



FIAP

GRADUAÇÃO

45697056



TDS

Responsive Web Development

Prof. Alexandre Carlos profalexandre.jesus@fiap.com.br

Prof. Luís Carlos lsilva@fiap.com.br





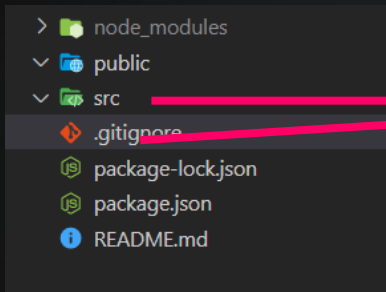
React

Novo projeto c/ pasta src e public do zero

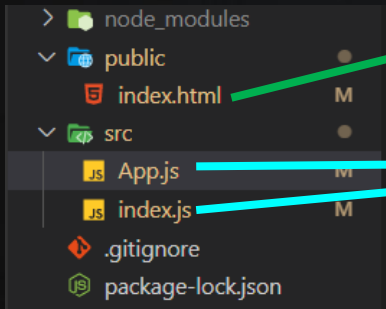
45897056



Agora vamos consumir a API que fizemos na aula passada. Crie um novo projeto chamado **react-aula9** e vamos **apagar** todos os arquivos das pastas **src** e **public**.



Apagamos todos os arquivos da pasta src e public



Logo após, na pasta public, recrie os arquivos index.html contendo a div c/ id='root'

E na pasta src os arquivos index.js com o ReactDOM.render() e o arquivo App.js como nosso componente principal.



Instalando bibliotecas

45697056
■ ■ ■

Para esta aula vamos precisar das 3 bibliotecas, a do **styled-components**, **react-router-dom** e a do **react-icons**. Vamos instalar as 3 em nosso projeto, se você já inicializou ele com o npm start, abra um outro terminal para executar as instalações (ctrl +shift + `) no VSCode.

- * Primeiro vamos instalar o styled-components:

```
npm install --save styled-components
```

- * Em segundo, vamos instalar o react-router-dom:

```
npm install --save react-router-dom
```

- * Em segundo, vamos instalar o react-icons:

```
npm install react-icons --save
```



Projeto com algumas novidades

45697056

No **App** vamos criar as Rotas que usaremos na aplicação. Para não termos erros já crie também os Componentes **ListaProduto.jsx** e **FormProduto.jsx** na pasta **components**:

```
src > JS App.js > ...
1 | import { BrowserRouter, Switch, Route } from "react-router-dom";
2 | import ListaProduto from "../components/ListaProduto";
3 | import FormProduto from "../components/FormProduto";
4 |
5 | const App = () => {
6 |   return (
7 |     <BrowserRouter>
8 |       <Switch>
9 |         <Route path="/" exact component={ListaProduto} />
10 |        <Route path="/incluir" component={FormProduto} />
11 |        <Route path="/editar/:id" component={FormProduto} />
12 |       </Switch>
13 |     </BrowserRouter>
14 |   );
15 | };
16 | export default App;
```

Está opção nos permite inserir parâmetros na path.



Projeto com algumas novidades

45697056

O primeiro componente que iremos usar será o **ListaProduto.jsx**. Nele iremos visualizar uma lista com os produtos (GET). Mas o arquivo vai ficar um pouco grande, vamos passo a passo, fique de olho no nº das linhas para não se perder:

```
export default function ListaProdutos(){  
  
  const [produtos, setProdutos] = useState([])  
  
  useEffect(()=>{  
    fetch("http://localhost:8080/LojaApp/rest/produto").then((resp)=>{  
      return resp.json();  
    }).then((resp)=>{  
      setProdutos(resp)  
      console.log(resp)  
    }).catch((error)=>{  
      console.log(error);  
    })  
  }, [])
```

O state produtos irá guardar nossos produtos, por isso um array vazio

Fetch é um método que permite acessar e manipular requisições e respostas HTTP de forma assíncrona.

Retorna promessas, onde transformamos os dados em json e depois inserimos em produtos.



Projeto com algumas novidades

45697056

Agora vamos criar a estrutura jsx do componente para vermos o retorno:

```
return(  
  <div>  
    <h1>Lista de Produtos</h1>  
    <table>  
      <thead>  
        <tr>  
          <th>Título</th><th>Preço</th><th>Quantidade</th><th></th>  
        </tr>  
      </thead>  
      <tbody>  
        {produtos.map((produto)=>(  
          <tr key={produto.codigo}>  
            <td>{produto.titulo}</td><td>R$ {produto.preco}</td>  
            <td>{produto.quantidade}</td><td></td>  
          </tr> ) )}  
      </tbody>  
      <tfoot>  
        <tr><td colspan='4'>Produtos do Servidor</td></tr>  
      </tfoot>  
    </table>  
  </div>  
)
```

Vamos deixar uma coluna vazia para usar com os botões depois

O map vai carregar todas as linhas com o retorno da requisição.



Projeto com algumas novidades

Ainda no componente `ListaProduto` vamos implementar a função para excluir um item da lista:

Devemos colocar a barra no final da uri e concatenar o id do produto que será excluído.

```
const handleDelete = (id)=>{
  fetch("http://localhost:8080/LojaApp/rest/produto/"+id,{
    method:"delete"
  }).then(()=>{
    window.location = "/"
  }).catch((error)=>{
    console.log(error);
  })
}
```

Na promessa retornaremos apenas o endereço que a página deverá ser recarregada.



Projeto com algumas novidades

45697056

Vamos inserir o botão Excluir e aproveitar e deixar o de inserir, que usaremos depois pronto.

