Paradigmas de Programação
React Native

Axios: App Contatos – 1ª Parte

Utilizando <FlatList/> e <ImageView>

Gil Eduardo de Andrade

APP: CONTATOS – FLATLIST

Descriação:

- O componente *<FlatList/>* permite renderizar, de modo dinâmico, um *array* de dados passado a sua propriedade (*props*) ***data***;
- O *array* é percorrido item a item, onde cada qual pode ser apresentado no formato de componente e personalizado via propriedade (*props*) ***style***;

APP: CONTATOS - IMAGEVIEW

Descrição:

- O componente *<ImageView>* permite que um conjunto de imagens (*array*) seja apresentado no formato *Dialog* (sobre a tela que está sendo renderizada no momento);
- O *array* de imagens é passado ao componente através da sua propriedade ***images*** ;

FlatList e ImageView na Prática

Contatos: App

www.gileduardo.com.br/ifpr/pp_rn/downloads/pp_rn_exapp09.zip

API EXTERNA – RANDOMUSER

Dados Externos *(App.js)*

https://randomuser.me/api/?nat=br&results=30


```
{
  - results: [
    - {
        gender: "female",
      - name: {
            title: "miss",
            first: "marialba",
            last: "teixeira"
        },
      - location: {
            street: "8936 rua castro alves ",
            city: "mesquita",
            state: "pernambuco",
            postcode: 80960,
          - coordinates: {
                latitude: "23.9231",
                longitude: "107.1157"
            },
          - timezone: {
                offset: "-3:00",
                description: "Brazil, Buenos Aires, Georgetown"
            }
        },
        email: "marialba.teixeira@example.com",
      - login: {
            uuid: "af05e38a-c36b-46b8-bb62-28dbed423a38",
            username: "bluecat560",
            password: "herbie",
```

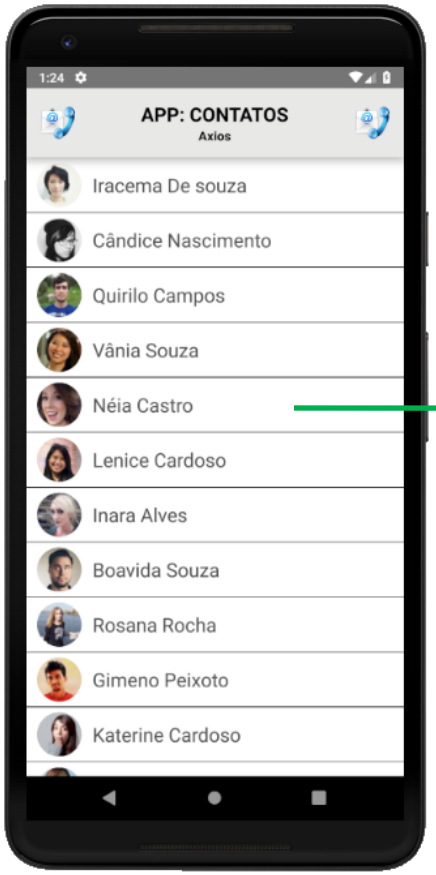

A API RandomUser disponibiliza/gera um conjunto de informações sobre usuários que pode ser utilizado como teste por desenvolvedores em suas aplicações.

Ao invocar a API é possível passar um conjunto de parâmetros que permite filtrar os dados que serão recebidos. No exemplo, estamos especificando a nacionalidade ***nat=br*** e a quantidade de usuário ***results=30*** que serão recebidos pela aplicação.

Os dados são recebidos em formato JSON, contendo informações como: *nome*, *sobrenome*, *endereço*, *foto*, *login*, *senha*, entre outros.

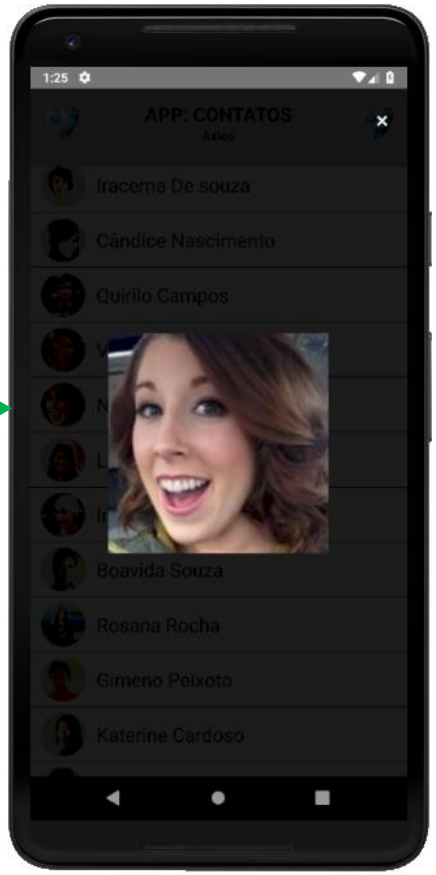
APP: CONTATOS

Layout *(App.js)*



Tela Inicial - <FlatList />

Ao selecionar um contato



Tela Inicial - <ImageView />

APP CONTATOS – AXIOS

Dados Externos *(App.js)*

```
constructor(props) {
    super(props);

    this.state = {
        contatos : [],
        carregando : false,
        error : false,
        images: [],
        visible: false,
        nome: '',
    };
}
```

```
axios
    .get('https://randomuser.me/api/?nat=br&results=30')
    .then(response => {
        const { results } = response.data;
        this.setState({
            contatos : results,
            carregando : false
        });
```

