

React Native

Projeto de Interfaces de Dispositivos Móveis

**Projeto CRUD com
Firebase**

Aula 06

Introdução

- **Projeto CRUD - Parte 2**
 - Construção de uma nova aplicação
 - Reuso de componentes da aplicação passada
 - Data Storage com o Firebase

Introdução

- **Criando um novo projeto**
 - Crie um novo projeto no react-native:
 - `react-native init aula_crud`
 - `npm install react-navigation –save`
 - `npm install react-navigation-stack –save`
 - `npm install react-native-gesture-handler --save`
 - `npm install firebase`
 - Copie e cole a pasta “commons” do projeto passado.
 - Coloque App.js dentro de src e modifique o arquivo index.js

Componentes

- em src/components:
 - LivroAddScreen.js
 - LivroListarScreen.js
 - LivroMenuScreen.js
 - LivroEditarScreen.js
 - Routes.js

```
import { createAppContainer } from 'react-navigation';
import { createStackNavigator } from 'react-navigation-stack';
```

```
import LivroMenuScreen from './LivroMenuScreen';
import LivroAddScreen from './LivroAddScreen';
import LivroListarScreen from './LivroListarScreen';
```

```
const MainStack = createStackNavigator(
  {
    LivroMenuScreen,
    LivroAddScreen,
    LivroListarScreen
  },
  {
    initialRouteName: 'LivroMenuScreen',
    defaultNavigationOptions: {
      headerStyle: {
        backgroundColor: '#2c2c2c',
      },
      headerTintColor: '#fff',
      headerTitleStyle: {
        fontWeight: 'bold',
      }
    }
  }
)
```

```
const Routes = createAppContainer(MainStack);
```

```
export default Routes;
```

Routes.js

Inicialmente, criando o arquivo de rotas, para poder navegar entre as páginas da nossa aplicação.

```

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

import { Cartao, CartaoItem, MeuBotao, Header } from './commons'

export default class LivroMenuScreen extends Component {

  static navigationOptions = {
    title: "Menu"
  };

  render() {
    return (
      <Cartao>
        <Header titulo="Sistema de Livros" />
        <CartaoItem>
          <MeuBotao onPress={() => this.props.navigation.navigate('LivroListarScreen')}>
            Listar Livros
          </MeuBotao>
        </CartaoItem>
        <CartaoItem>
          <MeuBotao onPress={() => this.props.navigation.navigate('LivroAddScreen')}>
            Adicionar Livro
          </MeuBotao>
        </CartaoItem>
      </Cartao>
    );
  }
}

```

LivroMenuScreen.js

A tela de menu terá apenas dois botões, já fazendo uso dos nossos componentes reusáveis.

Listar

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

import { MeuSpinner, Cartao, CartaoItem, MeuLabelText, Header, MeuBotao } from './commons'

import * as firebase from 'firebase';
import 'firebase/firestore';

export default class LivroListarScreen extends Component {

  static navigationOptions = {
    title: "Listar Livros"
  };

  constructor(props) {
    super(props);
    this.unsubscribe = null;
    this.ref = firebase.firestore().collection('livros');
    this.state = { loading: true, livros: [] };
  }
}
```

LivroListarScreen.js

Inicialize o “state” com as variáveis **loading** (algo está carregando) e **livros**, onde ficarão os livros carregados.

No que também é criado uma **ref** para a conexão com a coleção “livros”.

A variável **unsubscribe** é explicada mais a frente.


```

componentDidMount() {
  this.unsubscribe = this.ref.onSnapshot(this.alimentarLivros.bind(this)); //onSnapshot
}

alimentarLivros(query) {
  let livros = [];
  query.forEach((doc) => {
    const { titulo, preco, autor, imagem } = doc.data();
    livros.push({
      key: doc.id,
      titulo,
      autor,
      preco,
      imagem
    });
  }); //forEach
  this.setState({ loading: false, livros });
}

```

LivroListarScreen.js

Em `componentDidMount`, criamos um “**snapshot**” da nossa conexão com ‘**livros**’, feita no construtor. Note que usamos o **this.ref**. Resumidamente, um **snapshot** “escuta” mudanças na base firebase e avisa ao cliente (app móvel). Isso é feito através da função `onSnapshot`.

Já na função `onSnapshot`, passamos como parâmetro outra função, que irá alimentar a variável `livros` local, através de um objeto do firebase chamado “`query`”.

```

renderAlert(key) {
  Alert.alert(
    'Excluir livro',
    'Tem certeza?',
    [
      { text: 'Sim', onPress: () => this.excluirLivro(key) },
      { text: 'Cancelar', onPress: () => console.log('Cancelar Pressed') },
    ],
    { cancelable: false },
  );
}

```

```

excluirLivro(key){
  this.setState({loading:true});
  firebase.firestore().collection('livros').doc(key).delete()
  .then(()=>{
    this.setState({loading:false});
  })
  .catch(()=>{
    this.setState({loading:false});
  });
}

```

LivroListarScreen.js

Em renderAlert, mostramos um alert para ter certeza sobre a operação de excluir.

A função excluirLivro acessa o firebase chamando o método “delete”. Como é uma promessa, trabalhamos com o then, em caso de sucesso e o catch, em caso de erro.

