

# **Desenvolvimento de Software para WEB**

Conectando a aplicação CRUD com o Firebase

# O Firebase

# Firebase

- “Firebase is a mobile and web application development platform developed by Firebase, Inc. in 2011, then acquired by Google in 2014. As of March 2020, the Firebase platform has 19 products, which are used by more than 1.5 million apps.”

– *Wikipedia*

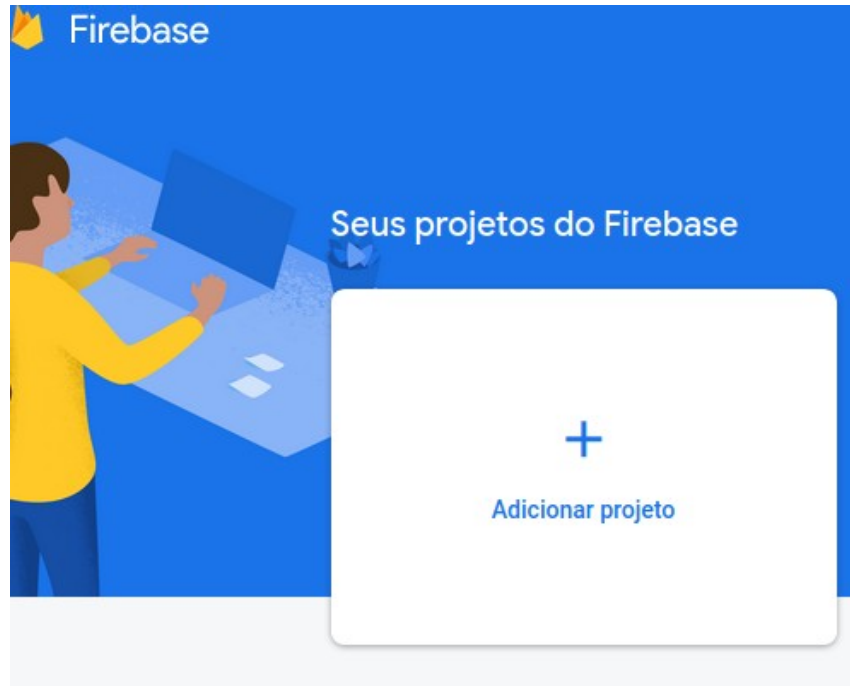
# Firebase

- Acesse o site:
  - <https://console.firebase.google.com/u/0/?hl=pt-br>
  - (Atenção, é necessário ter uma conta no Google!)



