

# React Native

Projeto de Interfaces de Dispositivos Móveis

**Projeto Albums (Parte 1)**

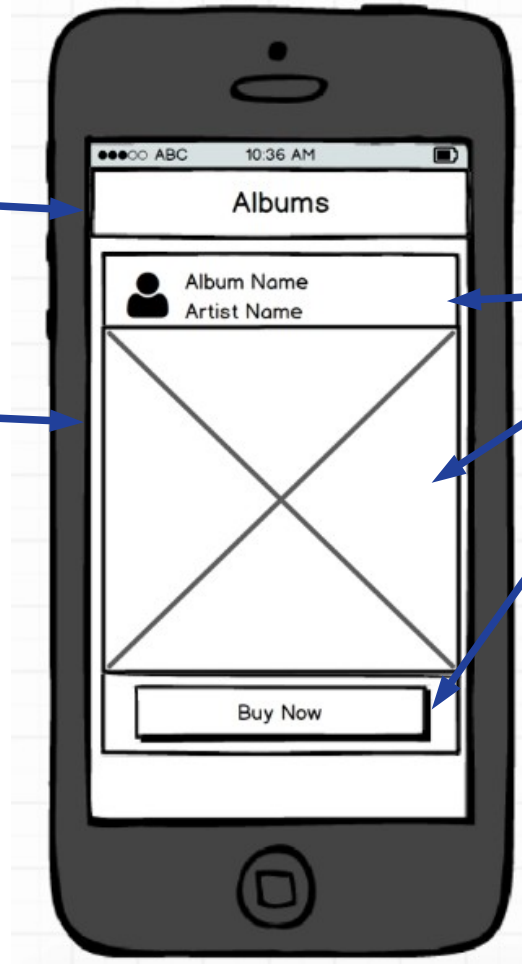
**Aula 03**

# Introdução

- **Projeto Albums**
  - Apresentação
  - Separação de componentes em arquivos
  - Folhas de estilo

**Header Component**

**Card Component**



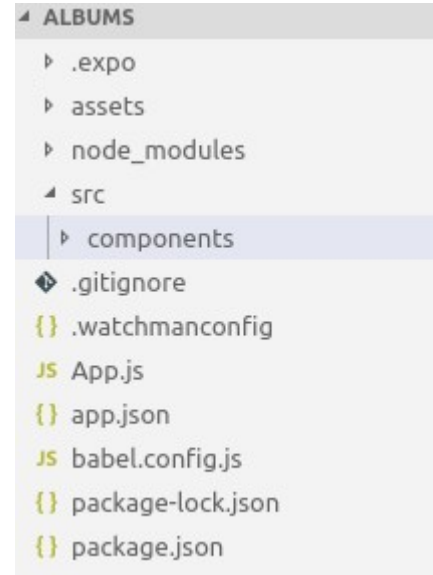
**Card Section**

# Crie a aplicação

- expo init albums
- cd albums
- npm install

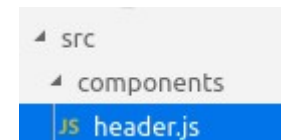
# Organizando as pastas

- crie a pastas src/components
  - é nela que ficarão os componentes da aplicação.



# Header

- Crie o arquivo header.js, dentro de src/components
- Por padrão, todos os nossos componentes irão ficar dentro de src/components.



# Header

- header.js

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

/*const Header = () => {
  return (
    <Text>Cabeçalho</Text>
  );
};*/

class Header extends Component {
  render() {
    return (
      <Text>Cabeçalho da Aplicação</Text>
    );
  }
}

export default Header;
```

**Pode-se usar tanto uma função, como uma classe para representar um componente.**

**No entanto, vamos ficar com a abordagem da classe.**

# Header

- Em App.js (principal)

```
import React, {Component} from 'react';
import { AppRegistry } from 'react-native';

import Header from './src/components/header';

export default class App extends Component{
  render(){
    return (
      <Header/>
    );
  }
}

AppRegistry.registerComponent('albums',()=>App);
```

O componente Header é chamado como uma tag JSX.





# Estilos (textStyle)

header.js

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

class Header extends Component {
  render() {
    return (
      <Text style={styles.textStyle}>Cabeçalho da Aplicação</Text>
    );
  }
}

const styles = StyleSheet.create({
  textStyle: {
    fontSize: 20
  }
})
```