Inserindo os botões de Editar e excluir na coluna que deixamos vazia.

```
40 <tbody>
41   {produtos.map((produto)=>(
42     <tr key={produto.codigo}>
43       <td>{produto.titulo}</td><td>R$ {produto.preco}</td>
44       <td>{produto.quantidade}</td>
45       <td>
46         <Link title="Editar" to={`/editar/${produto.codigo}`}>Editar</Link>
47         <button title="Excluir" onClick={handleDelete.bind(this, produto.codigo)}>
48           Excluir
49         </button>
50       </td>
51     </tr>
52   )</tbody>
```



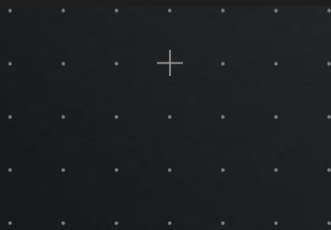
Projeto com algumas novidades

45697056

Já vamos deixar também um botão para irmos para nosso próximo componente onde poderemos Editar os produtos:

Link para página onde vamos editar os produtos

```
51  
52     return(  
53         <DivLista>  
54             <h1>Lista de Produtos</h1>  
55  
56             <Link to="/incluir">Inserir Produto</Link>  
57  
58             <table>
```





Projeto com algumas novidades

45697056

Vamos criar uma formatação simples e usar ícones nos botões:

```
3 import styled from 'styled-components'
4 import { FaEdit, FaTrash } from "react-icons/fa";
5
6 const DivLista = styled.div`
7   width: 70%; margin: auto; font-family: Arial;
8   h1{text-align:center;}
9   a{text-decoration:none; padding: 10px 15px; margin-bottom: 20px;
10     background-color: yellowgreen; color: white; display: inline-block;}
11   table{width: 100%; margin: auto;}
12   thead tr{ background-color: darkblue; color: white;}
13   thead tr th{padding: 10px;}
14
15   tbody tr:nth-child(2n+2) { background: #ccc;}
16   tbody tr td a{background: none; margin-bottom: 5px; color: blue;}
17   tbody tr td button{color: red; background:none; border: none;}
18
19   tfoot tr td{text-align: center; background: #333; color: white;}
20
21`
```

Não esqueça de inserir os botões no lugar de Editar e Excluir



Projeto com algumas novidades

45697056

Vamos para o **FormProduto.jsx**, faremos ele em algumas partes:

```
import {Link, useParams} from 'react-router-dom'
```

```
export default function FormProdutos(){
```

```
  let {id} = useParams();
```

```
  const [novo, setNovo] = useState({  
    codigo: id,  
    titulo:"",  
    preco:"",  
    quantidade:""  
  })
```

```
  let metodo = "post"  
  if(id){  
    metodo = "put"  
  }
```

```
  const handleChange = e=>{  
    setNovo({...novo, [e.target.name]:e.target.value})  
  }
```

Propriedades do react-router-dom para pegar o id da URI.

State para criarmos um novo produto.

Variável para trocarmos automaticamente o método POST e PUT.

Função para pegar os dados no formulário e inserir no state



Projeto com algumas novidades

Continuação:

```
const handleSubmit = e=>{
  e.preventDefault()

  fetch("http://localhost:8080/LojaApp/rest/produto/"+(id ? id : ""),{
    method: metodo,
    headers:{
      "Content-Type":"application/json"
    },
    body: JSON.stringify(novo)
  }).then(()=>{
    window.location = "/"
  })
}

useEffect(()=>{
  if(id){
    fetch("http://localhost:8080/LojaApp/rest/produto/"+id)
    .then(resp=>{
      return(resp.json())
    }).then(data=>{
      setNovo(data)
    })
  }
},[id])
```

Função que será disparada ao submeter o formulário.

Ternário que só insere o id se o tivermos.

useEffect carrega as informações no Formulário para editar



Projeto com algumas novidades

45697056

Vamos construir a estrutura do Formulário, utilizando nossas funções para pegar os dados e submeter o formulário:

```
53
54     return(
55         <div>
56             <h1>Formulário Produtos!!!</h1>
57             <form onSubmit={handleSubmit}>
58                 <input type="text" name="titulo" value={novo.titulo}
59                 placeholder="Título" onChange={handleChange}/><br/>
60                 <input type="number" name="preco" value={novo.preco}
61                 placeholder="Preço" onChange={handleChange} step="0.01" /><br/>
62                 <input type="number" name="quantidade" value={novo.quantidade}
63                 placeholder="Quantidade" onChange={handleChange} /><br/>
64                 <button type="submit">Enviar</button>
65                 <Link to="/">Cancelar</Link>
66             </form>
67         </div>
68     )
69 }
```



Projeto com algumas novidades

45697056

Para finalizar vamos criar uma formatação simples e inserir ícones nos botões:

```
2  import { Link } from 'react-router-dom'
3  import styled from 'styled-components'
4  import { FaLocationArrow, FaRegTimesCircle } from "react-icons/fa";
5
6  const DivForm = styled.div`
7    width: 70%; margin: auto; font-family: Arial;
8    h1{text-align: center;}
9    form{ width: 80%; margin:auto;}
10   form input{width: 100%; padding: 5px; margin-bottom: 5px;}
11   a{background: red; margin-bottom: 5px; color: white; text-decoration: none;
12     padding: 5px;}
13   button{color: white; background:green; border: none; display: inline-block;
14     padding:6px; margin-right: 10px;}
15
16
17`
```



Exercício

45697056



Crie um projeto React chamado exercicio8, ele deverá consumir a API Java do Exercício da apostila anterior



DUVIDAS





Copyright © 2015 - 2021 Prof. Luís Carlos S. Silva
Prof. Alexandre Carlos de Jesus

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).