

Desenvolvimento de Software para WEB

Projeto CRUD

Projeto CRUD

Introdução

- A ideia por trás deste projeto é criar uma aplicação simples onde será possível:
 - Criar uma entidade (CREATE)
 - Editar uma entidade (EDIT)
 - Listar as entidades criadas (LIST)
 - Apagar uma entidades criada (DELETE)
 - Mostrar uma entidade criada (RETRIEVE)

Introdução

- As tecnologias usadas nesse projeto serão:
 - React
 - Node
 - Express
 - Mongo (Futuramente o Firebase)
- MERN Stack: **M**ongo, **E**xpresse, **R**ead e **N**ode.

Parte 1

A Interface e Navegação

Criação do Projeto

- Execute o comando:
 - **create-react-app crud**
- Entre na pasta do projeto e instale as seguintes bibliotecas:
 - **cd crud**
 - **npm install bootstrap --save**
 - **npm install react-router-dom --save**

Criação do Projeto

- Crie o sistemas de pastas, dentro de src:
 - components
- Dentro de components, crie os arquivos:
 - Create.jsx
 - Edit.jsx
 - List.jsx
 - Home.jsx

Primeiros Componentes

- Create.jsx

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

export default class Create extends Component {
  render() {
    return (
      <div>
        <p>Create Component!!</p>
      </div>
    )
  }
}
```


Primeiros Componentes

- Edit.jsx

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

export default class Edit extends Component {
  render() {
    return (
      <div>
        <p>Edit Component!!</p>
      </div>
    )
  }
}
```

Primeiros Componentes

- List.jsx

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

export default class List extends Component {
  render() {
    return (
      <div>
        <p>List Component!!</p>
      </div>
    )
  }
}
```

Primeiros Componentes

- Home.jsx

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

export default class Home extends Component {
  render() {
    return (
      <div>
        <p>Home Component!!</p>
      </div>
    )
  }
}
```

Habilitando as Rotas

- index.js

```
import React from 'react';  
import ReactDOM from 'react-dom';  
import { BrowserRouter } from 'react-router-dom';  
import './index.css';  
import App from './App';  
import * as serviceWorker from './serviceWorker';
```

[//https://learnwithparam.com/blog/basic-routing-in-react-router/](https://learnwithparam.com/blog/basic-routing-in-react-router/)

```
ReactDOM.render(  
  <BrowserRouter>  
    <App />  
  </BrowserRouter>,  
  document.getElementById('root')  
);
```

*BrowserRouter é um
dos roteadores mais usados.*

No caso, aqui envolve toda a aplicação.

O Componente Principal

- App.js será nosso componente principal.
- A partir dele criaremos rotas para os outros componentes da aplicação, assim como a barra de navegação entre as páginas.

O Componente Principal

- Modifique o App.js (versão 0)

```
import React, { Component } from 'react';
import 'bootstrap/dist/css/bootstrap.min.css';
import { BrowserRouter as Router } from 'react-router-dom';

export default class App extends Component {
  render() {
    return (
      <Router>
        <div className="container">

          <h2>Projeto CRUD</h2> <br />

        </div>
      </Router>
    );
  }
}
```

O Componente Principal


- Barra de navegação (versão 1)

```
import React, { Component } from 'react';
import 'bootstrap/dist/css/bootstrap.min.css';
import { BrowserRouter as Router, Link } from 'react-router-dom';
```

```
export default class App extends Component {
  render() {
    return (
      <Router>
        <div className="container">
          <nav className="navbar navbar-expand-lg navbar-light bg-light">
            <Link to="/" className="navbar-brand">CRUD</Link>

          </nav>
          <h2>Projeto CRUD</h2> <br />

        </div>
      </Router>
    );
  }
}
```



Links só podem ser criados dentro de um roteador.