# Estilos (viewStyle v.0)

header.js

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

class Header extends Component {
  render() {
    return (
      <View style={styles.viewStyle}>
        <Text style={styles.textStyle}>Cabeçalho da Aplicação</Text>
      </View>
    );
  }
}

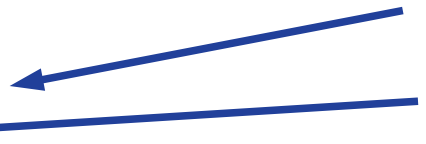
const styles = StyleSheet.create({
  viewStyle: {
    backgroundColor: '#F2F2F2'
  },
  textStyle: {
    fontSize: 20
  }
});
```

# Estilos (viewStyle v.1)

```
viewStyle: {  
  justifyContent: 'center',  
  alignItems: 'center',  
  height: 60,  
  paddingTop: 15,  
  backgroundColor: '#F2F2F2',  
  shadowColor: '#000',  
  shadowOffset: { width: 0, height: 2 },  
  shadowOpacity: 0.2  
},
```

alinha verticalmente

alinha horizontalmente



# Props – Generalizando o Header

- header.js

```
class Header extends Component {  
  render() {  
    return (  
      <View style={styles.viewStyle}>  
        <Text style={styles.textStyle}>{this.props.title}</Text>  
      </View>  
    );  
  }  
}
```

# Props – Generalizando o Header

- App.js

```
export default class App extends Component{  
  render(){  
    return (  
      <Header title='Albums'/>  
    );  
  }  
}
```

# Listando Albums

- Vamos usar a API free:
  - [http://rallycoding.herokuapp.com/api/music\\_albums](http://rallycoding.herokuapp.com/api/music_albums)

# Listando Albums

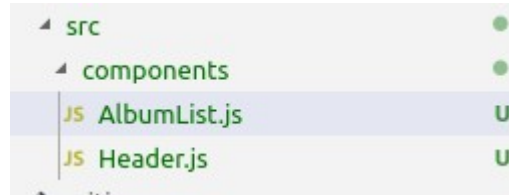
- Criando o componente AlbumList.js

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

```
class AlbumList extends Component{
```

```
  render(){  
    return (  
      <View>  
        <Text>Album List!</Text>  
      </View>  
    );  
  }  
}
```

```
export default AlbumList;
```



# Listando Albums

- Usando AlbumList em App.js

```
import AlbumList from './src/components/AlbumList';

export default class App extends Component{
  render(){
    return (
      <View>
        <Header title='Albums' />
        <AlbumList />
      </View>
    );
  }
}
```

**Atenção!** Não se pode retornar apenas mais de uma linha de JSX. Coloque os “irmãos” <Header> e <AlbumList> dentro de uma tag pai <View>.




# Listando Albums

```
constructor(props){  
  super(props);  
  this.state = {albums:[]};  
}
```

Fazendo o **fetch** em  
AlbumList.js

```
componentWillMount(){  
  return fetch('http://rallycoding.herokuapp.com/api/music_albums')  
    .then((response)=>response.json())  
    .then(  
      (responseJson) =>{  
        this.setState(  
          {albums:responseJson}  
        );  
      }  
    );  
}
```



A chamada do `setState` faz com que a tele seja renderizada novamente.

# Listando Albums

AlbumList.js

- função renderAlbums e render

```
renderAlbums(){  
    return this.state.albums.map(album=><Text>{album.title}</Text>);  
}  
  
render(){  
    return (  
        <View>  
            {this.renderAlbums()}  
        </View>  
    );  
}
```

# Listando Albums

AlbumList.js

- Adicionando uma key (para evitar o warning)

```
renderAlbums(){  
  return this.state.albums.map(album=>  
    <Text key={album.title}>{album.title}</Text>  
  );  
}
```

# Detalhes do Album

- Crie o arquivo AlbumDetail.js em src/components
- Iremos passar, através do props, o objeto “album” inteiro para o componente AlbumDetail

# Detalhes do Album

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

AlbumDetail.js

```
class AlbumDetail extends Component {
```

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

 **inicialize o props no construtor**

