

FIA/P GRADUAÇÃO

Hybrid Mobile App Development

Profs: [Andrey Masiero, Douglas Cabral, Gustavo Calixto]

#07 – Introdução do Desenvolvimento com React Native

Até agora



React Native

- Não gera um aplicativo web based
 - Não emula um browser no dispositivo
- Ele constrói um aplicativo nativo para Android e IOS com o mesmo código.
- Problema: Para IOS precisa de um sistema Apple para desenvolver.



Pré-requisitos

- NodeJS
- Python
- Java 8
- Android Studio + SDK com Emulador
- Xcode (apenas para mac)
- Watchman (apenas para mac)



FIAPgram

Vamos ser influencers na FIAP =D

Começando o projeto

- Vamos desenvolver o FIAPgram, clone do Instagram.
- Por que clone do Instagram?
 - É um projeto que aborda várias funcionalidades que permitem você desenvolver diversos outros aplicativos.
 - Teremos funcionalidades como login, chat, manipulação de imagens e muito mais.



Criando o projeto

- No terminal, utilizamos o comando
`npm -g install react-native-cli`
- Ele instalará o pacote do React Native para nós.

```
C:\Program Files\cmdr
```

```
λ npm -g install react-native-cli
```

```
C:\Users\andre\AppData\Roaming\npm\react-native -> C:\Users\andre\AppData\Roaming\npm\node_modules\react-native-cli\index.js  
+ react-native-cli@2.0.1
```

```
updated 4 packages in 13.559s
```

```
C:\Program Files\cmdr
```

```
λ
```

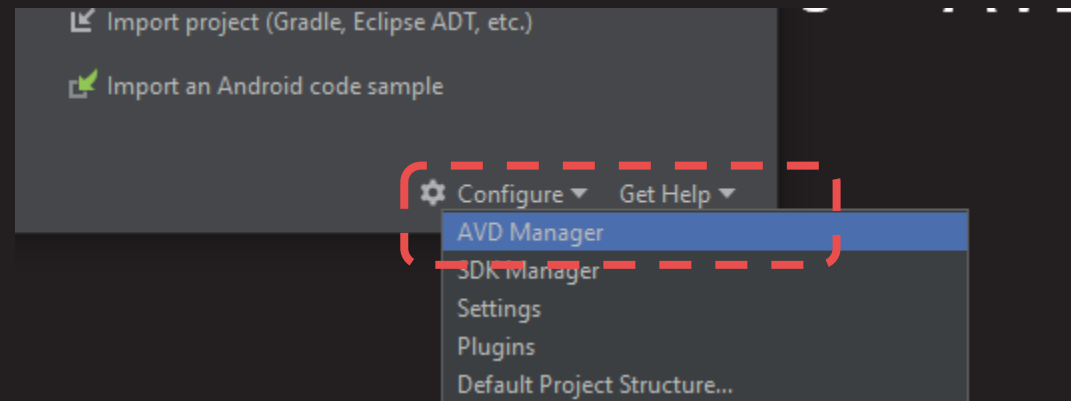

Criando o projeto

- Depois de instalado o pacote react-native-cli, utilizamos o seguinte comando para criar o projeto:
`react-native init <nome-projeto>`

```
C:\Users\andre\Desktop\fiapgram
λ react-native init FIAPgram
This will walk you through creating a new React Native project
in C:\Users\andre\Desktop\fiapgram\FIAPgram
'yarn' não é reconhecido como um comando interno
ou externo, um programa operável ou um arquivo em lotes.
Installing react-native...
Consider installing yarn to make this faster: https://yarnpkg.com
[.....] \ fetchMetadata: sill resolveWithNewModul
```

Executando o projeto

- Para o Android é necessário inicializar o emulador antes.
- Na tela inicial do Android Studio selecione Configure -> AVD Manager



- Clique no botão de play para iniciar o emulador.

