 3.16.1.

Listagem 3.16.1

```
<Button title="Alterar texto" onPress={() => {setOutputText("Novo Texto")}}/>
```

3.17 Caso deseje, é possível converter a definição do componente para utilizar uma classe, como na Listagem 3.17.1. Não deixe de comentar a definição prévia, feita com uma função.

Listagem 3.17.1

```
export default class App extends React.Component{
constructor (props){
super (props);
this.state = {
outputText: "Novo Texto"
}
}
render (){
return (
<View style={styles.container}>
<Text>{this.state.outputText}</Text>
<Button
title="Alterar texto"
onPress={() => {this.setState({outputText: "Texto Inicial"})}}
/>
</View>
);
}
}
```

3.18 O exemplo da Listagem 3.18.1 mostra um contador simples. Uma variável que é incrementada a cada vez que o botão é clicado. Note que ela faz parte do estado do componente.

Listagem 3.18.1

```
export default class App extends React.Component{
constructor (props){
super (props);
this.state = {
contador: 0
}
}
render (){
return (
<View style={styles.container}>
<Text>{this.state.contador}</Text>
<Button
title="Alterar texto"
onPress={() => {this.setState({contador: this.state.contador + 1})}}
/>
</View>
);
}
}
```

Referências

React – A JavaScript library for building user interfaces. 2020. Disponível em <https://reactjs.org/> Acesso em janeiro de 2020.

React Native · A framework for building native apps using React. 2020. Disponível em <https://facebook.github.io/react-native/>. Acesso em janeiro de 2020.