

# PRÁTICAS AVANÇADAS EM DESENVOLVIMENTO WEB

Davi Schneid - [davi.Schneid@gmail.com](mailto:davi.Schneid@gmail.com)

29/07/2024

# Agenda

- ▶ React.js ou Angular.js

- ▶ Conceitos
- ▶ React X Angular
- ▶ Componentização
- ▶ useState

- ▶ Dependências

- ▶ Cors
- ▶ Axios

- ▶ Criar projeto web

- ▶ Criar páginas
- ▶ Criar componentes
- ▶ Navegar entre páginas
- ▶ Consumir API Backend.

# React

## Tipo de Ferramenta

Biblioteca focada em interfaces de usuário

## Arquitetura

Usa componentes reutilizáveis e independentes

## Vinculação de Dados

Unidirecional, facilitando a rastreabilidade do estado.

## DOM Document Object Model

Utiliza o Virtual DOM para melhorar o desempenho.

## Desempenho

Atualiza apenas componentes que mudaram

## Curva de Aprendizado

Geralmente mais fácil para desenvolvedores familiarizados com JavaScript.

## Linguagem

Usa JSX, uma extensão de sintaxe para JavaScript.

## Gerenciamento de Estado

Necessita de bibliotecas adicionais como Redux ou MobX.

## Flexibilidade

Alta flexibilidade para integrar com outras bibliotecas

## Ecossistema

Grande ecossistema com muitas bibliotecas de terceiros.

## Modularidade

Alta modularidade com componentes independentes.

# Angular

## Tipo de Ferramenta

Framework completo para desenvolvimento web.

## Arquitetura

Segue o padrão MVC, separando lógica de apresentação e negócios.

## Vinculação de Dados:

Bidirecional, permitindo alterações no modelo e na visão simultaneamente.

## DOM Document Object Model

Usa o Real DOM, o que pode afetar a performance em grandes aplicações.

## Desempenho

Pode enfrentar gargalos devido ao ciclo de digestão

## Curva de Aprendizado

Curva mais íngreme devido à sua complexidade e variedade de conceitos.

## Linguagem

Usa JavaScript, com a possibilidade de usar TypeScript.

## Gerenciamento de Estado

Gerenciamento de estado é incorporado no framework.

## Flexibilidade

Menos flexível, mas fornece uma solução completa

## Ecossistema

Ecossistema completo com ferramentas embutidas.

## Modularidade

Facilita a modularidade, mas pode ser mais complexa.

# React X Angular

Característica	React	AngularJS
Tipo	Biblioteca de JavaScript	Framework completo
Arquitetura	Baseada em componentes	MVC (Model-View-Controller)
Vinculação de Dados	Unidirecional	Bidirecional
DOM	Virtual DOM	Real DOM
Curva de Aprendizado	Relativamente rápida	Mais íngreme devido à complexidade
Linguagem	JavaScript com JSX	JavaScript (principalmente), pode usar TypeScript
Desempenho	Alto, devido ao Virtual DOM	Pode ter problemas de desempenho em grandes aplicações devido ao ciclo de digestão
Gerenciamento de Estado	Necessita de bibliotecas como Redux, MobX, etc.	Incorporado no framework, mas pode usar bibliotecas adicionais
Flexibilidade	Alta, fácil integração com outras bibliotecas	Menos flexível, mas completo
Ecossistema	Grande, com muitas bibliotecas e ferramentas de terceiros	Completo com ferramentas embutidas
Suporte e Comunidade	Mantido pelo Facebook, grande comunidade	Mantido pelo Google, grande comunidade
Documentação	Ampla e clara	Ampla e detalhada
Curva de Aprendizado Inicial	Rápida para quem conhece JavaScript	Mais complexa devido à variedade de conceitos e ferramentas
Modularidade	Alta, componentes independentes	Modularidade facilitada, mas pode ser complexa
Principais Usos	Interfaces de usuário dinâmicas e interativas	Aplicações web completas e estruturadas

# React - Angular

## ► Características do DOM (Document Object Model)

### 1. Estrutura em Árvore:

1. A DOM organiza o documento HTML ou XML em uma estrutura hierárquica em forma de árvore. Cada elemento, atributo e texto é um nó na árvore.

### 2. Interatividade e Manipulação:

1. A DOM permite que os desenvolvedores interajam com o conteúdo do documento. Eles podem acessar, modificar, adicionar ou remover elementos e atributos.

### 3. Dinâmica:

1. Através da DOM, é possível atualizar o conteúdo de uma página web sem precisar recarregá-la, permitindo a criação de experiências de usuário interativas e dinâmicas.