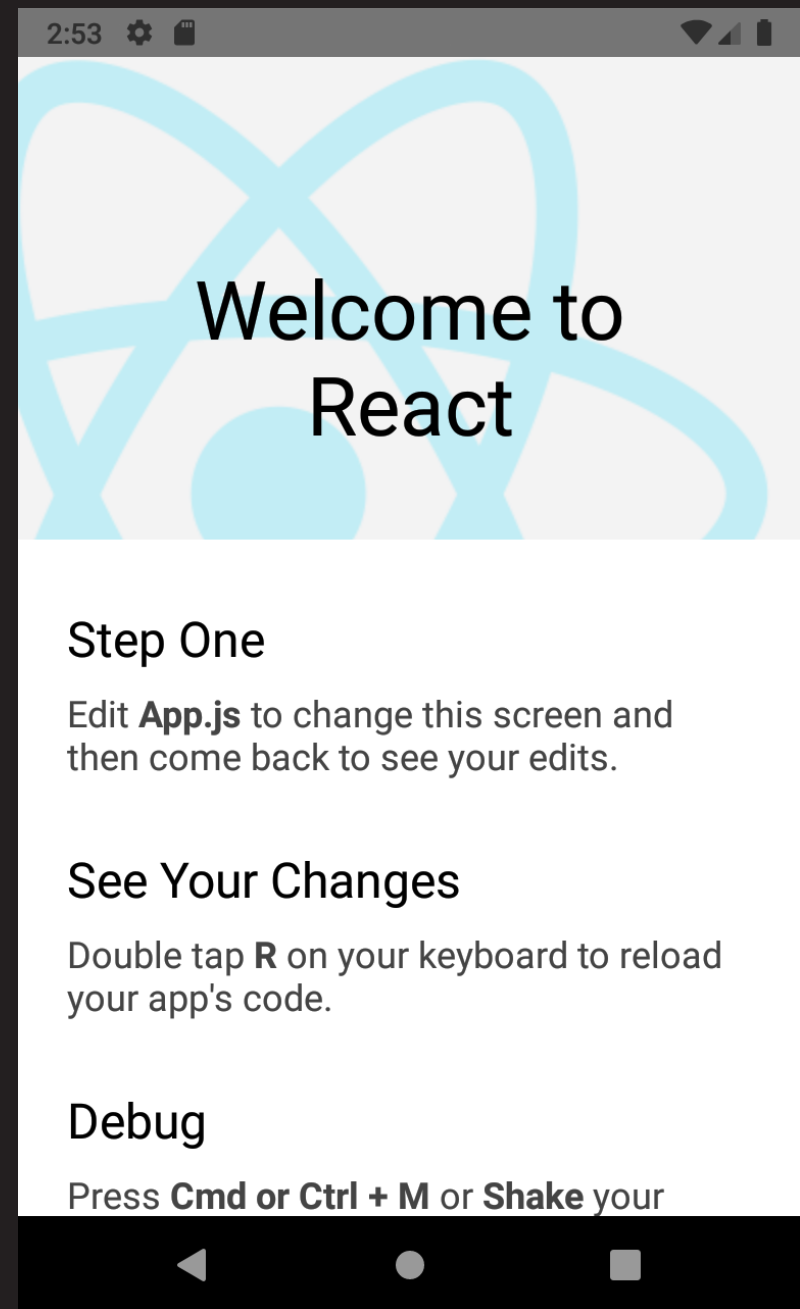
Executando o projeto

- Volte ao terminal, entre na pasta do projeto e execute o comando:

```
react-native run-android  
(ou run-ios)
```

- Resultado deve ser esse →

O projeto inicial pode ser encontrado no endereço:
<https://github.com/amasiero/fiapgram/tree/v0>



Entendendo o projeto

- O arquivo de entrada é o index.js
- Ele quem chama o arquivo com o código inicial e registra como saída ao sistema.

```
import {AppRegistry} from 'react-native';  
import App from './App';  
import {name as appName} from './app.json';  
  
AppRegistry.registerComponent(appName, () => App);
```

- Raramente uma mudança é feita nele.