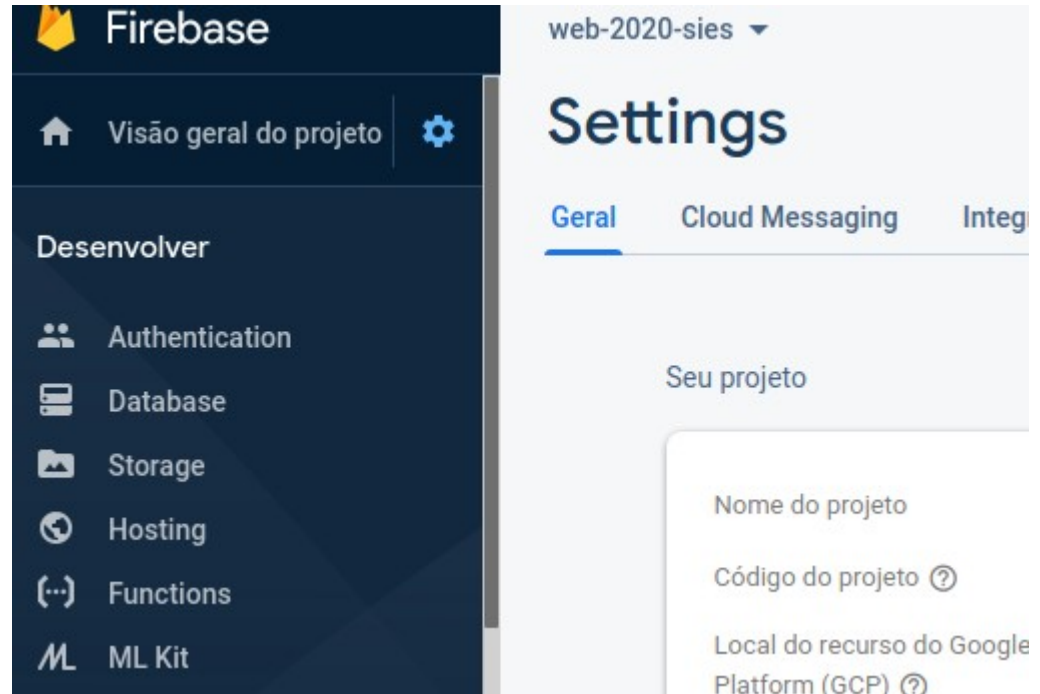
# Firebase

- Você deverá criar um novo projeto.



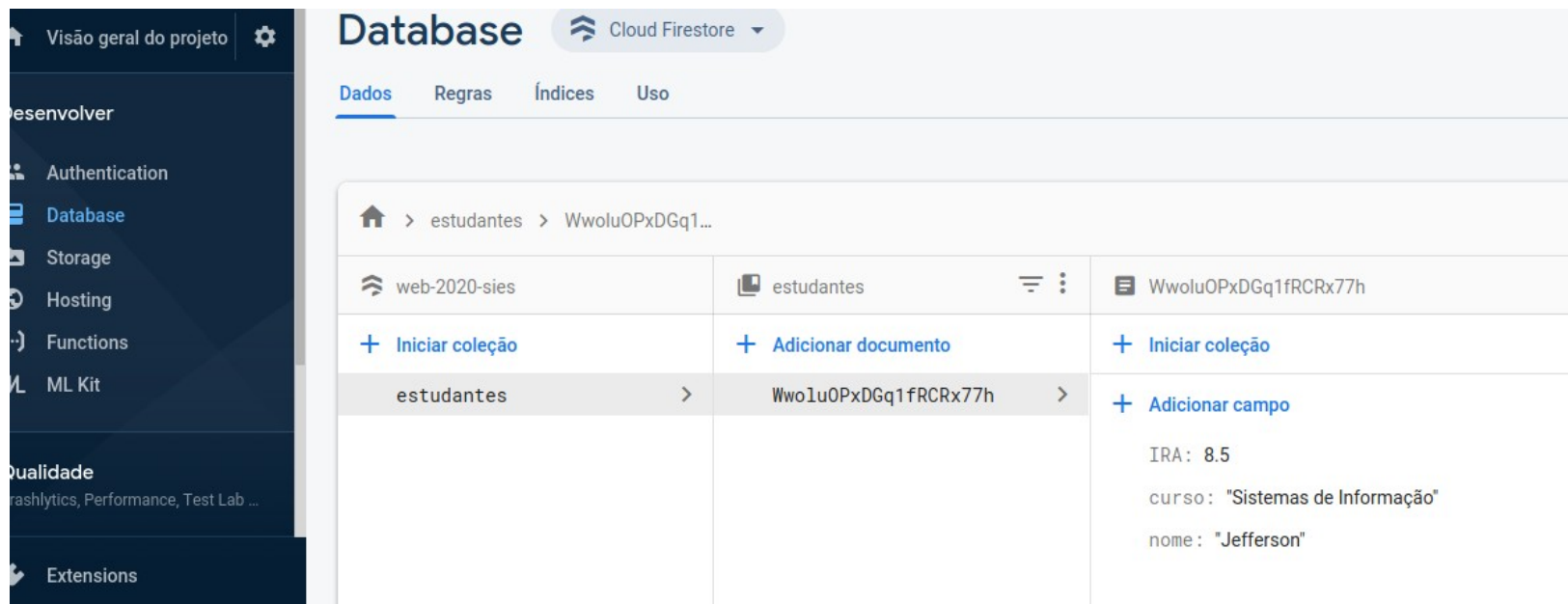
# Firebase

- Ao configurar todo o projeto (deixe as configurações padrão), você será apresentado a tela de console:



# Firestore

- Clique em “Database” e crie uma nova coleção que irá representar os estudantes. Chame ela de “estudantes”. Alimente os dados com o primeiro estudante. Não se preocupe com o id, ele é gerado **automaticamente**.