O Componente Principal

- Links de navegação (versão 2)

```
<nav className="navbar navbar-expand-lg navbar-light bg-light">
  <Link to="/" className="navbar-brand">CRUD</Link>
  <div className="collapse navbar-collapse" id="navbarSupportedContent">
    <ul className="navbar-nav mr-auto">
      <li className="nav-item">
        <Link to="/" className="nav-link">Home</Link>
      </li>
      <li className="nav-item">
        <Link to="/create" className="nav-link">Create</Link>
      </li>
      <li className="nav-item">
        <Link to="/list" className="nav-link">List</Link>
      </li>
    </ul>
  </div>
</nav>
```


O Componente Principal

- Lógica das Rotas (versão 3)

```
import { BrowserRouter as Router, Switch, Route, Link } from 'react-router-dom';
```

```
import Create from './components/Create';
```

```
import Edit from './components/Edit';
```

```
import List from './components/List';
```


```
import Home from './components/Home';
```

```
...
```

O Componente Principal

- Lógica das Rotas (versão 3)

```
...  
<h2>Projeto CRUD</h2> <br />  
<Switch>  
  <Route exact path="/" component={Home} />  
  <Route path="/create" component={Create} />  
  <Route path="/edit/:id" component={Edit} />  
  <Route path="/list" component={List} />  
</Switch>
```



O Switch, é um componente que recebe vários componentes Route e dado o caminho que for passado na URL um deles é renderizado.

Resultado

CRUD Home Create List

Projeto CRUD

Home Component!!

Parte 2

Formulário Criar Estudante

```

export default class Create extends Component {
  render() {
    return (
      <div style={{marginTop: 10}}>
        <h3>Criar Estudante</h3>
        <form>
          <div className="form-group">
            <label>Nome: </label>
            <input type="text" className="form-control"/>
          </div>
          <div className="form-group">
            <label>Curso: </label>
            <input type="text" className="form-control"/>
          </div>
          <div className="form-group">
            <label>IRA: </label>
            <input type="text" className="form-control"/>
          </div>
          <div className="form-group">
            <input type="submit" value="Criar" className="btn btn-primary"/>
          </div>
        </form>
      </div>
    )
  }
}

```

Create.jsx

Submetendo o Form

- O próximo passo é submeter o Formulário para o Back-End (ou seja, o componente `Create.jsx`)
- Para cada campo, teremos uma função que captura seu valor e modifica o estado do componente `Create.jsx`
- Vamos ver como ficaria o código para submeter apenas o nome, para os outros campos, a lógica é a mesma.

Submetendo o Form

- Crie o construtor

```
constructor(props){  
  super(props)  
  this.state = {nome:""}  
  
  this.setNome = this.setNome.bind(this)  
  this.onSubmit = this.onSubmit.bind(this)  
}
```

Submetendo o Form

- Os método de ajuste do 'nome' e submissão do formulário:

```
setNome(e){  
  this.setState({nome:e.target.value})  
}
```

```
onSubmit(e){  
  //https://www.robinwieruch.de/react-preventdefault  
  e.preventDefault() //impede que o browser faça o reload, perdendo assim a informação  
  console.log('Nome: ' + this.state.nome)  
  this.setState({nome:''})  
}
```


Submetendo o Form

- Modificando o Formulário

```
...  
<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>
```