```
  render() {  
    return (  
      <View>  
        <Text>{this.props.album.title}</Text>  
      </View>  
    );  
  }  
}
```

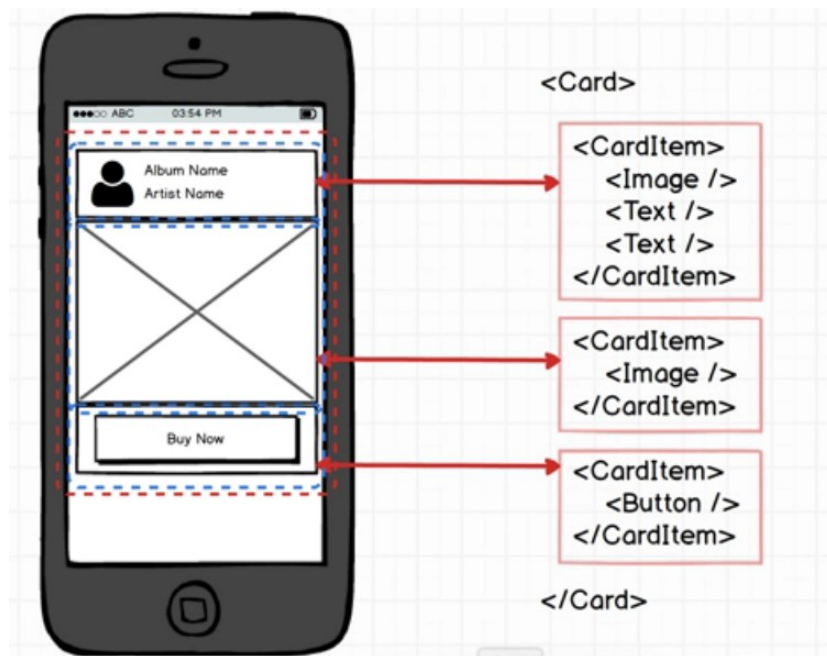
 **acesse o props.album**

```
export default AlbumDetail;
```

# Detalhes do Album

- Para não escrevermos todo o estilo do AlbumDetail de forma “amarrada”, iremos criar mais dois componentes que serão usado por AlbumDetail.
  - Card
  - CardItem

# Detalhes do Album



<https://www.udemy.com/the-complete-react-native-and-redux-course/>

# Card e CardItem

- Crie os componentes em src/components
  - Card.js
  - CardItem.js



# Card com estilo

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

class Card extends Component{

  render(){
    return (
      <View style={styles.containerStyle}>

      </View>
    );
  }
}
```

```
const styles = StyleSheet.create({
  containerStyle: {
    borderWidth: 1,
    borderRadius: 2,
    borderColor: '#ddd',
    borderBottomWidth: 0,

    shadowColor: '#000',
    shadowOffset: {width:0, height:2},
    shadowOpacity: 0.1,
    shadowRadius: 2,

    elevation: 1,

    marginLeft: 5,
    marginRight: 5,
    marginTop:10
  }
});

export default Card;
```

# Passando Componente

- O componente **Card** é responsável em renderizar os detalhes de cada álbum em vários **CardItem**
- Por enquanto, como não temos ainda o **CardItem**, vamos passar o **AlbumDetail** como props para **Card**, usando o **Card** como uma tag em **AlbumDetail**.



# Passando Componente

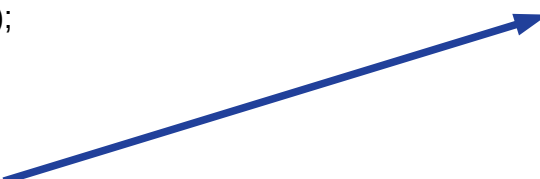
- Em AlbumDetail

```
import Card from './Card';

class AlbumDetail extends Component {

  constructor(props) {
    super(props);
  }

  render() {
    return (
      <Card>
        <Text>{this.props.album.title}</Text>
      </Card>
    );
  }
}
```



Não se usa mais o <View> e sim o <Card>. Agora <Card> é responsável em renderizar tudo o que for passado para ele. No caso, apenas esse <Text>.