### 4. Acessibilidade:

1. A DOM pode ser acessada e manipulada usando linguagens de programação como JavaScript. As APIs da DOM fornecem métodos e propriedades para acessar e modificar o documento.

# React - Angular

## ► Tipos de DOM

### 1. Real DOM:

1. O Real DOM é a representação atual do documento na página web. Qualquer manipulação direta no Real DOM pode ser lenta, pois cada alteração pode causar re-renderização da página.

### 2. Virtual DOM:

1. O Virtual DOM é uma técnica usada por bibliotecas como React. Ele cria uma cópia leve e eficiente do DOM na memória, onde as manipulações são feitas. Apenas as mudanças mínimas necessárias são aplicadas ao Real DOM, o que melhora o desempenho.

# useState

- ▶ React Hook: é um hook que permite adicionar o estado React a componentes funcionais.
- ▶ Campos array de campos que serão atualizados
- ▶ setCampos nome da função

```
//Utilizada para auxiliar no controle de outras funcoes da aplicacao  
import React, { useState } from 'react';
```

```
//cria novo estado para os campos da tela  
const [campos, setCampos] = useState({  
  nome: '',  
  idade: 0,  
  cidade: ''  
});
```

- ▶ Benefícios
  - ▶ Simplicidade
  - ▶ Componentes Funcionais
  - ▶ Isolamento

# Componentes

- ▶ Componentes são blocos de construção independentes e reutilizáveis de uma interface de usuário em React.
  - ▶ Funcionais
  - ▶ Componentes de Classe

```
2  import React from 'react';
3
4  import { useNavigate } from 'react-router-dom';
5
6  const BotaoVoltar = () => {
7    const navigate = useNavigate();
8
9    const handleGoBack = () => {
10     navigate(-1); // Navega para a página anterior
11   };
12
13   return (
14     <div className="button-container">
15       <button onClick={handleGoBack}>
16         Voltar
17       </button>
18     </div>
19   );
20 };
21
22 export default BotaoVoltar;
```

```
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```



# O que é CORS?

- ▶ Define como os navegadores e servidores interagem para determinar se uma solicitação de recurso de um domínio diferente (origem cruzada) deve ser permitida.
- ▶ **Origem (Origin):** Refere-se à combinação do protocolo (http/https)
- ▶ Solicitação de Origem Cruzada
- ▶ Como Funciona CORS?
  - ▶ Cabeçalhos HTTP
    - ▶ Origin
    - ▶ Access-Control-Allow-Origin
    - ▶ Access-Control-Allow-Methods
    - ▶ Access-Control-Allow-Headers
    - ▶ Access-Control-Allow-Credentials
  - ▶ Preflight Requests

```
fetch('http://api.siteB.com/data', {  
  method: 'GET',  
  credentials: 'include' // Inclui cookies e cabeçalhos de autorização  
})  
.then(response => response.json())  
.then(data => console.log(data));
```

Origin: http://siteA.com

```
Access-Control-Allow-Origin: http://siteA.com  
Access-Control-Allow-Methods: GET  
Access-Control-Allow-Headers: Content-Type  
Access-Control-Allow-Credentials: true
```

# Axios

- ▶ Axios é uma biblioteca de JavaScript utilizada para fazer requisições HTTP de maneira fácil e intuitiva.
  - ▶ Suporta promessas (promises)
  - ▶ Funciona tanto no navegador quanto no Node.js.
  - ▶ Interceptores de requisição e resposta
  - ▶ Suporte o cancelamento de requisições
  - ▶ Transformação de dados antes e depois das requisições
  - ▶ `npm install axios`