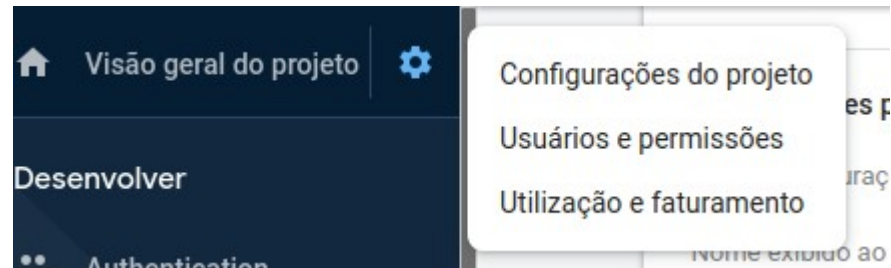


# CRUD

- Pronto, o Firebase já está minimamente configurado. Agora vamos partir para a projeto CRUD em React, o qual irá se comunicar com o firebase, através de uma **chave única** (objeto Json).
- Para facilitar, copia e cole o projeto “crud” em React que você já deve (ou deveria) ter implementado com a comunicação com express/mongo.
  - [https://github.com/jeffersoncarvalho/WEB\\_2020-1/tree/master/IMPLEMENTACOES/aula-es-si/aula03-crud](https://github.com/jeffersoncarvalho/WEB_2020-1/tree/master/IMPLEMENTACOES/aula-es-si/aula03-crud)
- Vamos trabalhar em cima de código antigo, modificando a comunicação com o **express**, para a comunicação com o **firebase**.

# CRUD

- Instale o firebase no seu projeto:
  - **npm install firebase --save**
- Em App.js, será criada a comunicação com Firebase (via contexto, como veremos adiante). Antes, você deve criar/consultar a chave no console do firebase. Acesse “Configurações de projeto:”



# CRUD

- Procure a chave (**atenção, por questões de segurança, não compartilhe!**), ou a gere.
- Eu já tinha adicionado um aplicativo para acessar o firebase. Caso você **não** tenha feito isso, terá de passar pelo processo de criação de aplicativo, o qual você deverá escolher qual o tipo de aplicativo que você está implementando irá acessar o firebase. No nosso caso, um **aplicativo web**.

# CRUD

- Copie a chave apresentada em **firebaseConfig**:

## Firestore SDK snippet

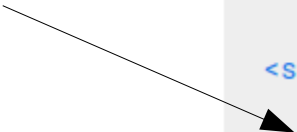
☒ CDN  ☐ Configuração 

Copie e cole esses scripts na parte inferior da tag <body>, mas antes de usar qualquer serviço do Firebase:

```
<!-- The core Firebase JS SDK is always required and must be listed
<script src="https://www.gstatic.com/firebasejs/7.14.4/firebase-app.

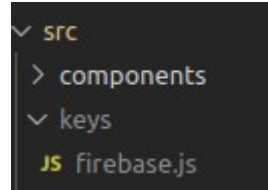
<!-- TODO: Add SDKs for Firebase products that you want to use
https://firebase.google.com/docs/web/setup#available-libraries

<script>
  // Your web app's Firebase configuration
  var firebaseConfig = {
```



# CRUD - Configuração

- Na pasta src do seu projeto, cria a pasta “**keys**” e dentro desta pasta o arquivo **firebase.js**



- Depois, procure a o arquivo .gitignore do seu projeto (**esse passo é importante pois irá previnir o upload da sua chave do google em um projeto público do github!**) e adicione:

```
# dependencies
/src/keys
/node_modules
/.pnp
.pnp.js
```

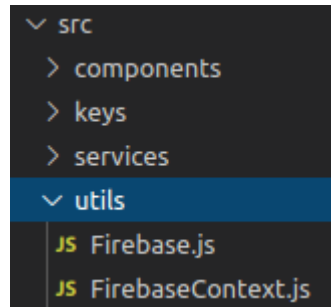
# CRUD - Configuração

- O arquivo **firebase.js** irá armazenar a chave (em key, copie a **sua** chave):

```
var key = {  
  apiKey: "Ô louco",  
  authDomain: "web-2020-sies.firebaseio.com",  
  databaseURL: "https://web-2020-sies.firebaseio.com",  
  projectId: "web-2020-sies",  
  storageBucket: "web-2020-sies.appspot.com",  
  messagingSenderId: "Ô bafulê!",  
  appId: "99% anjo mas aquele 1%..."  
}  
export default key
```