```
renderConteudo() {  
  if (this.state.loading) {  
    return <Cartao><CartaoItem><MeuSpinner /></CartaoItem></Cartao>  
  }  
  return <FlatList  
    data={this.state.livros}  
    renderItem={({ item }) =>  
      <Cartao>  
  
        <CartaoItem>  
          <MeuLabelText label="Título" texto={item.titulo} />  
        </CartaoItem>  
        <CartaoItem>  
          <MeuLabelText label="Autor" texto={item.autor} />  
        </CartaoItem>  
        <CartaoItem>  
          <MeuLabelText label="Preço" texto={item.preco} />  
        </CartaoItem>  
      </Cartao>  
    } />  
}
```

LivroListarScreen.js

Função renderConteudo decide se irá renderizar a página ou um spinner.

MeuLabelText é um novo componente nosso criado para essa aplicação.

```

    <CartaoItem>
      <MeuBotao
        onPress={() => this.props.navigation.navigate("LivroEditarScreen", { livro: item })}
      >
        Editar
      </MeuBotao>
      <MeuBotao
        onPress={()=>this.renderAlert(item.key)}
      >
        Excluir
      </MeuBotao>
    </CartaoItem>
  </Cartao>
}
/>
}

render() {
  return (
    <View>
      <Cartao><Header titulo="Sistema de Livros" /></Cartao>
      {this.renderConteudo()}
    </View>
  );
}
}

```

LivroListarScreen.js

Continuação da renderConteudo. Nenhuma novidade...

Note que o link para chamar a página de edição passa como parâmetro o livro a ser editado.

```

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

class MeuLabelText extends Component{
  render(){
    return(
      <View style={estilos.containerEstilo}>
        <Text style={estilos.labelEstilo}>{this.props.label}</Text>
        <Text style={estilos.textoEstilo}>{this.props.texto}</Text>
      </View>
    );
  }
}

```

MeuLabelText.js

Praticamente o mesmo código de
MeuInputText.

```

const estilos = StyleSheet.create({
  containerEstilo:{
    flex:1,
    flexDirection:"row",
    alignItems:"center",
    height:40
  },
  labelEstilo:{
    fontSize:18,
    paddingLeft:10,
    flex:1,
    fontWeight:"bold"
  },
  textoEstilo:{
    color:'#000',
    paddingRight:5,
    paddingLeft:5,
    fontSize:18,
    lineHeight:23,
    flex:4
  }
});

export {MeuLabelText}

```

Editor

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

```
import { MeuSpinner, Cartao, CartaoItem, MeuInput, Header, MeuBotao } from './commons'
```

```
import * as firebase from 'firebase';
import 'firebase/firestore';
```

```
export default class LivroEditarScreen extends Component {
```

```
  static navigationOptions = {
    title: "Editar Livro"
  };
```

```
  constructor(props) {
    super(props);
    const livro = this.props.navigation.getParam("livro", null);
    this.state = { loading: false, titulo: livro.titulo, autor: livro.autor, preco: livro.preco, key: livro.key }
  }
```

LivroEditarScreen.js

Recebe em seu construtor o livro passado como parâmetro pela página de listar. Depois, inicializa o state.

```

updateLivro(){
  this.setState({loading:true});
  firebase.firestore().collection('livros').doc(this.state.key)
    .set({
      titulo: this.state.titulo,
      autor: this.state.autor,
      preco: this.state.preco
    })
    .then(()=>{
      this.setState({loading:false});
    })
    .catch(()=>{
      this.setState({loading:false});
    });
}

renderBotao(){
  if(this.state.loading){
    return <MeuSpinner/>
  }
  return <MeuBotao onPress={()=>this.updateLivro()}>Atualizar</MeuBotao>
}

```

LivroEditarScreen.js

Lê os dados de state e chama o método “set”, para realizar um update. Uma promessa.

renderBotao decide se renderiza o botão ou um spinner.