```
axios.post('http://localhost:3001/api/usuarios', campos)
  .then(response => {
    setMensagem('Formulário enviado com sucesso!');
    console.log(response.data);
  });
```

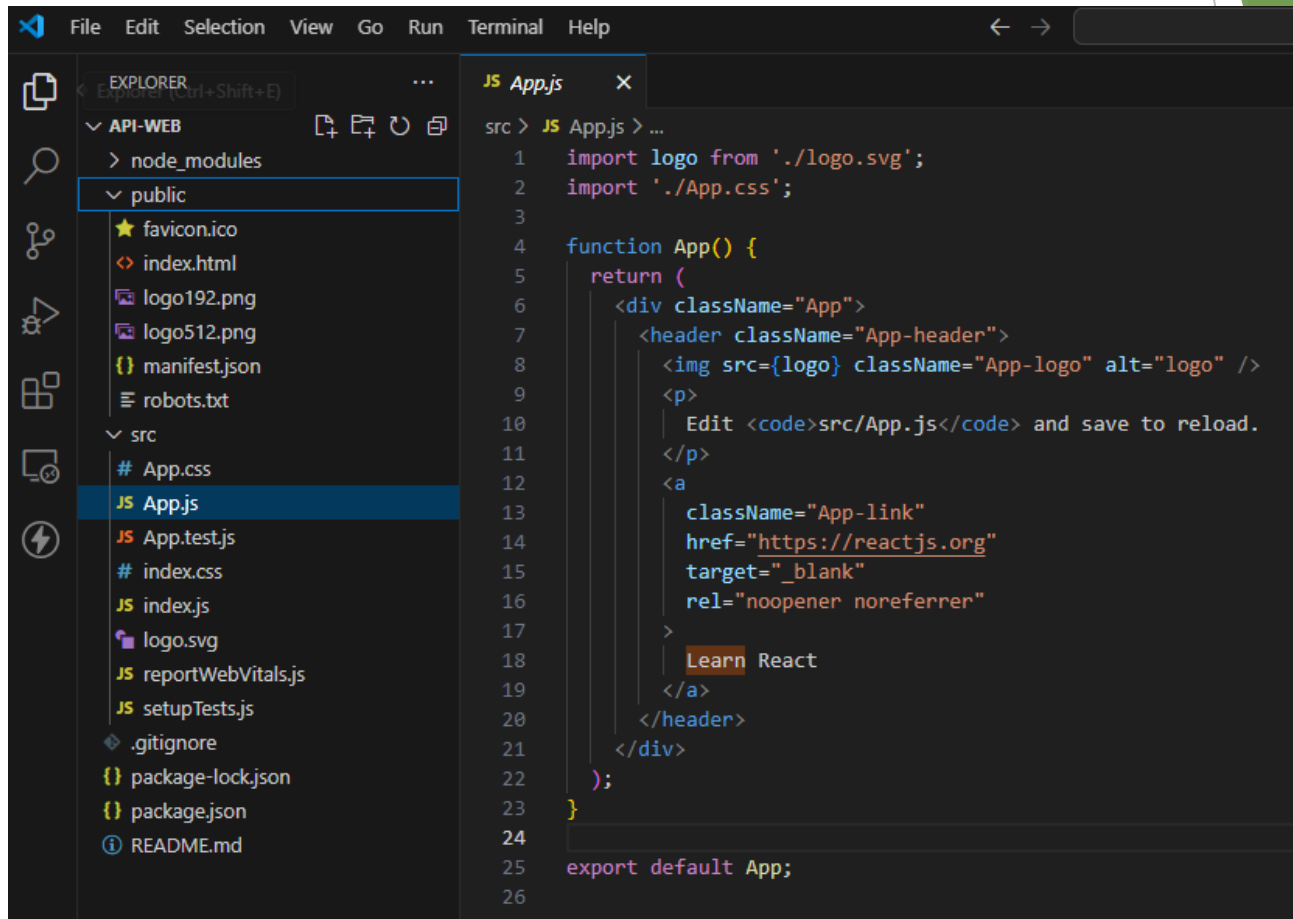
# API Web

- ▶ Abrir o Visual Studio Code
  - ▶ Abrir um novo terminal de comando.
  - ▶ Executar o comando: `npx create-react-app api-web`
  - ▶ Entrar no diretório da aplicação, executando o comando: `cd api-web`
  - ▶ Iniciar a aplicação, executando o comando: `npm start`



# API Web

- ▶ Abrir a pasta api-web pelo Visual Studio Code



# API Web

- ▶ Criar um componente Header
- ▶ Crie o arquivo chamado de Header.js dentro da pasta src
- ▶ Escreva o seguinte código dentro dele:

```
src > JS Header.js > [🔗] default

1  //cria a funcao Header
2  function Header() {
3      return (
4          <header>
5              <h1>Componente Header</h1>
6          </header>
7      );
8  }
9
10 export default Header;
```

# API Web

- ▶ Alterar o arquivo App.js
- ▶ Importar o componente Header
- ▶ Inserir o componente e renderizar dentro da DIV

```
File Edit Selection View Go Run Terminal Help
EXPLORER
API-WEB
  node_modules
  public
  src
    App.css
    App.js
    App.test.js
    Header.js
    index.css
    index.js
    logo.svg
    reportWebVitals.js
    setupTests.js
    .gitignore
    package-lock.json
    package.json
    README.md
JS App.js x JS index.js JS Header.js
src > JS App.js > App
1 import logo from './logo.svg';
2 import './App.css';
3
4 import Header from './Header';
5
6 function App() {
7   return (
8     /* Importamos o componente Header criado */
9     <Header/>
10
11     <header className="App-header">
12       <img src={logo} className="App-logo" alt="logo" />
13       <p>
14         Edit <code>src/App.js</code> and save to reload.
15       </p>
16       <a
17         className="App-link"
18         href="https://reactjs.org"
19         target="_blank"
20         rel="noopener noreferrer"
21       >
22         Learn React
23       </a>
24     </header>
25   </div>
26 );
27
28 }
29
30 export default App;
31
```

# API Web

- ▶ Criar um componente Footer
- ▶ Crie o arquivo chamado de Footer.js dentro da pasta src
- ▶ Escreva o seguinte código dentro dele:

```
src > JS Footer.js > Footer

1  //cria a funcao Footer
2  function Footer() {
3      return (
4          <footer className="App-footer">
5              <p>&copy; 2024 Meu Site. Todos os direitos reservados.</p>
6          </footer>
7      );
8  }
9
10 export default Footer;
```

# API Web

- ▶ Alterar o arquivo App.js
- ▶ Importar o componente Header
- ▶ Inserir o componente renderizar dentro da DIV

```
src > JS Appjs > ...

1  import logo from './logo.svg';
2  import './App.css';
3
4  //Importa o componente Header
5  import Header from './Header';
6
7  //Importa o componente Header
8  import Footer from './Footer';
9
10 function App() {
11   return (
12     <div className="App">
13       {/* Importamos o componente Header criado como HTML */}
14       <Header title="Parametro de titulo" />
15
16       <header className="App-header">
17         <img src={logo} className="App-logo" alt="logo" />
18         <p>
19           Edit <code>src/App.js</code> and save to reload.
20         </p>
21         <a
22           className="App-link"
23           href="https://reactjs.org"
24           target="_blank"
25           rel="noopener noreferrer"
26         >
27           Learn React
28         </a>
29       </header>
30
31       {/* Importamos o componente Footer criado como HTML */}
32       <Footer/>
33     </div>
34   );
35 }
36
37 export default App;
```



# API Web

- ▶ Criar pasta paginas dentro da pasta /src
- ▶ Criar o arquivo Cadastro.js dentro da pasta paginas
- ▶ Escrever o código dentro dele:

```
src > paginas > JS Cadastro.js > Cadastro
1
2 import Header from '../Header';
3
4 import '../App.css';
5
6
7
8 function Cadastro() {
9   return (
10     <div className="App">
11       <Header title="Formulario de Cadastro" />
12       Formulário de cadastro
13     </div>
14   )
15 }
16
17
18 export default Cadastro;
```

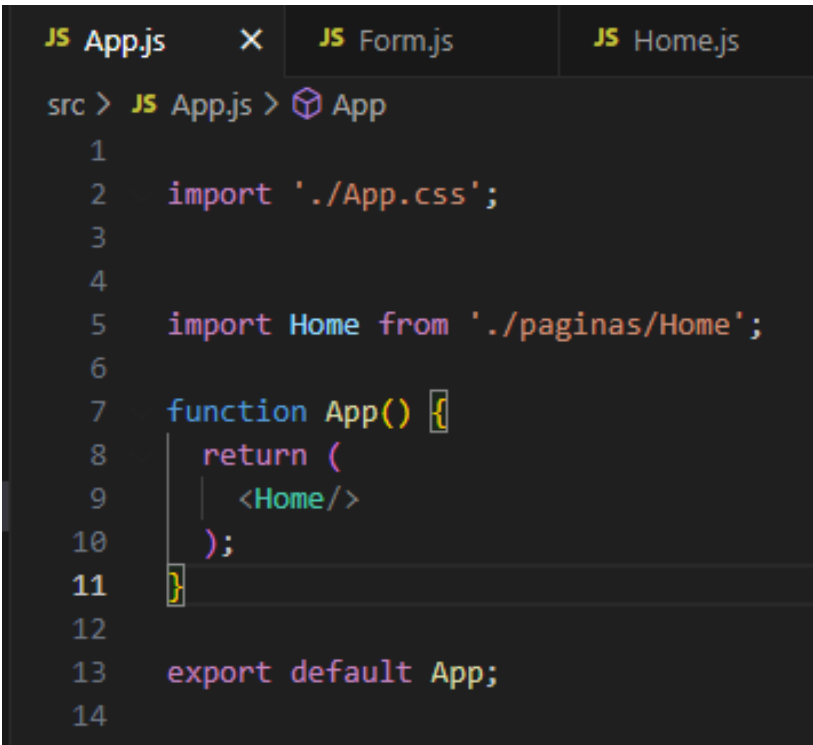
# API Web

- ▶ Criar o arquivo Home.js dentro da pasta paginas
- ▶ Copiar o código dentro do App.js e colar dentro do Home.js
- ▶ Alterar nome function Home no arquivo Home.js
- ▶ Ajustar o diretório dos Imports