*Métodos sendo chamados
no formulário.*

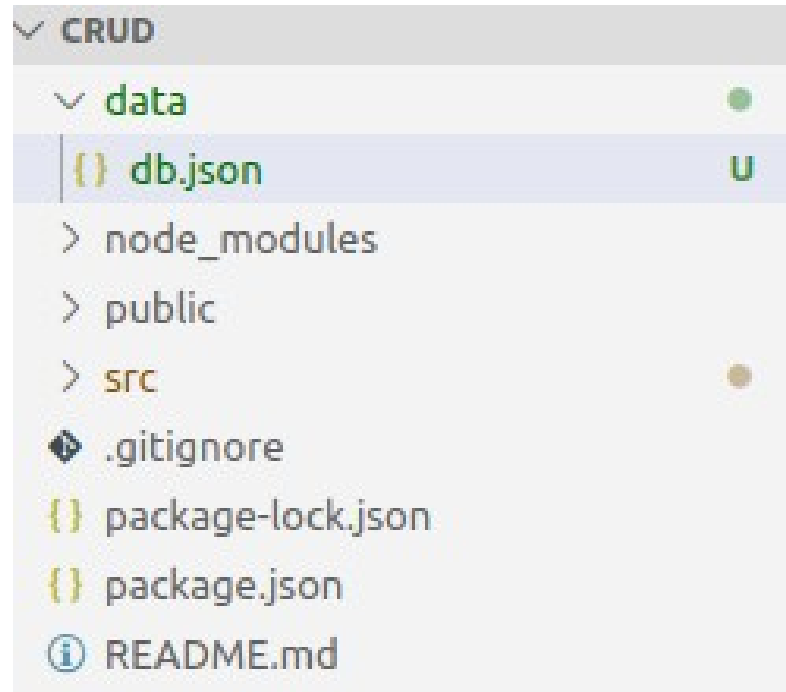
Submetendo o Form

- Exercício
 - Agora implemente para o resto dos campos, a mesma lógica do campo nome. Mostre os valores no método “onSubmit”

Json-Server

- <https://github.com/typicode/json-server>
- Instalando:
 - `sudo npm install -g json-server`
- Crie uma pasta chamada “data”, onde serão salvos os arquivos relativos as entidades do projeto CRUD.

Json-Server



Json-Server

- O arquivo db.json

```
{  
  "estudantes": [  
    {"id": 1, "nome": "Jefferson", "curso": "SI", "IRA": 9.0},  
    {"id": 2, "nome": "Thomas", "curso": "SI", "IRA": 7.5}  
  ]  
}
```

Json-Server

- `json-server --watch data/db.json --port 3001`
- Acesse <http://localhost:3001/estudantes>
- Dica:
 - Instale um cliente REST no seu navegador (extensão)
 - Postman ou ARC

Json-Server

The screenshot displays the ARC REST client interface. On the left sidebar, the 'History' section shows a list of requests, with the current request 'GET http://localhost:3001/estudantes' highlighted. The main panel is titled 'Request' and shows the method 'GET' and the request URL 'http://localhost:3001/estudantes'. Below this, the 'Parameters' section is expanded, showing 'Headers' and 'Variables' tabs. The 'Headers' tab is active, displaying a table with 'Header name' and 'Header value' columns. A green status bar indicates '200 OK' and '21.00 ms'. Below the status bar, the response body is shown as a JSON array of two objects, with the first object expanded to show its properties: 'id', 'nome', 'curso', and 'IRA'.

ARC

Request

HTTP request

Socket

History

Today

GET http://localhost:3001/estudantes

Method GET Request URL http://localhost:3001/estudantes

SEND

Parameters

Headers Variables

Toggle source mode Insert headers set

Header name Header value

ADD HEADER

Headers are valid Headers size: 0 bytes

200 OK 21.00 ms DETAILS

```
[Array[2]
  -0: {
    "id": 1,
    "nome": "Jefferson",
    "curso": "SI",
    "IRA": 9
  },
  -1: {
```

React → Json-Server

- O problema agora é fazer com que nossa aplicação se comunique com o Json-Server (e futuramente com o Mongo, via Express).
- Para ajudar nessa tarefa, iremos instalar o Axios.
 - `npm install axios --save`
- O Axios facilita a chamada dos métodos HTTP (PUT, DELETE, POST, GET...)

React → Json-Server

- Create.jsx

```
...
import axios from 'axios'

export default 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)
  }

  setNome(e){
    this.setState({nome:e.target.value})
  }

  ...
```

- Create.jsx

```
onSubmit(e){  
  //https://www.robinwieruch.de/react-preventdefault  
  e.preventDefault() //impede que o browser faça o reload, perdendo assim a informação  
  
  const novoEstudate = {nome:this.state.nome,  
                        curso:this.state.curso,  
                        IRA:this.state.IRA}  
  
  axios.post('http://localhost:3001/estudantes',novoEstudate)  
    .then(  
      (res)=>{  
        console.log(res.data.id)  
      }  
    )  
    .catch(  
      (error)=>{  
        console.log(error)  
      }  
    )  
  
  this.setState({nome:"",curso:"",IRA:""})  
}
```