soluções mobile

prof. Thyerri Mezzari

React Native:
primeiros passos

Ecossistema / Stack

JavaScript ou **JS** é uma linguagem de programação interpretada, juntamente com HTML e CSS, o JavaScript é uma das três principais tecnologias WEB. É uma linguagem amplamente usada em navegadores web (client-side) e também usada em servidores através de Node.js.

Baseada em *ECMAScript*, se caracteriza como uma linguagem multiparadigma com suporte a estilos de programação orientados a eventos, funcionais e imperativos (orientado a objetos e prototype-based).

...e não têm nada haver com Java!

Ecossistema / Stack

Em relação ao *React Native*, nosso uso de **JavaScript** *(e/ou TypeScript)* vai desde a codificação de nossa aplicação, a até configurações de projeto e uso de utilitários de linha de comando. Quanto a este último, ao trabalhar com RN devemos voltar de cabeça ao mundo "JavaScript" e focar no uso de ferramentas como o NPM e Node.js.

Sempre que nosso projeto necessitar de um novo plugin / pacote de terceiros para completar as funcionalidades de nossa aplicação React Native o comando indicado será `` `npm install xxx-yyy` `` para instalar uma nova dependência. Sempre que formos rodar um novo comando para o RN como por exemplo rodar nossa aplicação poderemos fazer `` `npx react-native run-android` `` (ou `` `npx expo android` ``).

Ou... **npx expo start**

Editores de Código

Esse tópico será breve, como estamos trabalhando com tecnologias de base "web" e open-source, podemos utilizar o editor de código que acharmos mais adequado para o trabalho de core do seu projeto RN/Expo.

Ou seja, qualquer editor de código que saiba lidar com *JavaScript* será útil e válido. **Sublime Text, VS Code ou vim** todos ótimas alternativas open-source. Uma ótima vantagem de trabalhar com RN é não estarmos presos a IDEs pesadas como *Android Studio* 100% do tempo.

Caso o seu editor de preferência seja o VS Code, recomendo estes dois artigos de personalização:

https://medium.com/react-native-training/vscode-for-react-native-526ec4a368ce
https://blog.rocketseat.com.br/ambiente-desenvolvimento-javascript/

Como o RN funciona

JS <-> BRIDGE <-> NATIVE

Como funciona o framework

Conforme já introduzido a base "programável" do ***React Native*** é o **JavaScript** (e seus derivados, com o TypeScript). Sabendo disso fica interessante imaginar com o framework funciona "por baixo dos panos" uma vez que diferente do PhoneGap/Cordova/ionic o RN não utiliza HTML ou outras técnicas baseadas em *webviews* para fazer um app funcionar.

Inicialmente a "mágica" do RN funciona desta forma: Um app desenvolvido com React Native trabalha inicialmente com **duas "threads"** importantes. A *primeira* que chamamos de "main thread" trabalha 100% na camada nativa do App. Ela cuida do contato com o iOS ou Android, e também é ela que renderiza os elementos/componentes visuais em tela e captura os gestos (toque, arrastar, soltar) e interações dos usuários.

Como funciona o framework

A *segunda* thread é a especfíca do React Native e é ela responsável por executar o JavaScript de nossa aplicação em uma engine separada para interpretação da linguagem. Ou seja essa a thread que cuida da nossa "lógica de negócio" da Aplicação, é nessa thread que nós desenvolvedores normalmente trabalhamos.

Muito Importante: essas duas "threads" nunca se conversam diretamente (vamos melhorar isso adiante) e na teoria nunca deveria "bloquear" uma a outra.

E como essas "threads" interagem?

Sabido que as duas "threads" principais do RN nunca se conversam diretamente, fica claro o ponto de que deve haver um "intermediário" para comunicação entre elas.