```
src > paginas > JS Home.js > Home
1  import logo from '../logo.svg';
2  import '../App.css';
3
4  //Importa o componente Header
5  import Header from '../Header';
6
7  //Importa o componente Footer
8  import Footer from '../Footer';
9
10 function Home() {
11   return (
12     <div className="App">
13       /* Importamos o componente Header criado como HTML */
14       <Header title="Parametro de titulo" />
15
16       <header className="App-header">
17         <img src={logo} className="App-logo" alt="logo" />
18         <p>
19           Edit <code>src/App.js</code> and save to reload.
20         </p>
21         <a
22           className="App-link"
23           href="https://reactjs.org"
24           target="_blank"
25           rel="noopener noreferrer"
26         >
27           Learn React
28         </a>
29       </header>
30
31       /* Importamos o componente Footer criado como HTML */
32       <Footer />
33     </div>
34   );
35 }
36
37
38 export default Home;
39
```

# API Web

- ▶ Alterar o arquivo App.js
- ▶ Escrevendo o seguinte código:
- ▶



```
JS App.js  X  JS Form.js  JS Home.js
src > JS App.js > App
1
2  import './App.css';
3
4
5  import Home from './paginas/Home';
6
7  function App() {
8    return (
9      <Home/>
10    );
11  }
12
13  export default App;
14
```

# API Web

- ▶ Navegar em mais páginas.
- ▶ Para isso, instalar pacote de roteamento de páginas.
- ▶ Executar o comando: `npm install react-router-dom`
- ▶ Executar o comando `npm install react-router-dom@latest`

```
PS C:\Users\SenacRs\PraticasAvancadasDesenvolvimentoWeb\api-web> npm install react-router-dom

added 3 packages, and audited 1546 packages in 11s

261 packages are looking for funding
  run `npm fund` for details

8 vulnerabilities (2 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

```
PS C:\Users\SenacRs\PraticasAvancadasDesenvolvimentoWeb\api-web> npm install react-router-dom@latest

up to date, audited 1546 packages in 8s

261 packages are looking for funding
  run `npm fund` for details

8 vulnerabilities (2 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

# API Web

- ▶ Criar uma pasta chamada rotas dentro de src
- ▶ Essa pasta vai conter as rotas da aplicação.
- ▶ Criar o arquivo rotas.js dentro da pasta rotas
- ▶ Escrever o código abaixo

```
src > rotas > JS rotas.js > Rotas
1
2
3 import React from 'react';
4 import ReactDOM from 'react-dom';
5
6 //importa 3 objetos da lib
7 import { Route, Routes, BrowserRouter } from 'react-router-dom';
8
9 //Importa a página Home
10 import Home from '../paginas/Home';
11
12 //Importa a página Cadastro
13 import Cadastro from '../paginas/Cadastro';
14
15 function Rotas() {
16   return (
17     <BrowserRouter>
18       <Routes>
19         <Route element={<Home />} path="/" exact component={Home}/>
20         <Route element={<Cadastro />} path="/cadastro" component={Cadastro} />
21       </Routes>
22     </BrowserRouter>
23   )
24 }
25
26 export default Rotas;
```

# API Web

- ▶ Alterar o arquivo App.js para ele chamar as rotas criadas.
- ▶ Escrever o código abaixo no App.js
- ▶ Validar as rotas no Browser