Entendendo o projeto

- App.js

Importação dos componentes do React e React Native

Definição da classe App, ela deve ser filha da classe React.Component

Método render devolve o que será exibido na tela do aplicativo.

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

```
class App extends React.Component {  
  render() {  
    return (  
      <>  
      </>  
    );  
  }  
}
```

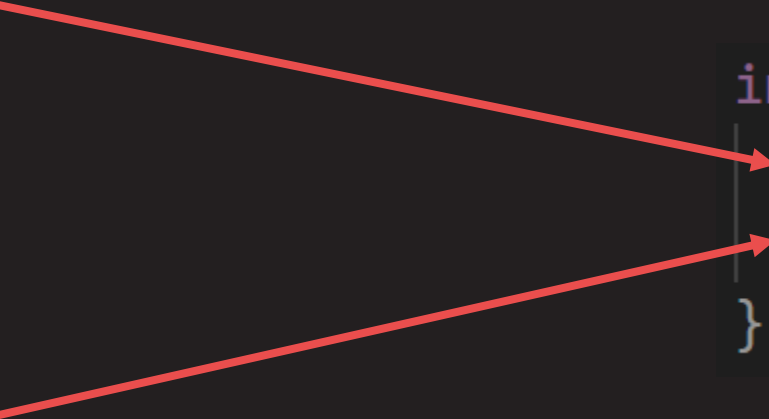
```
export default App;
```

Entendendo do projeto

View : é um container de componente, auxilia na organização.

Text : é um componente para exibição de texto no aplicativo. Funciona como uma label.

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



Hora de Codar

Entendendo o código e começando o modelo de nossa aplicação

Começando o FIAPgram

Vamos observar o feed do Instagram.
Podemos definir 4 partes nele:

Cabeçalho

Imagem / Foto

Legenda

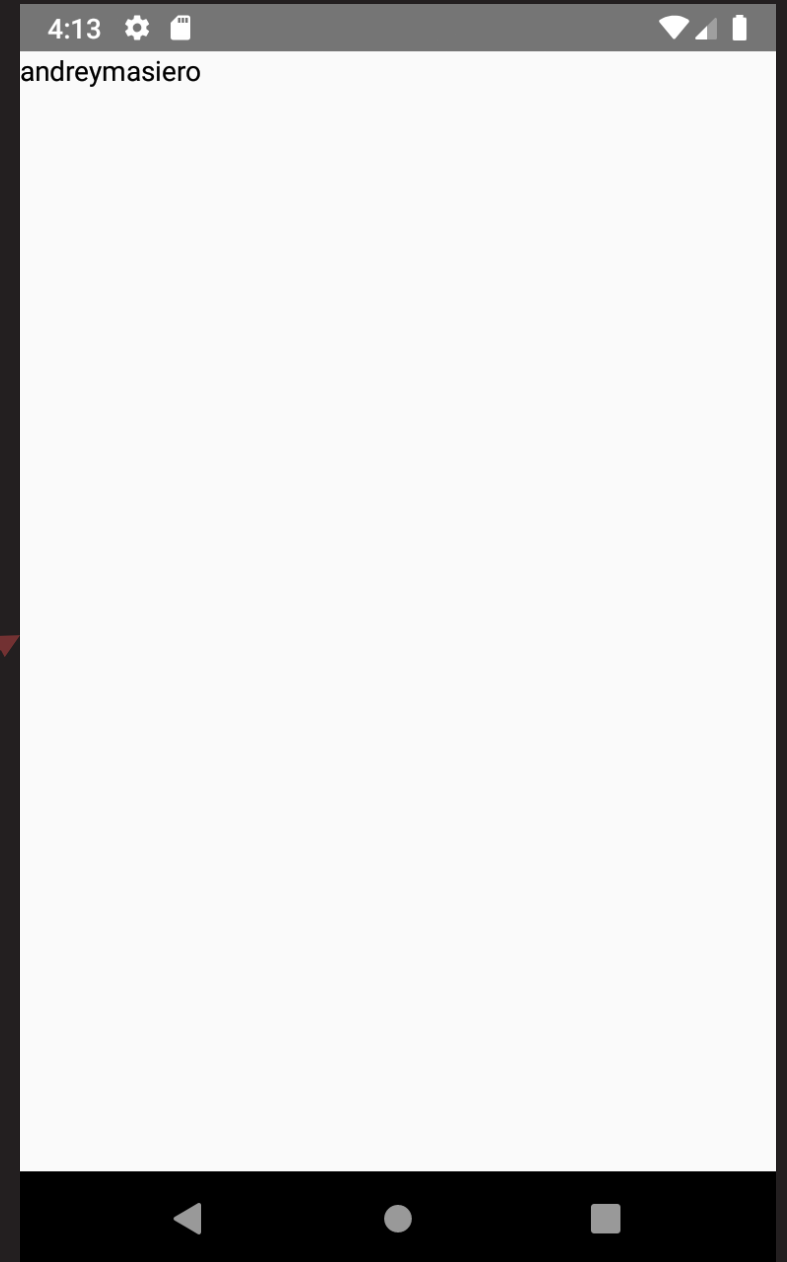
Comentários



Começando pelo Cabeçalho

A View é o container do cabeçalho. No Text, vamos colocar o nome do usuário. Fixo por enquanto.

```
render() {  
  return (  
    <>  
    <View>  
      <Text>andreymasiero</Text>  
    </View>  
  </>  
);  
}
```



Trabalhando com a imagem

Nesse momento, não importa muito a imagem que vamos colocar. Pode selecionar uma da internet mesmo como:

<https://i.imgur.com/ZrDq6UQ.jpg>

Importe o componente Image

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

Adicione dentro da View.

```
<View>  
  <Image  
    source={{uri : 'https://i.imgur.com/ZrDq6UQ.jpg'}}  
  />  
  <Text>andreymasiero</Text>  
</View>
```

Trabalhando com a imagem

Para aparecer a imagem, é necessário ajustar seu tamanho. Para isso, utilizamos a propriedade style, que recebe um objeto json com atributos como no arquivo CSS.

```
<Image  
  style={{width : 48, height : 48}}  
  source={{uri : 'https://i.imgur.com/ZrDq6UQ.jpg' }}  
>
```



Melhorando o estilo

Assim como no HTML e CSS, incluir estilo direto no componente é uma má prática.

Para nos auxiliar como isso o React Native possui um objeto que nos ajuda a criar o estilo. Ele é o StyleSheet.

Vamos importar ele:

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

Melhorando o estilo

Fora da classe, vamos declarar uma constante chamada de styles e usar o StyleSheet para criar nossa folha de estilos.

```
const styles = StyleSheet.create({  
  imgProfile : {  
    height : 48,  
    width : 48  
  }  
});
```

Agora, é só chamar no componente desejado. Nesse caso, a foto do cabeçalho.

```
<Image  
  style={styles.imgProfile}  
  source={{uri : 'https://i.imgur.com/ZrDq6UQ.jpg'}}  
>
```

Estilizando o cabeçalho

Vamos alinhar o nosso cabeçalho. Começando pelo container. Uma coisa interessante no React Native é que ele é baseado todo em flex, isso facilita o trabalho de alinhar os componentes.

Três classes foram criadas para deixar o header de acordo com o nosso aplicativo.