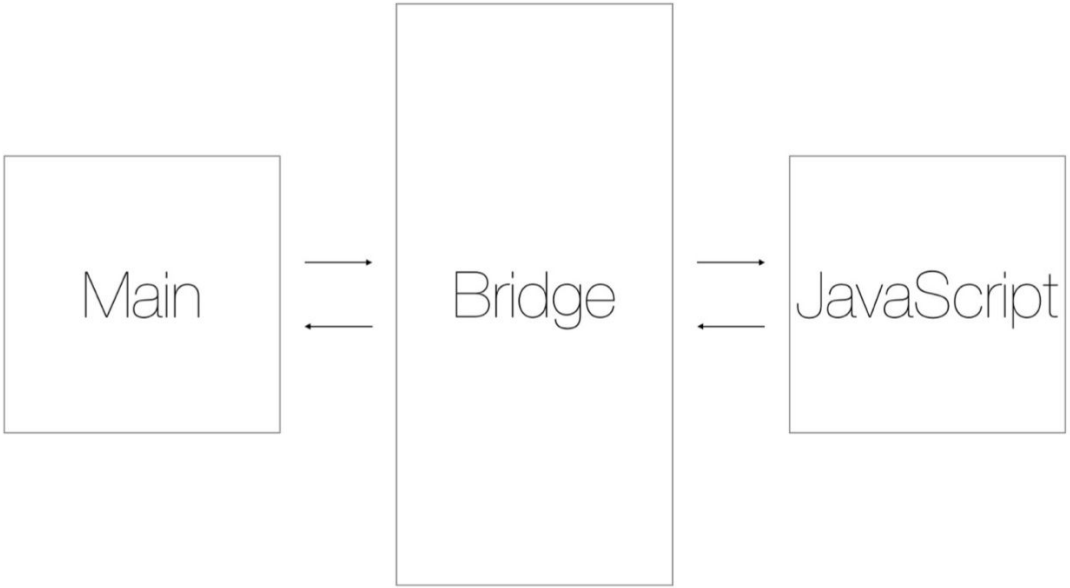
E é a partir deste paradigma que iremos definir o funcionamento do que é conhecido na comunidade RN como "**bridge**" (ponte), que basicamente o core do framework open-source React Native.

E como essas "threads" interagem?

Partiremos do início, essa bridge têm três importantes características:

- ***Asynchronous***: Essa bridge garanta comunicação assíncrona entre as camadas e que nunca deverá "bloquear" uma a outra.
- ***Batched***: Deverá transferir e comunicar os comandos solicitados de forma otimizada e sequencial.
- ***Serializable***: As duas camadas nunca trocaram exatamente os mesmos dados, sempre uma forma de contato "serializada" (formato JSON comumente) será usada entre as partes.

E como essas "threads" interagem?



Funcionamento no iPhone OS / iPad OS

Funcionamento no Android

Relevante ao assunto

https://trac.webkit.org/wiki/JavaScriptCore

https://github.com/react-native-community/jsc-android-buildscripts

https://github.com/facebook/hermes

Estrutura do Projeto

src/App.js

Estrutura do Projeto

Um dos principais colaboradores do Facebook envolvido por trás dos projetos React dentro da empresa é o **Dan Abramov**, como o Facebook nunca definiu um padrão oficial de nome de pastas e organizações de arquivos para projeto React e React Native, ele e outros membros da equipe oficial vivem sendo perguntados sobre "qual o melhor jeito de estrutura um projeto em React ou RN", como resposta ele desenvolveu este site:

https://react-file-structure.surge.sh

https://legacy.reactjs.org/docs/faq-structure.html

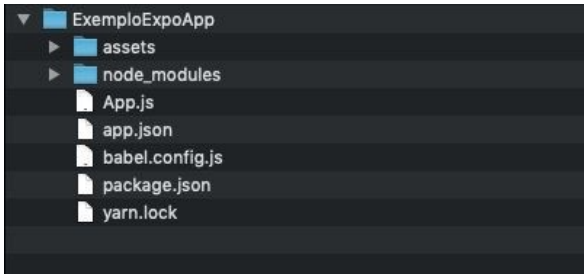
Estrutura do Projeto

Uma ótima sugestão de organização:

src/
- **actions**: funções úteis que normalmente "produzem" algum conteúdo, por exemplo conectar em um servidor e baixar dados
- **components**: componentes/trechos de código que são reutilizados em diversas telas ao longo de um app
  - **assets/**: arquivos estáticos, como imagens, fontes, vídeos, svg, mp3 e etc
    - *img*
    - *fonts*
    - *svg*
- **routes**: arquivos de configurações com a organização dos possíveis caminhos e telas navegáveis dentro do app
- **stores**: essa pasta não é obrigatória, mas sempre quem app possui um controle de estado global, costumo armazenar nesta pasta
  - **views**: todas as telas da aplicação (compostas de diversos components, é claro)

Estrutura do Projeto

Por fim em relação a estrutura de um projeto, vale destacar uma grande diferença entre um projeto RN "puro" e um projeto baseado em Expo:



Como podemos ver as imagens acima, o projeto RN "puro" te possibilita o acesso direto a base de seu projeto em cada plataforma nativa, enquanto no caso do Expo não temos acesso (inicialmente) a isto, pois o Expo gerencia toda essa parte "avançada" para nós.

Estrutura do Projeto

Caso você precise criar um código específico que rode apenas em uma determinada plataforma, existem dois caminhos óbvios:

1) Criar um arquivo específico, basta fazê-lo ***.android.js*** *ou* ***.ios.js***
2) Utilizar a API ***Platform*** (https://reactnative.dev/docs/platform-specific-code#platform-module) para criar um código específico

Components e React

Padrão de código da *lib* que se estende ao RN

React *(lib)*

Todo ***framework/lib JavaScript*** moderno, possui alguma padronização para criação de componentes reutilizáveis ao longo de sua aplicação *(web ou mobile)*. Um botão que aparece em várias telas, uma caixa de texto imputável, listas, seletores, tudo que possa ser reutilizável deve ser concebido e organizado como um componente.

Em React temos dois tipos concretos de componentes: **(sub)class based component e functional components *(com ou sem hooks)***.

Além dessas estruturas é básico saber como se controla o estado de componentes em React e também dominar o uso de **JSX**.

Functional Components

Em React, componentes de função são os mais simples de serem escritos, contém apenas um método render e não possuem seu próprio estado.

```
function HelloMessage(props) {
    return <Text>Olá, {props.name}!</Text>;
}
```

Functional Components *(com estado + hooks)*

Com o uso de hooks podemos "incrementar" as funcionalidades de componentes funcionais e agregar também um controle de estado próprio para cada componente:

```
function Contador() {
    const [count, setCount] = React.useState(1);
    const incrementarCount = () => setCount(count + 1);

    return (
        <View>
            <Text>Contagem {count}</Text>
            <Button title="Incrementar" onPress={() => incrementarCount()} />
        </View>
    );
}
```

Não se esqueça de *importar*

Vale ressaltar que sempre antes de usarmos um componente em React Native, seja ele nativo (que pertence ao core do framework) ou de terceiros, será necessário usar o comando "***`import`***" para trazer o componente "a contexto de uso".

As partes mais básicas que foram herdadas da lib React, são importadas do próprio pacote:

```
import React, { useState } from "react";
```

Não se esqueça de *importar*

Vale ressaltar que sempre antes de usarmos um componente em React Native, seja ele nativo (que pertence ao core do framework) ou de terceiros, será necessário usar o comando "***`import`***" para trazer o componente "a contexto de uso".

Outras partes como componentes nativos são importadas da lib "react-native":

```
import { Text } from "react-native";
```

Não se esqueça de *importar*

Vale ressaltar que sempre antes de usarmos um componente em React Native, seja ele nativo (que pertence ao core do framework) ou de terceiros, será necessário usar o comando "***`import`***" para trazer o componente "a contexto de uso".

