FIAP

GRADUAÇÃO

45697056

TDS

Responsive Web Development

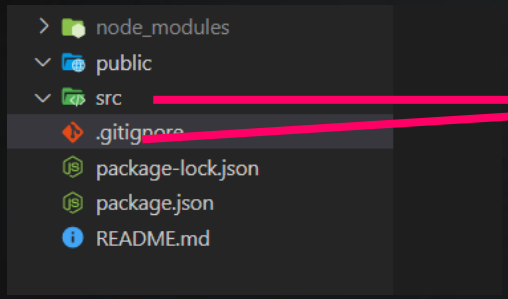
Prof. Alexandre Carlos profalexandre.jesus@fiap.com.br

Prof. Luís Carlos lsilva@fiap.com.br

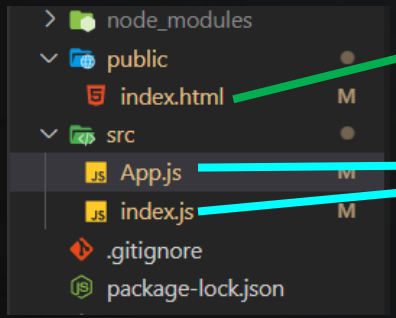
Novo projeto c/ pasta src e public do zero

45697056

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-componentes, 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

Na aula de hoje vamos consumir a API que criamos na aula passada utilizando o java e o Jersey.

A primeira coisa que precisamos fazer, para evitar erros de permissão de acesso (CORS) é colocar a URL da nossa API Java no package.json da seguinte forma:

package.json > {} scripts

```
{
  "name": "crud_jersey",
  "version": "0.1.0",
  "private": true,
  "proxy": "http://localhost:8080/LojaApp",
  "dependencies": {
    "@testing-library/jest-dom": "^5.14.1",
    "@testing-library/react": "^11.2.7",
    "@testing-library/user-event": "^12.8.3",
    "react": "^17.0.2",
```

Rode sua API Java e copie o endereço principal, a URL dela. Não coloque os paths internos.

Projeto com algumas novidades

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 > 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

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 olha no nº das linhas para não se perder:

src > components > ListaProduto.jsx > ListaProduto

```
1   import { useEffect, useState } from "react"
2
3   const ListaProduto = ()=>{
4
5       const [produtos, setProdutos]= useState([])
6
7       useEffect(()=>{
8           fetch("/rest/produto").then((resp)=>{
9               return resp.json();
10          }).then((resp)=>{
11              setProdutos(resp);
12              console.log(resp);
13          }).catch((error) => {
14              console.log(error);
15          })
16      },[])
```

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

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:

```
17      },[])
18
19
20      const handleDelete = (id)=>{
21          fetch("/rest/produto/"+id,{
22              method: "delete"
23          }).then(()=>{
24              window.location = "/"
25          }).catch((error) => {
26              console.log(error);
27          })
28      }
29
30
31      return(
```

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

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

Projeto com algumas novidades

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

Já vamos deixar também um botão para irmos para nosso próximo componente onde poderemos 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>
```

Link para página onde vamos editar os produtos

Projeto com algumas novidades

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

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

src > components > FormProduto.jsx > FormProduto

```
3   const FormProduto = ()=>{
4
5       let id = null;
6
7       if(props.match.path.toLowerCase().includes('editar')){
8           id = props.match.params.id;
9       }
10
11      const [novo, setNovo]= useState({
12          codigo: id,
13          titulo:"",
14          preco:"",
15          quantidade:""
16      })
17
18      let metodo = "post"
19      if(id){
20          metodo = "put"
21      }
22
23      const handleChange = e=>{
24          setNovo({...novo,[e.target.name]: e.target.value})
25      }
26
```

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:

```
26
27      const handleSubmit = e=>{
28          e.preventDefault()
29
30          fetch("/rest/produto/"+(id ? id : ""),{
31              method : metodo,
32              headers:{
33                  "Content-Type" : "application/json"
34              },
35              body: JSON.stringify(novo)
36          }).then(()=>{
37              window.location = "/"
38          })
39      }
40
41      useEffect(()=>{
42          if(id){
43              fetch("/rest/produto/"+id)
44              .then(resp=>{
45                  return(resp.json())
46              }).then(data=>{
47                  setNovo(data)
48              })
49          }
50      },[id])
```

Função que será disparada ao submeter o formulário.

Ternário que só insere o id se o tivermos.

Carrega as informações no Formulário para editar

Projeto com algumas novidades

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

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

```
import { Link } from 'react-router-dom'
import styled from 'styled-components'
import { FaLocationArrow, FaRegTimesCircle } from "react-icons/fa";

const DivForm = styled.div`
  width: 70%; margin: auto; font-family: Arial;
  h1{text-align: center;}
  form{ width: 80%; margin:auto;}
  form input{width: 100%; padding: 5px; margin-bottom: 5px;}
  a{background: red; margin-bottom: 5px; color: white; text-decoration: none;
    padding: 5px;}
  button{color: white; background:green; border: none; display: inline-block;
    padding:6px; margin-right: 10px;}


`
```

Exercício

45697056

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

DUVIDAS