**Semana OmniStack – 10° Edição**

*Mobile*

# Abordagem tradicional



# Abordagem do React Native



O React Native implementa dentro da aplicação o framework JSCore: dá o entendimento JS para o SO mobile, ou seja, IOS e Android passam a interpretar código JS para construir e manipular a interface de acordo com a plataforma nativa.

O código **NÃO** é convertido.



O Expo é um framework para react native para utilizar as funcionalidades do celular (câmera, mapas, geolocalização, sensor..)



Existem alguns casos que o expo ainda não suporta, como o bluethooth.

# Criação do Projeto

Instalação do expo: yarn global add expo-cli OU npm install -g expo-cli

Para criar o projeto após a instalação: expo init mobile – escolha o template blank – marque Y para o yarn.

Entre na pasta do projeto e rode yarn start.

Após isso será aberto uma página do expo e será gerado um QRCode. Com isso, instale o app expo no seu celular e use um leitor para ler o QRCode. Se desejar rodar no emulador na máquina, escolha a opção run on android/device emulator OU run on ios simulator.

Guia de possíveis erros do expo: <https://github.com/Rocketseat/expo-common-issues>

# Principais Diferenças

ReactJS x React Native

|  |  |
| --- | --- |
| **ReactJS** | **React Native** |
| Utilização de tags HTML | Utilização de tags próprias (View, Text..) |
| Estilo com class e id - CSS | Estilo com style – todos os componentes - CSS |

# Estrutura do Projeto

* Babel.config: consegue fazer a conversão do código JS de acordo com as versões das plataformas em questão. O código q voce escreveu pode não rodar na versão X do android, esse arquivo serve para fazer essa conversão. Utilizado também no ReactJS.
* App.json: informações sobre o app: modo da tela, ícone do app..
* Assets: ícone e splash screen do app. Se for mudar lembre-se de deixar nas mesmas dimensões que veio previamente de exemplo.
* Src: todo o código da app.
* Pages: páginas da aplicação.

# Projeto

## Criação da Navegação

React Navigation: <https://reactnavigation.org/docs/en/getting-started.html>

Comando: yarn add react-navigation

Se estiver utilizando o expo deve rodar também: expo install react-native-gesture-handler react-native-reanimated react-native-screens react-native-safe-area-context

Rode também o comando:

yarn add react-navigation-stack @react-native-community/masked-view

### Navegação por pilhas

Comando: yarn add [react-navigation-stack](https://github.com/react-navigation/stack)

Funcionamento: ocorre via botão, por uma ação do usuário.

//fica por fora de toda a navegação da aplicação

import { createAppContainer } from 'react-navigation'

import { createStackNavigator } from 'react-navigation-stack'

import Main from './pages/Main'

import Profile from './pages/Profile'

//componente

const Routes = createAppContainer(

    //passagem do objeto com as rotas da aplicação

    createStackNavigator({

        //mudança do título da aplicação

        //screen: qual componente será renderizado

        //opções específicas da tela: navigationOption:{ }

        Main: {

            screen: Main,

            navigationOptions: {

                //headerTitle: pode colocar uma imagem/componente

                //o title coloca um título em String

                title: 'DevRadar'

            }

        },

        Profile: {

            screen: Profile,

            navigationOptions: {

                title: 'Perfil no GitHub'

            }

        }

    }, {

        //opções aplicadas a todas as telas

        defaultNavigationOptions: {

            //muda a cor do título

            headerTintColor: '#FFF',

            //estilização do header

            headerStyle: {

                backgroundColor: '#7D40E7'

            }

        }

    })

)

export default Routes

App.js

import React from 'react';

import { StatusBar } from 'react-native'

import Routes from './src/routes'

export default function App() {

  return (

    <>

      <*StatusBar* *barStyle*='light-content' *backgroundColor*='#7D40E7'/>

      <*Routes* />

    </>

  );

}

#### Navegando entre as rotas

Quando você cria a navegação é passado propriedades gerais sobre elas ao componente em que ela está sendo utilizada.

function Main({ *navigation* }) {

Passa o nome da rota e um objeto com os parâmetros que você quer q a próxima rota tenha acesso.

<*Callout* *onPress*={() => {

                    navigation.navigate('Profile', { github\_username: 'CarlaPonciano' })

                }}>

### Navegação por abas

Cria um menu superior ou inferior que o usuário pode navegar, direita e esquerda também.

### Navegação Drawer

Puxa um menu da esquerda para a direita e aparece um menu com opções – menu de hambúrguer.

## WebView

<https://docs.expo.io/versions/latest/sdk/webview/>

Comando: expo install react-native-webview

import { WebView } from 'react-native-webview';

//pega a rota que está recebendo

function Profile({ *navigation* }) {

    //pega o parâmetro enviado

    const githubUsername = navigation.getParam('github\_username')

    return <*WebView* *style*={{ flex: 1 }} *source*={{ uri: `https://github.com/${githubUsername}` }} />

}

## Ícones

<https://material.io/resources/icons/?style=baseline>

//ícones

import { MaterialIcons } from '@expo/vector-icons'

<*TouchableOpacity* *onPress*={() => {}} *style*={styles.loadButton}>

                    <*MaterialIcons* *name*="my-location" *size*={20} *color*="#FFF" />

                </*TouchableOpacity*>

## Subir o teclado

Utilize a biblioteca Keyboard do react-native e olhe a documentação para ver como fazer (verifique a altura do teclado e jogue o teclado para cima)