# CRUD - Configuração

- A forma como nossa aplicação se comunica com o firebase usará um padrão implementado por um contexto.
- O contexto irá garantir que teremos apenas uma única instância do firebase durante toda a aplicação, evitando assim múltiplas conexões sem necessidade.
- Em **src**, crie a pasta **utils**, e os arquivos **Firestore.js** e **FirestoreContext.js**:



# CRUD - Configuração

- O arquivo Firebase.js


```
import firebase from 'firebase/app'
import 'firebase/firestore'

import firebase_key from '../keys/firebase'

export default class Firebase{
  constructor(){
    firebase.initializeApp(firebase_key)
  }

  getFirestore(){
    return firebase.firestore()
  }
}
```

*O arquivo Firebase.js irá acessar o firebase.js dentro da pasta keys*





# CRUD - Configuração

- O arquivo `FirebaseContext.js`

```
import React from 'react';  
  
const FirebaseContext = React.createContext(null);  
  
export default FirebaseContext;
```

- O arquivo cria um objeto de contexto através da API do React. Inicialmente o contexto está nulo, mas iremos iniciá-lo adiante com um objeto de conexão com o firebase.
- Esse contexto é então compartilhado com toda a aplicação.

# CRUD - Configuração

- Em **index.js**, iremos disponibilizar (**Provider**) o contexto junto com uma única instância de conexão com o firebase para toda a aplicação (todos os componentes).

...

```
import Firebase from './utils/Firebase'
```

```
import FirebaseContext from './utils/FirebaseContext';
```

```
ReactDOM.render(  
  <FirebaseContext.Provider value={new Firebase()}>  
    <BrowserRouter>  
      <App />  
    </BrowserRouter>  
  </FirebaseContext.Provider>,  
  document.getElementById('root')  
)  
;
```

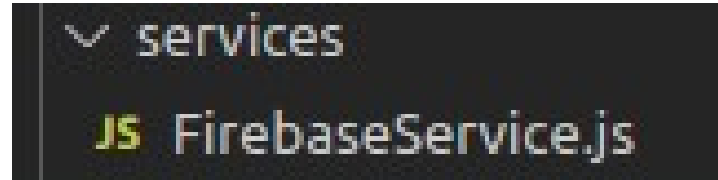
...

# CRUD - Operações

- As operações do CRUD são: Create, Read, Update e Delete.
  - Obs.: Read inclui Listar e Recuperar
  - Obs.: Update é o nosso Edit
- Para que o contexto seja consumido (**Consumer**) pelos componentes filhos de **index.js**, faz-se necessário que eles importem o arquivo que criamos de contexto, `FirestoreContext`.
- Em cada um dos componentes do nosso projeto CRUD, iremos envolver o componente original com um componente pai o qual irá passar para o filho uma props com a conexão com o firebase.

# CRUD - Services

- A camada de serviços “esconde” da visão (os componentes que renderizam JSX) os detalhes de conexão com a base de dados (no nosso caso, o Firebase).
- Crie a pasta **services**, e dentro dela o arquivo **FirestoreService.js**
  - Obs.: o ideal é criar uma arquivo service, para cada uma das entidades do seu sistema (Ex.: EstudanteService, ProfessorService, CursoService, etc.)
  - No nosso caso, como temos apenas a entidade estudante, vamos criar apenas um arquivo de services.