O componente Platform permite que verifiquemos qual o SO que está rodando e trocar a configuração do estilo.

```
const styles = StyleSheet.create({
  header : {
    flexDirection : 'row',
    alignItems: 'center',
    padding : 16
  },
  imgHeader : {
    height : 48,
    width : 48,
    borderRadius : 50,
    marginRight : 8
  },
  userHeader : {
    fontFamily : Platform.os === 'ios' ?
    'AvenirNext-Regular' : 'Roboto',
    fontSize : 16
  }
});
```

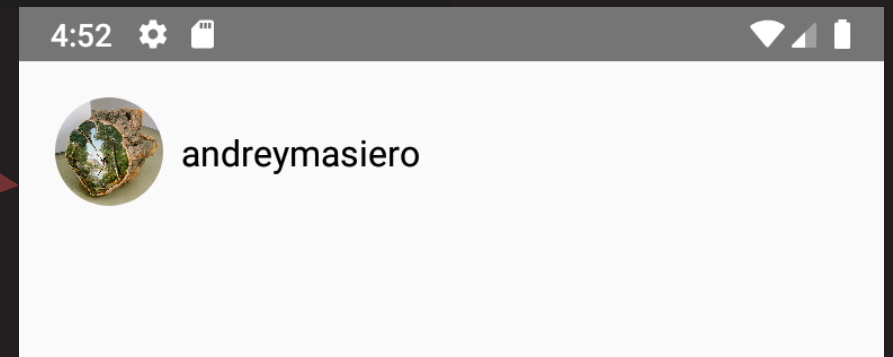
Estilizando o cabeçalho

Aplicamos eles nos componentes:

```
<View style={styles.header}>  
  <Image  
    style={styles.imgHeader}  
    source={{uri : 'https://i.imgur.com/ZrDq6UQ.jpg'}}  
  />  
  <Text style={styles.userHeader}>andreymasiero</Text>  
</View>
```

E conferimos o resultado.

Bem melhor né?



Inserindo a imagem do feed

Para ajustar a imagem da largura do dispositivo, podemos utilizar o componente Dimensions (não esqueça de importa-lo).

```
const width = Dimensions.get('screen').width
```

Agora podemos utilizar a constante no nosso estilo.

```
picture : {  
  width : width,  
  height : width  
}
```



```
<Image  
  style={styles.picture}  
  source={{uri : 'https://i.imgur.com/ZrDq6UQ.jpg'}}  
/>
```


Inserindo a imagem do feed



andreymasiero



Curtida e legenda

Para a parte de curtida, vamos utilizar o Font-Awesome. Ele prove diversos ícones, prontos para uso. A instalação dele deve ser feita no terminal, na pasta do projeto. Execute os seguintes comandos:

```
$ npm i --save react-native-svg  
$ npm i --save @fortawesome/fontawesome-svg-core  
$ npm i --save @fortawesome/free-solid-svg-icons  
$ npm i --save @fortawesome/react-native-fontawesome  
$ npm i --save @fortawesome/free-regular-svg-icons
```

Curtida e legenda

O próximo passo é importar os ícones no App.

```
import { FontAwesomeIcon } from '@fortawesome/react-native-fontawesome'  
import { faHeart as sHeart } from '@fortawesome/free-solid-svg-icons'  
import { faHeart as rHeart } from '@fortawesome/free-regular-svg-icons'
```

Como existem dois componentes com o mesmo nome (faHeart) devemos aplicar um apelido para cada um através da palavra reservada **as**.

Curtida e legenda

Vamos começar com o componente de curtidas.

```
<View style={styles.containerLikes}>
  <FontAwesomeIcon icon={false ? sHeart : rHeart}
    size={32}
    style={false ? styles.heart : {}}/>
  <Text style={styles.leftSpace} >2 curtidas</Text>
</View>
```

Aproveitamos e já colocamos os operadores ternários para exibir um ícone ou outro de acordo se a foto está ou não curtida. O mesmo foi aplicado na questão do estilo.

Curtida e legenda

O próximo componente é a legenda da imagem.

```
<Text style={styles.leftSpace} numberOfLines={3} >  
  <Text style={styles.userPost}>andreymasiero </Text>  
  Landscape painted on the surface of a cut log  
</Text>
```

O atributo `numberOfLines` determina o máximo de linhas que irão aparecer do texto. Depois disso, ele simplesmente corta o conteúdo e não o exibe.