```
src > JS App.js > [e] default
1
2 import './App.css';
3 import Rotas from './rotas/rotas';
4
5
6 function App() {
7   return (
8     <Rotas/>
9   );
10 }
11
12 export default App;
13
```

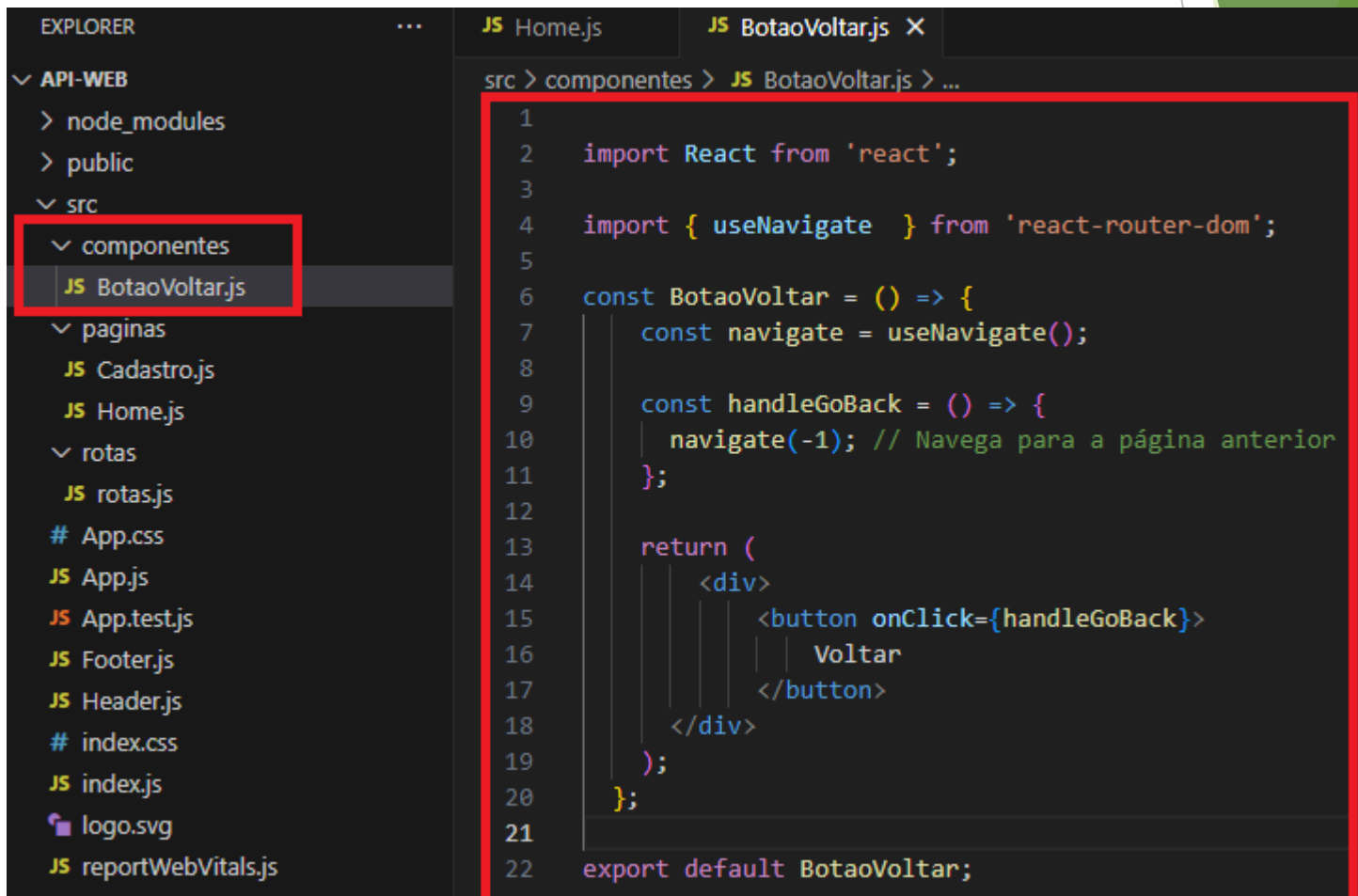
# API Web

- ▶ Adicionar Link á pagina Home
- ▶ Escrever o código abaixo no Home.js

```
src > paginas > JS Home.js > ...
1  import logo from '../logo.svg';
2  import '../App.css';
3
4  //Importa o componente Header
5  import Header from '../Header';
6
7  //Importa o componente Header
8  import Footer from '../Footer';
9
10 //Importa o recurso para criar link do react
11 import {Link} from 'react-router-dom';
12
13 function Home() {
14   return (
15     <div className="App">
16       /* Importamos o componente Header criado como HTML */
17       <Header title="Parametro de titulo" />
18
19       <header className="App-header">
20
21         <img src={logo} className="App-logo" alt="logo" />
22
23         <p>Praticas avançadas em Desenvolvimento Web.</p>
24
25         <Link to="/cadastro">Acessar cadastro</Link>
26
27       </header>
28
29       /* Importamos o componente Footer criado como HTML */
30       <Footer/>
31
32     </div>
33   );
34 }
35
36 export default Home;
37
```

# API Web

- ▶ Criando um componente
- ▶ Criar a pasta componentes dentro de SRC
- ▶ Criar o arquivo BotaoVoltar.js na pasta componentes
- ▶ Escrever o código no Botaovoltar.js



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays the project structure. The 'src' folder is expanded, and the 'componentes' subfolder is highlighted with a red box. Inside 'componentes', the file 'JS BotaoVoltar.js' is also highlighted with a red box. The main editor area shows the code for 'BotaoVoltar.js', which is also enclosed in a red box. The code defines a functional component that uses 'useNavigate' from 'react-router-dom' to create a 'Voltar' button that navigates to the previous page.