## Maps

Maps: <https://docs.expo.io/versions/latest/sdk/map-view/>

Instalação: expo install react-native-maps

import MapView from 'react-native-maps'

function Main() {

    return <*MapView* *style*={styles.map} />

}

export default Main

const styles = StyleSheet.create({

  map: {

      flex : 1

  }

})

## Localização

<https://docs.expo.io/versions/latest/sdk/location/>

Comando: expo install expo-location

//requestPermissionAsync: pede permissões ao usuário para utilizar a localização

//getCurrentPositionAsync: pega a localização do usuário

import { requestPermissionsAsync, getCurrentPositionAsync } from 'expo-location'

function Main() {

    const [currentRegion, setCurrentRegion] = useState(null)

    //carrega a localização do usuário para abrir mostrando no mapa

    //[] será executada apenas 1 vez

    //chamada assíncrona

    useEffect(() => {

        async function loadInitialPosition() {

            //pede inicialmente a permissão

            //retorna um objeto com várias informações, pega o granted - boolean (deu permissão ou não)

            const { granted } = await requestPermissionsAsync()

            //se sim, busca as coordenadas

            if (granted) {

                //pode utilizar o GPS do usuário para pegar uma localização mais precisa, basta passar o objeto com enableHighAccuracy

                const { coords } = await getCurrentPositionAsync({

                    enableHighAccuracy: true,

                })

                const response = await getCurrentPositionAsync({

                    enableHighAccuracy: true,

                })

*console*.log(response)

                const { latitude, longitude } = coords

*console*.log(coords.latitude, coords.longitude)

                setCurrentRegion({

                    latitude,

                    longitude,

                    //cálculos navais(complexos) para obter o zoom no mapa

                    latitudeDelta: 0.04,

                    longitudeDelta: 0.04

                })

            }

        }

        loadInitialPosition()

    }, [])

    if(!currentRegion) return null

    return <*MapView* *initialRegion*={currentRegion} *style*={styles.map} />

}

## Marcação no Mapa

//Marker: marcação dentro do mapa

import MapView, { Marker } from 'react-native-maps'

<*MapView* *initialRegion*={currentRegion} *style*={styles.map}>

            <*Marker* *coordinate*={{ latitude: -20.5118397, longitude: -43.7133091}}>

                <*Image* *style*={styles.avatar} *source*={{ uri: 'https://avatars2.githubusercontent.com/u/49000683?s=460&v=4' }} />

            </*Marker*>

            <*Marker* *coordinate*={{ latitude: -20.5079666, longitude: -43.7180824}}>

                <*Image* *style*={styles.avatar} *source*={{ uri: 'https://avatars2.githubusercontent.com/u/49000683?s=460&v=4' }} />

            </*Marker*>

        </*MapView*>

const styles = StyleSheet.create({

  map: {

      flex : 1

  },

  avatar: {

      width: 54,

      height: 54,

      borderRadius: 4,

      borderWidth: 4,

      borderColor: '#FFF'

  }

})

### Balão de Informações no marker

import MapView, { Marker, Callout } from 'react-native-maps'

<*Marker* *coordinate*={{ latitude: -20.5118397, longitude: -43.7133091}}>

                <*Image* *style*={styles.avatar} *source*={{ uri: 'https://avatars2.githubusercontent.com/u/49000683?s=460&v=4' }} />

                <*Callout*>

                    <*View* *style*={styles.callout}>

                        <*Text* *style*={styles.devName}>Carla Ponciano</*Text*>

                        <*Text* *style*={styles.devBio}>Tenho como valor que se podemos pensar em uma solução criativa, podemos desenvolvê-la e aplicá-la no mercado.</*Text*>

                        <*Text* *style*={styles.devTechs}>React Native, ReactJS, NodeJS</*Text*>

                    </*View*>

                </*Callout*>

            </*Marker*>

callout: {

    width: 260

  },

  devName: {

    fontWeight: 'bold',

    fontSize: 16

  },

  devBio: {

    color: '#666',

    marginTop: 5

  },

  devTechs: {

    marginTop: 5

  }

# Realizar chamadas à API

Rode o comando: yarn add axios

import axios from 'axios'

const api = axios.create({

    //depende do SO e do ambiente

    //celular: IP que aparece no expo: xx:porta\_no\_node

    //baseURL: 'http://192.168.56.1:3333'

    //emulador ios: coloca localhost no ip

    //emulador android: coloca o ip do expo, se n funcionar coloque: 10.0.2.2

    baseURL: 'http://192.168.56.1:3333'

})

export default api

//importação da api

import api from '../services/api'

//carregar usuários: carrega toda vez que o usuário clica no botão

    async function loadDevs() {

        const { latitude, longitude } = currentRegion

*console*.log(latitude, longitude, techs)

        try {

            const response = await api.get('/search', {

                params: {

                    latitude,

                    longitude,

                    techs

                }

            })

*console*.log(response.data.devs)

            setDevs(response.data.devs)

        } catch(err) {

*console*.error(err);

        }

    }

# Curiosidades

## PWA

Progressive web apps – é possível utilizar o expo no desenvolvimento de PWAs. Construção de aplicações web com a sintaxe do React Native.

## Publicação

<https://docs.expo.io/versions/latest/workflow/publishing/>

você vai ter que alugar o mac na internet para gerar o .ipa e depois subir na Apple Store

<https://www.macincloud.com/>

## Documentação Expo

<https://docs.expo.io/versions/latest/>