# Passando Componente

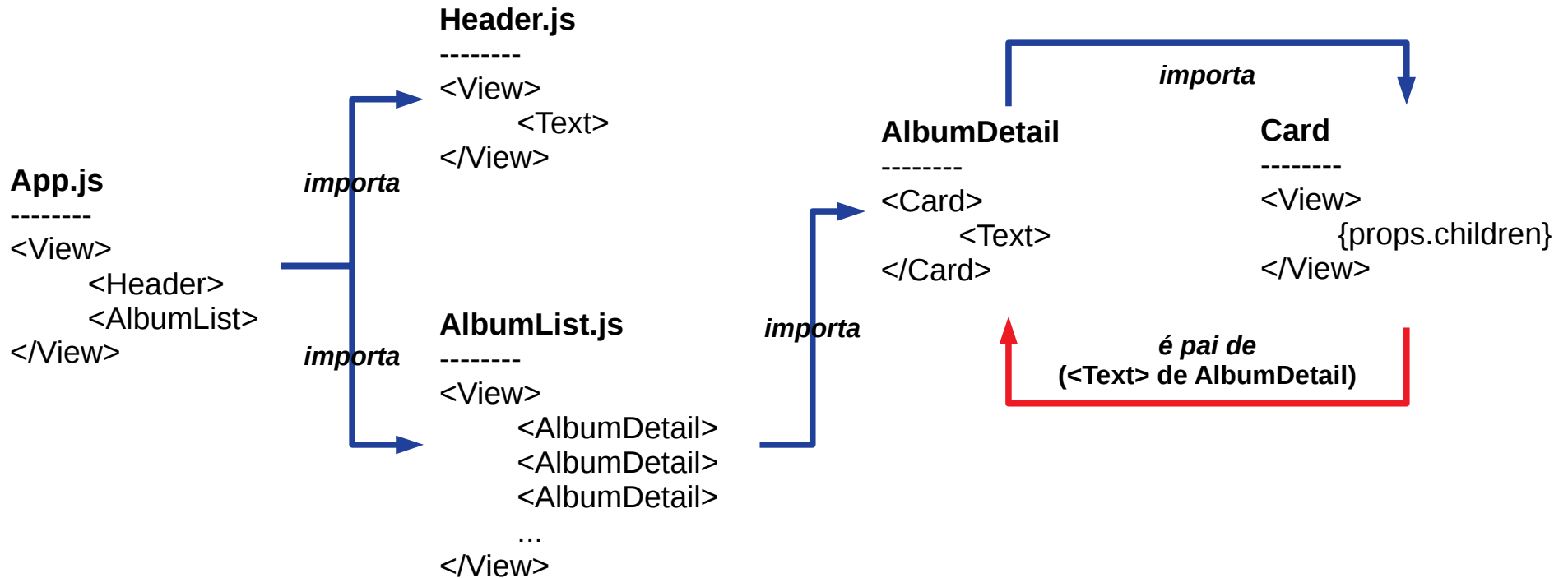
- Em Card

```
class Card extends Component{  
  constructor(props){  
    super(props);  
  }  
  
  render(){  
    return (  
      <View style={styles.containerStyle}>  
        {this.props.children}  
      </View>  
    );  
  }  
}
```

**{this.props.children}** “chama” o componente filho de Card, ou seja, todo JSX que for passado dentro d tag <Card></Card> vai ser renderizado aqui.

Até agora, de acordo com o slide anterior, será o componente AlbumDetail.

# Visão geral (até agora)



# CardItem

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

```
class CardItem extends Component{

  constructor(props){
    super(props);
  }

  render(){
    return (
      <View style={style.containerStyle}>
        {this.props.children}
      </View>
    );
  }
}
```

```
const style = StyleSheet.create({
  containerStyle: {
    borderBottomWidth: 1,
    padding: 5,
    backgroundColor: '#fff',
    borderColor: '#ddd',
    position: 'relative',
    justifyContent: 'flex-start',
    flexDirection: 'row',
  }
});

export default CardItem;
```

# AlbumDetail

```
import Card from './Card';
import CardItem from './CardItem';

class AlbumDetail extends Component {
  constructor(props) {
    super(props);
  }

  render() {
    return (
      <Card>
        <CardItem>
          <Text>{this.props.album.title}</Text>
        </CardItem>
      </Card>
    );
  }
}
```

# Visão Geral