```
1
2  import React from 'react';
3
4  import { useNavigate } from 'react-router-dom';
5
6  const BotaoVoltar = () => {
7    const navigate = useNavigate();
8
9    const handleGoBack = () => {
10      navigate(-1); // Navega para a página anterior
11    };
12
13    return (
14      <div>
15        <button onClick={handleGoBack}>
16          Voltar
17        </button>
18      </div>
19    );
20  };
21
22  export default BotaoVoltar;
```



# API Web

- ▶ Utilizar o componente
- ▶ Adicionar o componente dentro da página Cadastro.js

```
src > paginas > JS Cadastro.js > Cadastro
1
2 import Header from '../Header';
3
4 import '../App.css';
5
6 import BotaoVoltar from '../componentes/BotaoVoltar';
7
8 function Cadastro() {
9   return (
10     <div className="App">
11       <Header title="Formulario de Cadastro" />
12       Formulário de cadastro
13       <BotaoVoltar></BotaoVoltar>
14     </div>
15   );
16 }
17
18
19
20 export default Cadastro;
```

# API Web

- ▶ Criar os campos do formulário cadastro
- ▶ Alterar o arquivo Cadastro.js escrevendo o código abaixo

```
src > paginas > JS Cadastro.js > Cadastro
1
2   import Header from '../Header';
3
4   import '../App.css';
5
6   import BotaoVoltar from '../componentes/BotaoVoltar';
7
8   function Cadastro() {
9
10    return [
11      <div className="App">
12        <Header title="Formulario de Cadastro" />
13
14        <form>
15          <fieldset>
16            <legend>
17              <h2>Dados de Cadastro</h2>
18            </legend>
19
20            <div>
21              <label>Nome:
22                <input type="text" name="nome" id="nome" />
23              </label>
24            </div>
25
26            <div>
27              <label>Idade:
28                <input type="number" name="idade" id="idade" />
29              </label>
30            </div>
31
32            <div>
33              <label>Cidade:
34                <input type="text" name="cidade" id="cidade" />
35              </label>
36            </div>
37
38            <input type="submit" value="Salvar" />
39          </fieldset>
40        </form>
41
42        <BotaoVoltar></BotaoVoltar>
43
44      </div>
45    ];
46  }
47
48  export default Cadastro;
```

# API Web

- ▶ Salvar os dados do formulário.
- ▶ Ler o estado de cada campo da tela, criando uma nova função
- ▶ Escrever o código abaixo na página cadastro.js:

```
7
8 //Utilizada para auxiliar no controle de outras funcoes da aplicacao
9 import {useState } from 'react';
10
11
12 function Cadastro() {
13
14     //cria novo estado para os campos da tela
15     const [campos, setCampos] = useState({
16         nome: '',
17         idade: 0,
18         cidade: ''
19     });
20
21
22     function handleInputChange(event) {
23         campos[event.target.name] = event.target.value;
24         setCampos(campos);
25     }
26
27     function handleFormSubmit(event){
28         event.preventDefault();
29         console.log(campos);
30     }
31
32
33     return (
```

# API Web

- ▶ Adicionar o evento `onChange={handleInputChange}` em cada input da página
- ▶ Adicionar o evento `onSubmit={handleFormSubmit}` no formulário

```
<form onSubmit={handleFormSubmit}>
  <fieldset>
    <legend>
      <h2>Dados de Cadastro</h2>
    </legend>

    <div>
      <label>Nome:
      <input type="text" name="nome" id="nome" onChange={handleInputChange}/>
    </div>

    <div>
      <label>Idade:
      <input type="number" name="idade" id="idade" onChange={handleInputChange}/>
    </div>

    <div>
      <label>Cidade:
      <input type="text" name="cidade" id="cidade" onChange={handleInputChange}/>
    </div>