```

render() {
  return (
    <View>
      <Cartao>
        <Header titulo="Sistema de Livros" />
        <CartaoItem>
          <MeuInput label="Título" value={this.state.titulo} onChangeText={({titulo) => this.setState({ titulo })} />
        </CartaoItem>
        <CartaoItem>
          <MeuInput label="Autor" value={this.state.autor} onChangeText={({autor) => this.setState({ autor })} />
        </CartaoItem>
        <CartaoItem>
          <MeuInput label="Preço" value={this.state.preco + ""} onChangeText={({preco) => this.setState({ preco })} />
        </CartaoItem>
        <CartaoItem>
          {this.renderBotao()}
        </CartaoItem>
      </Cartao>
    </View>
  );
}
}

```

LivroEditarScreen.js

Lê os dados e coloca no state, usando o setState. Um pequeno problema no preço para o value inicial.

Adicionar

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

import {Cartao,Cartaoltem,MeuSpinner,MeuInput, MeuBotao, Header}
from './commons';
```

```
import * as firebase from 'firebase';
import 'firebase/firestore';
```

```
export default class LivroAddScreen extends Component{

  static navigationOptions = {
    title: "Adicionar Livro"
  };

  constructor(props){
    super(props);
    this.state = {loading:false,titulo:"",autor:"",preco:0}
  }
```

LivroAddScreen.js

Muito parecido com editar, só que agora iremos incluir um novo documento.

```
adicionarLivro(){
  this.setState({loading:true});
  firebase.firestore().collection('livros').add(
    {
      autor:this.state.autor,
      titulo:this.state.titulo,
      preco:this.state.preco,
      imagem:"imagem.png"
    }
  )
  .then(()=>{
    this.setState({loading:false})
    this.props.navigation.navigate("LivroListarScreen");
  })
  .catch(()=>{
    this.setState({loading:false})
  })
}
```

```

renderBotao(){
  if(this.state.loading){
    return <MeuSpinner/>
  }
  return <MeuBotao onPress={()=>this.adicionarLivro()}>Adicionar</MeuBotao>
}

render() {
  return (
    <View>
      <Cartao>
        <Header titulo="Sistema de Livros" />
        <CartaoItem>
          <MeuInput label="Título" placeholder="As Tranças do Rei Careca " onChangeText={({titulo) => this.setState({ titulo })} />
        </CartaoItem>
        <CartaoItem>
          <MeuInput label="Autor" placeholder="Fulano de Tal" onChangeText={({autor) => this.setState({ autor })} />
        </CartaoItem>
        <CartaoItem>
          <MeuInput label="Preço" placeholder="0.00" onChangeText={({preco) => this.setState({ preco })} />
        </CartaoItem>
        <CartaoItem>
          {this.renderBotao()}
        </CartaoItem>
      </Cartao>
    </View>
  );
}
}

```

Bugs: Setting a timer for a long period of time



SohamToraskar commented on 22 Apr • edited ▼



To sort this out you need to hard code the value, increase the value of the variable `MAX_TIMER_DURATION_MS`. Here are the steps:

Go to `node_modules/react-native/Libraries/Core/Timer/JSTimers.js`

Look for the variable `MAX_TIMER_DURATION_MS`

Change `60 * 1000` to `10000 * 1000`

Save the changes and re-build your app.

This worked for me.

<https://stackoverflow.com/a/46678121>



10



1



2

Recuperar

```

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

import { Cartao, CartaoItem, MeuBotao, Header, MeuSpinner } from './commons'

import * as firebase from 'firebase';
import 'firebase/firestore';

export default class LivroRecuperarScreen extends Component {

  static navigationOptions = {
    title: "Recuperar Livro"
  };

  constructor(props){
    super(props);
    this.state = {loading:true,livros_id:[],loading_pressed:false, key_pressed:"null"};
  }

  componentDidMount(){
    firebase.firestore().collection("livros").onSnapshot(
      (query)=>{
        livros_id = [];
        query.forEach((doc)=>{
          livros_id.push({
            key:doc.id
          })
        });
        this.setState({livros_id,loading:false})
      }
    );
  }
}

```

```

recuperarLivro(key){
  this.setState({loading_pressed:true});
  firebase.firestore().collection('livros').doc(key)
    .get()
    .then((doc)=>{
      if(doc.exists){
        alert("Titulo: \t"+doc.data().titulo+
              "\nAutor: \t"+doc.data().autor);
      }
      this.setState({loading_pressed:false});
    })
    .catch()=>{
      this.setState({loading_pressed:false});
    })
}

renderBotao(key){
  if(key!==this.state.key_pressed){
    if(this.state.loading_pressed){
      return <CartaoItem><MeuSpinner/></CartaoItem>
    }
  }
  return (
    <MeuBotao
      onPress={()=>{
        this.setState({key_pressed:key})
        this.recuperarLivro(key);
      }}
    >
      Recuperar Livro
    </MeuBotao>
  );
}

```



```

renderConteudo(){
  if(this.state.loading){
    return <CartaoItem><MeuSpinner/></CartaoItem>
  }
  return(
    <FlatList
      data={this.state.livros_id}
      renderItem={({item})=>
        <Cartao>
          <CartaoItem>
            <Text>{item.key}</Text>
          </CartaoItem>
          <CartaoItem>
            {this.renderBotao(item.key)}
          </CartaoItem>
        </Cartao>
      }
    />
  );
}

render(){
  return(
    <Cartao>
      <Header titulo="Recuperar Livro"/>
      {this.renderConteudo()}
    </Cartao>
  );
}
}

```