Curtida e legenda

Vamos conferir os estilos criados nesse momento. E já conferir o resultado do App.

```
containerLikes : {  
  flexDirection : 'row',  
  alignItems : 'center',  
  padding : 8  
},  
heart : {  
  color : 'red'  
},  
leftSpace : {  
  paddingHorizontal : 8  
},  
userPost : {  
  fontWeight : 'bold'  
}
```



andreymasiero



2 curtidas

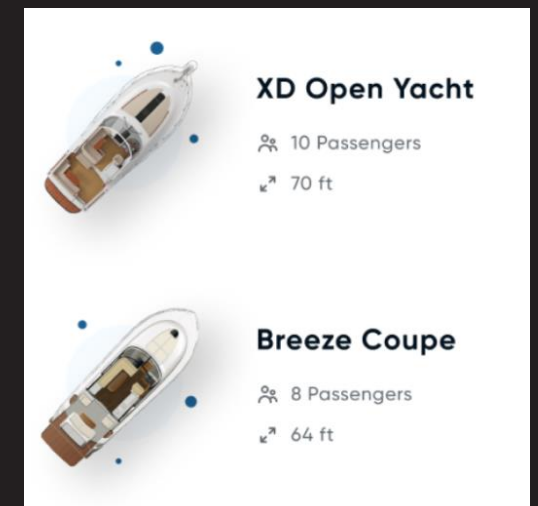
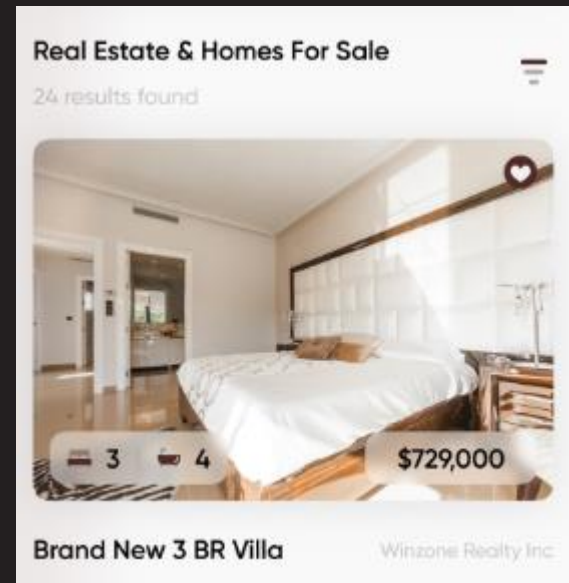
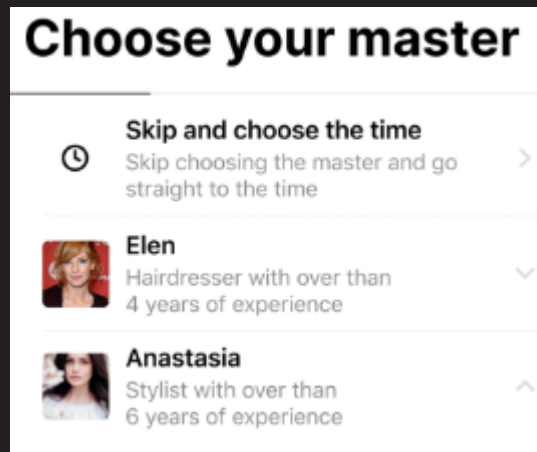
andreymasiero Landscape painted on the surface of a cut log

Exercícios

Hora do descanso.

Exercícios

- Replique os seguintes layouts:



Adaptações devem ser feitas.

Próximos Passos

O que veremos na próxima aula

Na próxima aula...

- Componentes Personalizados
- Aumentando o Feed
- Trabalhando com dados:
 - Props
 - State
 - componentDidMount

Copyright © 2020

Profs : [Andrey Masiero, Douglas Cabral, Gustavo Calixto]

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).