No construtor, da classe inicial (*App.js*) da aplicação, foram declarados, dentre outros atributos já utilizados na aplicação anteriores, dois arrays: (1) ***contatos[]***: que recebe as informações da API RandomUser; (2) ***images[]***: que recebe as imagens que serão renderizadas pelo *<ImageView>*. O atributo ***visible*** é utilizado para indicar quando o componente *<ImageView>* deve ficar visível ou não.

Os dados são recebidos da API, via cliente *axios*, e armazenados no ***state contatos***.

APP CONTATOS – IMAGEVIEW

Exibindo <ImageView/> *(App.js)*

```
onPress(pessoa) {

  const images = [{
      source: { uri: pessoa.picture.large },
      width: 240,
      height: 240,
  },];

  this.setState({
    images: images,
    visible: true,
  });
}
```

Quando um contato da lista é pressionado o método ***onPress()*** é invocado, recebendo como parâmetro o contato selecionado (os dados do contato – parâmetro ***pessoa***). Ao receber o parâmetro o método define um array contendo apenas a imagem do contato selecionado (***uri: pessoa.picture.large***) e define o tamanho com que a imagem deve ser exibida. Após isso os *states* ***images*** e ***visible*** (true) são atualizados/alterados para que o método render seja invocado e o *<ImageView/>* seja apresentado.

APP CONTATOS – IMAGEVIEW

Exibindo <ImageView/> *(App.js)*

```
render() {
  return (
    <View style={styles.container}>
      <HeaderBar />
      <View style={styles.list}>
        { this.renderList() }
      </View>
      <ImageView
        images={this.state.images}
        imageIndex={0}
        isVisible={this.state.visible}
      />
    </View>
  );
```

O método ***render()*** contém um método ***renderList()*** que verifique quem deve ser exibido (*<ActivityIndicator/>* ou Lista de Contatos) até o os dados da API serem carregados.

Além do método ***renderList()*** também já definido o componente <ImageView/>, contudo ele só é exibido quando sua propriedade (props) ***inVisible*** recebe o valor ***true***. Isso acorre quando um contato é selecionado, como descrito no slide anterior.

APP CONTATOS – FLATLIST

Componente ItemList *(ItemList.js)*

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

import Item from './Item';

const ItemList = props => {

    const { itens } = props;

    return (
        <FlatList
            style = { style.main }
            data = { itens }
            renderItem = { ({ item }) => (
                <Item pessoa={ item } onPress={(pessoa) => props.onPress(pessoa) } />
            )}
            keyExtractor = { item => item.login.uuid }
        />
    );
};
```

Para listagem dos contatos recebidos da API foi criado um componente chamado ***<ItemList>***. Ele recebe via ***props itens*** o ***array*** de contatos que deve ser exibido. Para isso ele utiliza o componente ***<FlatList/>*** agregado ao componente ***<Item/>*** também criado dentro da aplicação Contatos.

O componente ***<FlaIList/>*** recebe via propriedade (***props***) data o ***array itens*** que deve ser listado. Através da propriedade (***props***) ***renderItem,*** o ***<FlatList/>*** percorre todo o ***array***, e para cada elemento do mesmo gera um valor ***item***, que é passado para o componente ***<Item/>*** via propriedade (***props***) ***pessoa***. Além disso o componente ***<Item/>*** também recebe o método que deve ser invocado quando o contato da lista for selecionado, isso ocorre via propriedade (***props***) ***onPress***.

APP CONTATOS – FLATLIST

Componente Item *(Item.js)*

```
import UpperCase from '../util/UpperCase';

const Item = props => {

    return (
        <TouchableOpacity onPress={ (pessoa) => { props.onPress(props.pessoa) } }>
            <View style={style.view_item}>
                <Image style={style.avatar} source={{ uri : props.pessoa.picture.thumbnail }} />
                <Text style={style.text}>
                    {UpperCase(props.pessoa.name.first)} {UpperCase(props.pessoa.name.last)}
                </Text>
            </View>
        </TouchableOpacity>
    );
}
```

O componente *<Item/>* é envolvido por um componente *<TouchableOpacity/>*, o que torna-o *"clicável"*. Ele é composto, ainda, por um componente *<Image/>*, que apresenta a foto do contato, e por um componente *<Text/>*, que apresenta o primeiro e último nome do contato. Observe que um método *UpperCase()* foi criado dentro da pasta */util*, ele faz com a que a primeira letra de uma string seja colocada em caixa alta.

APP: CONTATOS – 1ª Parte

Exemplos Utilizados no Documento

Código-fonte do App Exemplo: Contatos

http://www.gileduardo.com.br/ifpr/pp_rn/downloads/pp_rn_exapp09zip

Exercício sobre o Conteúdo