E se você estiver usando o Expo, pode aproveitar alguns pacotes exclusivos:

```
import * as Location from 'expo-location';
```

Components principais

Do *core* do framework

Componentes Básicos

Difícil de não precisar

View: O componente base mais fundamental do React Native, seria o equivalente as divs do HTML ou o LinearLayout do Android Nativo

Text: Componente para exibição de textos estáticos

Image: Componente para exibir imagens locais ou remotas

TextInput: Componente para digitação e formulários

ScrollView: LinearLayout com barra de rolagem

Button: Botão semi-estilizado

Switch: Aquela "chavezinha" de ligar e desligar

Componentes de Lista

FlatList: componente de alta performance para renderizar lista de itens

SectionList: igual o anterior porém com separação de cabeçalho/agrupamento

Outros Componentes

ActivityIndicator: Famoso "carregando" / "loading"

Alert: Alertas e confirmações em geral

Modal: Janelas flutuantes ou fullscreen

StatusBar: Componente que controla a barra de status do app

https://reactnative.dev/docs/components-and-apis

Estilização (CSS?)

StyleSheet.create

Mas como eu deixo esse botão vermelho?

Como já falado em nossas aulas, o "pessoal" por trás do RN sempre tentou "aproximar" a experiência do desenvolvedor (DX) com a WEB. E na parte dos "estilos" e personalização visual não poderia deixar de ser diferente.

A API do React / React Native está apta a interpretar estilos "conhecidos" de CSS, desde que os mesmos sejam escritos em um formato semelhante a um objeto JS / JSON.

Mas como eu deixo esse botão vermelho?

Por exemplo, para o CSS abaixo relacionado a WEB:

```
.container {
  display: flex;
  background-color: #fff;
  align-items: center;
  justify-content: center;
}
```

Mas como eu deixo esse botão vermelho?

Teríamos uma conversão direta para JSS desta forma:

```
container: {
  flex: 1,
  backgroundColor: "#fff",
  alignItems: "center",
  justifyContent: "center",
}
```

Mas como eu deixo esse botão vermelho?

E para isolarmos esse objeto dentro de uma "folha de estilos" usável dentro de uma view, iremos precisar da API ***StyleSheet*** do React Native:

```
const styles = StyleSheet.create({
  container: {
    flex: 1, backgroundColor:
    "#fff", alignItems:
    "center", justifyContent:
    "center",
  },
});
```

Mas como eu deixo esse botão vermelho?

Depois de definirmos nossa "classe" podemos usar da seguinte forma em um componente:

```
<View style={styles.container}>
  ...
</View>
```

A ideia é "ligarmos" direto com o objeto `styles`, e ao invés de usarmos o termo "class" da web, usamos o atributo ***style***.

Observações finais sobre estilização

Unidades de medidas sempre ocultas, nunca especifique *px, rem ou dp*, a plataforma saberá qual a melhor unidade de acordo com o OS a ser usada.

Aprenda a lógica de **flexbox** do CSS, será muito útil também em RN.

Após dominar o estilo básico do React Native, você poderá estudar **styled-components** (https://styled-components.com/docs/basics#react-native)

Frameworks visuais

Facilitando, será?

Frameworks visuais

Por padrão o React Native não têm e não possui nenhum componente de UI requintado para uso, ou seja o React Native vem "de fábrica" cru.

Mas existem alguns frameworks bacana no mercado para composição de UI:

https://callstack.github.io/react-native-paper

https://nativebase.io
https://react-native-training.github.io/react-native-elements
https://wix.github.io/react-native-ui-lib/

Frameworks visuais

Para demonstrações em sala de aula, iremos trabalhar com o ***React Native Paper***.

Para instalarmos o mesmo, se estivermos no react-native ou expo:

https://callstack.github.io/react-native-paper/docs/guides/getting-started

obrigado 🚀