    <input type="submit" value="Salvar" />
  </fieldset>
</form>
```

# API Web

- ▶ Ao preencher os dados do formulário e clicar em Salvar.
- ▶ Perceber que os dados informados estão sendo capturados

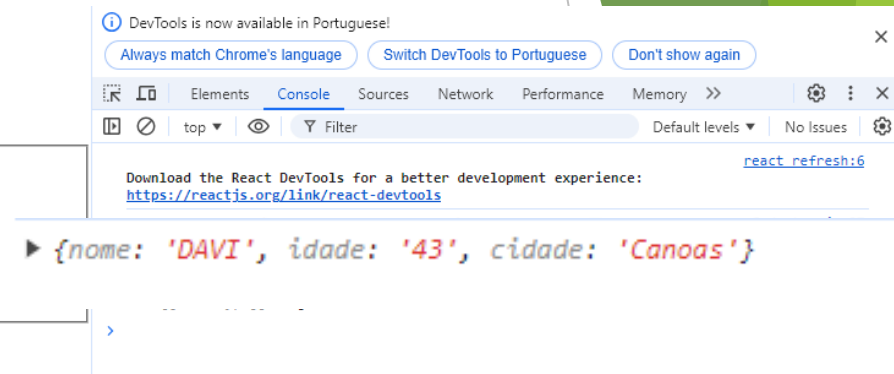
## Formulario de Cadastro

**Dados de Cadastro**

Nome:

Idade:

Cidade:



# API Router

- ▶ Mudar a porta da aplicação para 3001
- ▶ Editar o arquivo .env

```
.env
1 # The webapi port. Ex: 3000
2 PORT=3001
3
```

- ▶ Editar a porta para 3001 no arquivo Swagger.js

```
servers: [
  {
    url: 'http://localhost:3001/api',
  },
],
```

- ▶ Instalar a dependência no projeto ApiRouter: npm install cors

```
PS C:\Users\SenacRs\PraticasAvancadasDesenvolvimentoWeb\ApiRouter> npm install cors
added 2 packages, and audited 141 packages in 4s

15 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

# API Router

- ▶ Editar o arquivo server.js
- ▶ Adicionar código
- ▶ Rodar aplicação APIROUTER

```
JS server.js > ...
1
2   const express = require('express');
3   const sequelize = require('./data_base/db');
4   const usuariosRotas = require('./rotas/usuarioRotas');
5
6   //Importar o modulo Swagger
7   const setupSwagger = require('./swagger');
8
9   //importar o modulo cors para receber requisicoes de diferente origem
10  const cors = require('cors');
11
12  const app = express();
13  const PORT = process.env.PORT;
14
15
16  app.use(require("cors")());
17
18  //restringir chamadas somente da origem conhecida
19  const corsOptions = {
20    origin: 'http://localhost:3000', // Permitir apenas essa origem
21    methods: 'GET,HEAD,PUT,PATCH,POST,DELETE', // Métodos permitidos
22    credentials: true, // Permitir envio de cookies
23    optionsSuccessStatus: 204 // Status para requisições preflight
24  };
25
26  app.use(cors(corsOptions));
27
28  // Configurar Swagger
29  setupSwagger(app);
30
31  app.use(express.json());
32  app.use('/api', usuariosRotas);
33
34
35  sequelize.sync().then(() => {
36    app.listen(PORT, () => {
37      console.log(`Servidor rodando na porta ${PORT}`);
38    });
39  });
```

# API Web

- Adicionar o request POST usando o Axios

```
function handleFormSubmit(event){  
    event.preventDefault();  
    console.log(campos);  
    axios.post('http://localhost:3001/api/usuarios', campos).then(response => {  
        alert(response.status + ' cadastros!');  
    })  
}
```

- Cadastrar registro pela tela

The screenshot shows a web application interface. At the top, there's a navigation bar with a folder icon labeled 'Reservi' and a 'SharePoint - Get' link. Below this, a white modal dialog box is displayed with the text 'localhost:3000 diz' and '201 cadastros!' and an 'OK' button. Below the dialog, the title 'Dados de Cadastro' is centered. Underneath, there are three input fields: 'Nome:' with the value 'TESTE', 'Idade:' with the value '20', and 'Cidade:' with the value 'POA'. Below these fields are two buttons: 'Salvar' and 'Voltar'.



# API Web

- Adicionar uma mensagem de cadastro com sucesso.

```
13 function Cadastro() {
14
15     //cria novo estado para os campos da tela
16     const [campos, setCampos] = useState({
17         nome: '',
18         idade: 0,
19         cidade: ''
20     });
21
22     const [mensagem, setMensagem] = useState('');
23
24     function handleInputChange(event) {
25         campos[event.target.name] = event.target.value;
26         setCampos(campos);
27     }
28
29     function handleFormSubmit(event) {
30         event.preventDefault();
31         console.log(campos);
32         axios.post('http://localhost:3001/api/usuarios', campos).then(response => {
33             setMensagem('Formulário enviado com sucesso!');
34         })
35
36         // Mostrar mensagem de confirmação
37         setMensagem('Formulário enviado com sucesso!');
38
39         // Limpar mensagem após 3 segundos
40         setTimeout(() => {
41             setMensagem('');
42         }, 3000);
43
44 }
```

# API Web

- ▶ Adicionar validações dos campos.
- ▶ E limpar os campos da tela a cada cadastro.