# CRUD - Services

```
export default class FirebaseService {
```

```
  static list = (firestore, callback) => {  
    let ref = firestore.collection('estudantes')  
    ref.onSnapshot((query) => {  
      let estudantes = []  
  
      query.forEach((doc) => {  
        const { nome, curso, IRA } = doc.data()  
        estudantes.push({  
          _id: doc.id,  
          nome,  
          curso,  
          IRA  
        })//push  
      })//forEach  
      callback(estudantes)  
      //this._isMounted && this.setState({ estudantes: estudantes, loading: false })  
    })  
  }  
}
```

*Toda função de um service, recebe um firestore e uma função callback, a qual “retorna” algo para quem chamou.*

# CRUD - Services

```
static delete = (firestore, callback, id) => {  
  
  firestore.collection('estudantes').doc(id).delete()  
    .then(  
      () => {  
        //console.log(`${nome} apagado.`)  
        callback('ok')  
      }  
    )  
    .catch(  
      (error) => {  
        //console.log(error)  
        callback('nok')  
      }  
    )  
}
```

*No caso do delete, também passamos o id do documento a ser apagado.*

# CRUD - Services

```
static create = (firestore, callback, estudante) => {  
  firestore.collection('estudantes').add(  
    {  
      nome: estudante.nome,  
      curso: estudante.curso,  
      IRA: estudante.IRA  
    }  
  )  
  .then(  
    () => {  
      callback('ok')  
    }  
  )  
  .catch(  
    (error) => {  
      callback('nok')  
    }  
  )  
}
```

*No create, passamos também o objeto “estudante”, o qual será gravado no firebase.*

# CRUD - Services

```
static retrieve = (firestore, callback, id) => {  
  
  firestore.collection('estudantes').doc(id).get()  
    .then((doc) => {  
      const estudante = {  
        nome: doc.data().nome,  
        curso: doc.data().curso,  
        IRA: doc.data().IRA  
      }  
      callback(estudante)  
    })  
    .catch(error => callback(null))  
  
}
```

*No caso do retrieve, também passamos o id do documento a ser apagado.*



# CRUD - Services

```
static edit = (firestore, callback, id, estudante) => {
```

```
  firestore.collection('estudantes').doc(id).set({  
    nome: estudante.nome,  
    curso: estudante.curso,  
    IRA: estudante.IRA  
  })  
  .then(() => {  
    callback('ok')  
  })  
  .catch(() => {  
    callback('ok')  
  });
```

```
}
```

```
}
```

*No caso do edit, também passamos o id e o objeto “estudante” modificado.*

# CRUD – Listar

- Em **List.jsx** criamos uma componente simples, através de uma função, chamado ListPage (vamos manter o padrão “Page” na frente do nome original)
- Esse componente “Page” consome (**Consumer**) o objeto que faz a conexão com o firebase, armazenado dentro de FirebaseContext

# CRUD – Listar #0

```
import FirebaseContext from '../utils/FirebaseContext'
import FirebaseService from '../services/FirebaseService'

const ListPage = () => (
  <FirebaseContext.Consumer>
    {firebase => <List firebase={firebase} />}
  </FirebaseContext.Consumer>
)

class List extends Component {

  constructor(props) {
    super(props)

    //firebase
    this._isMounted = false

    this.state = { estudantes: [], loading: false }
    this.apagarElementoPorId = this.apagarElementoPorId.bind(this)
  }
}
```

# CRUD – Listar #1

```
componentDidMount() {  
  this._isMounted = true  
  this.setState({ loading: true })  
  
  FirebaseService.list(this.props.firebase.getFirestore(),  
    (estudantes) => {  
      this._isMounted && this.setState({ estudantes: estudantes, loading: false })  
    })  
}
```

*O firestore é passado como  
parâmetro para o  
FirebaseServices.*

# CRUD – Listar #2

```
componentWillUnmount() {  
  this._isMounted = false  
}
```

```
apagarElementoPorId(id) {  
  let tempEstudantes = this.state.estudantes  
  for (let i = 0; i < tempEstudantes.length; i++) {  
    if (tempEstudantes[i]._id === id) {  
      tempEstudantes.splice(i, 1)  
    }  
  }  
  this._isMounted && this.setState({ estudantes: tempEstudantes })  
}
```

# CRUD – Listar #3

```
montarTabela() {  
  if (!this.state.estudantes) return  
  if (this.state.loading) return this.loadingSpinner()  
  return this.state.estudantes.map(  
    (est, i) => {  
      return <TableRow estudante={est}  
        key={i}  
        apagarElementoPorId={this.apagarElementoPorId}  
        firebase={this.props.firebase} />  
    }  
  )  
}
```

*Passando a instância do  
firebase pro TableRow*

# CRUD – Listar #4

```
loadingSpinner() {  
  return (  
    <tr style={{ backgroundColor: '#fff' }}>  
      <td colSpan='6'>  
        <div className="text-center">  
          <div className="spinner-border ml-auto"  
            role="status"  
            aria-hidden="true"  
            style={{ width: '3rem', height: '3rem' }}></div><br />  
          <strong>Loading...</strong>  
        </div>  
      </td>  
    </tr>  
  )  
}
```

# CRUD – Listar #5

```
render() {  
  return (  
    <div style={{ marginTop: 10 }}>  
      <h3>Listar Estudantes</h3>  
      <table className="table table-striped" style={{ marginTop: 20 }}>  
        <thead>  
          <tr>  
            <th>ID</th>  
            <th>Nome</th>  
            <th>Curso</th>  
            <th>IRA</th>  
            <th colSpan="2"></th>  
          </tr>  
        </thead>  
        <tbody>  
          {this.montarTabela()}  
        </tbody>  
      </table>  
    </div>  
  )  
}  
}  
export default ListPage
```



# CRUD – Listar (TableRow)

```
import React, { Component } from 'react'  
import { Link } from 'react-router-dom'  
  
import FirebaseService from '../services/FirebaseService'
```

```
export default class TableRow extends Component {
```

```
  constructor(props) {  
    super(props)  
    this.apagar = this.apagar.bind(this)  
  }
```

```
  apagar(id, nome) {  
    let res = window.confirm(`Deseja apagar ${nome}?`)  
    if (res) {  
      FirebaseService.delete(this.props.firebase.getFirestore(),  
        (mensagem) => {  
          console.log(mensagem)  
        }, id)  
    }  
  }  
}
```

*Usando o firebase passado  
pelo props.*

# CRUD – Listar (TableRow)

```
render() {  
  return (  
    <tr>  
      <td>  
        {this.props.estudante._id}  
      </td>  
      <td>  
        {this.props.estudante.nome}  
      </td>  
      <td>  
        {this.props.estudante.curso}  
      </td>  
      <td>  
        {this.props.estudante.IRA}  
      </td>  
      <td style={{ textAlign: "center" }}>  
        <Link to={"/edit/" + this.props.estudante._id} className="btn btn-primary">Editar</Link>  
      </td>  
      <td style={{ textAlign: "center" }}>  
        <button onClick={() => this.apagar(this.props.estudante._id, this.props.estudante.nome)}  
          className="btn btn-danger">  
          Apagar  
        </button>  
      </td>  
    </tr>  
  )  
}
```

# CRUD – Listar

CRUD Home Create List

## Projeto CRUD

### Listar Estudantes

ID	Nome	Curso	IRA		
WwoluOPxDGq1fRCRx77h	Jefferson	Sistemas de Informação	8.5	<a href="#">Editar</a>	<a href="#">Apagar</a>

# CRUD - Criar

```
import React, { Component } from 'react'

import FirebaseContext from '../utils/FirebaseContext'
import FirebaseService from '../services/FirebaseService'

const CreatePage = () => (
  <FirebaseContext.Consumer>
    {firebase => <Create firebase={firebase} />}
  </FirebaseContext.Consumer>
)

class Create extends Component {

  constructor(props) {
    super(props)

    this.state = { nome: "", curso: "", IRA: "" }

    this.setNome = this.setNome.bind(this)
    this.setCurso = this.setCurso.bind(this)
    this.setIRA = this.setIRA.bind(this)
    this.onSubmit = this.onSubmit.bind(this)
  }
}
```

# CRUD - Criar

```
setNome(e) {  
  this.setState({ nome: e.target.value })  
}  
  
setCurso(e) {  
  this.setState({ curso: e.target.value })  
}  
  
setIRA(e) {  
  this.setState({ IRA: e.target.value })  
}  
  
onSubmit(e) {  
  e.preventDefault()  
  const estudante = {  
    nome: this.state.nome,  
    curso: this.state.curso,  
    IRA: this.state.IRA  
  }  
  FirestoreService.create(this.props.firebase.getFirestore(),  
    (mensagem) => {  
      console.log(mensagem)  
    },  
    estudante)  
  this.setState({ nome: "", curso: "", IRA: "" })  
}
```

```

render() {
  return (
    <div style={{ marginTop: 10 }}>
      <h3>Criar Estudante</h3>
      <form onSubmit={this.onSubmit}>

        <div className="form-group">
          <label>Nome: </label>
          <input type="text" className="form-control"
            value={this.state.nome} onChange={this.setNome} />
        </div>
        <div className="form-group">
          <label>Curso: </label>
          <input type="text" className="form-control"
            value={this.state.curso} onChange={this.setCurso} />
        </div>
        <div className="form-group">
          <label>IRA: </label>
          <input type="text" className="form-control"
            value={this.state.IRA} onChange={this.setIRA} />
        </div>

        <div className="form-group">
          <input type="submit" value="Criar Estudante" className="btn btn-primary" />
        </div>
      </form>
    </div>
  )
}
export default CreatePage

```

# CRUD - Criar

# CRUD - Criar

CRUD Home **Create** List

## Criar Estudante

Nome:

Curso:

IRA:

Criar Estudante

# CRUD - Edit

```
import React, { Component } from 'react'
```

```
import FirebaseContext from '../utils/FirebaseContext'
```

```
import FirebaseService from '../services/FirebaseService'
```

```
const EditPage = (props) => (  
  <FirebaseContext.Consumer>  
    {firebase => <Edit firebase={firebase} id={props.match.params.id} />}  
  </FirebaseContext.Consumer>  
)
```

```
class Edit extends Component {
```

```
  constructor(props) {
```

```
    super(props)
```

```
    this.state = { nome: "", curso: "", IRA: "" }
```

```
    this.setNome = this.setNome.bind(this)
```

```
    this.setCurso = this.setCurso.bind(this)
```

```
    this.setIRA = this.setIRA.bind(this)
```

```
    this.onSubmit = this.onSubmit.bind(this)
```

```
  }
```



# CRUD - Edit

```
componentDidMount() {  
  FirebaseService.retrieve(this.props.firebase.getFirestore(),  
    (estudante) => {  
      if (estudante)  
        this.setState({  
          nome: estudante.nome,  
          curso: estudante.curso,  
          IRA: estudante.IRA  
        })  
    },  
    this.props.id  
  )  
}  
  
setNome(e) {  
  this.setState({ nome: e.target.value })  
}  
  
setCurso(e) {  
  this.setState({ curso: e.target.value })  
}  
  
setIRA(e) {  
  this.setState({ IRA: e.target.value })  
}
```

# CRUD - Editar

```
onSubmit(e) {  
  e.preventDefault()  
  FirestoreService.edit(  
    this.props.firebase.getFirestore(),  
    (mensagem) => {  
      console.log(mensagem)  
    },  
    this.props.id,  
    {  
      nome: this.state.nome,  
      curso: this.state.curso,  
      IRA: this.state.IRA  
    }  
  )  
}
```

# CRUD - Editar

```
render() {  
  return (  
    <div style={{ marginTop: 10 }}>  
      <h3>Editar Estudante</h3>  
      <form onSubmit={this.onSubmit}>  
  
        <div className="form-group">  
          <label>Nome: </label>  
          <input type="text" className="form-control"  
            value={this.state.nome} onChange={this.setNome} />  
        </div>  
        <div className="form-group">  
          <label>Curso: </label>  
          <input type="text" className="form-control"  
            value={this.state.curso} onChange={this.setCurso} />  
        </div>  
        <div className="form-group">  
          <label>IRA: </label>  
          <input type="text" className="form-control"  
            value={this.state.IRA} onChange={this.setIRA} />  
        </div>  
  
        <div className="form-group">  
          <input type="submit" value="Editar Estudante" className="btn btn-primary" />  
        </div>  
      </form>  
    </div>  
  )  
}
```

```
export default EditPage
```

# CRUD - Editar

[CRUD](#) [Home](#) [Create](#) [List](#)

## Editar Estudante

Nome:

Curso:

IRA:

Editar Estudante