

SENAC
Campus Santo Amaro

TADS - Análise Desenvolvimento de Sistemas

PW - Programação Web



Aula #5 React JS - Introdução

Professor: Veríssimo - carlos.hypereira@sp.senac.br

05/09/2022

Sobre este documento

Este documento objetiva deixar registrado o conteúdo abordado em sala de aula pelo professor. Importante destacar que a Nota de Aula serve como guia ao professor, bem como serve aos alunos como um norte, quanto ao conteúdo desenvolvido em sala de aula.

Este documento não tem a pretensão de ser uma única fonte para estudo. Para tal, o aluno deverá assistir às aulas e fazer uso (consulta) à bibliografia recomendada na ementa da disciplina, e à bibliografia complementar, apontada pelo professor.

Preâmbulo da Aula

Esta aula visa introduzir elementos que compõem conceitos fundamentais de ***REACT JS***.

Importante destacar que esta aula possui uma abordagem prática, na qual os elementos conceituais servem de guia para a parte prática da aula.

Contents

1	Introdução	1
1.1	JSX - JavaScript Syntax eXtension	1
1.2	Configurando Ambiente	2
2	Laboratorio React	5
2.1	A Verificar o Ambiente	5
2.2	Criar o projeto	6
2.2.1	A Explorar a estrutura do projeto	8
2.2.2	O arquivo "package.json": Scripts e de- pendências	9
2.2.3	Componentes raiz: pasta <i>src</i>	11
2.3	Abordando App.js e JSX	14
2.3.1	Estrutura Básica do componente	14
2.3.2	Inserir JSX	14

Chapter 1

Introdução

O *React* é uma **biblioteca JavaScript** para **construção de interfaces de usuário**. Utilizando-se da técnica de componentes(pequenos e isolados códigos), ele permite compor UIs complexas.

E onde podemos utilizá-lo?:

- Web(React em conjunto com ReactDOM)
- React Native (Aplicativos Móveis)
- Aplicações de realidade virtual

1.1 JSX - JavaScript Syntax eXtension

Criado pela equipe de desenvolvimento do React, JSX é uma extensão de sintaxe para JavaScript. JSX parece HTML mas é uma mistura de JavaScript e HTML

```
1 <MyButton color="blue" shadowSize={2}>  
2   Clique aqui
```

```
3 </MyButton>
4 //
5 //
6 const nomeCliente = 'Joaquim Jose da Sillva Xavier
  ';
7 const element = <h1>Hello, {nomeCliente}</h1>;
8
9 ReactDOM.render(
10   element,
11   document.getElementById('root')
12 );
13 //Outro exemplo
14 function formatarNome(cliente) {
15   return cliente.firstName + ' ' + cliente.
     lastName;
16 }
17
18 const cliente = {
19   firstName: 'Joaquim',
20   lastName: 'Xavier'
21 };
22
23 const elemento = (
24   <h1>
25     Hello, {formatarNome(cliente)}!
26   </h1>
27 );
28
29 ReactDOM.render(
30   elemento,
31   document.getElementById('root')
32 );
```

Listing 1.1: Exemplos de JSX

1.2 Configurando Ambiente

Nesta seção discutiremos aspectos básicos como: i) o que precisamos fazer para criar um projeto; ii) o que precisamos ter

instalado dentro de configuração do ambiente.

1. Instalar Node JS : <https://nodejs.org/en/>
2. Gerenciadores de pacote
 - Node Package Manager - NPM
 - yarn - [/yarnpkg.com/](https://yarnpkg.com/)

Chapter 2

Laboratorio React

faremos....

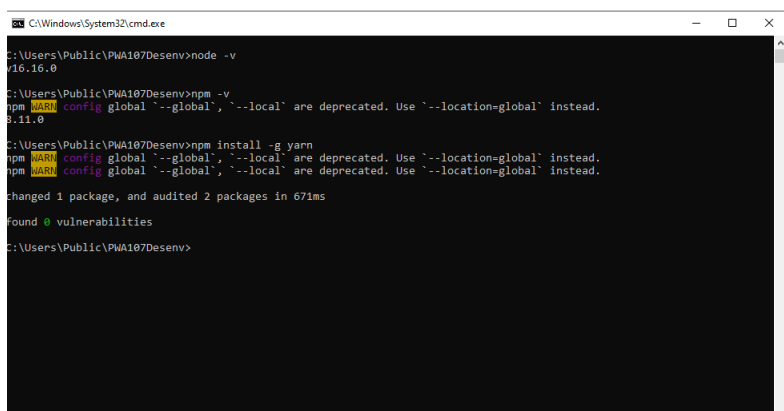
2.1 A Verificar o Ambiente

Verifique as versões do seu ambiente, dando os seguintes comandos no *prompt do DOS*:

- *node -v*
- *npm -v*
- *npm install -g yarn*

Veja o resultado, conforme demonstrado à figura 2.1.

Obs.: O **yarn** também serve como gerenciador de pacotes, e pode ser utilizado em substituição ao **npm**



```

C:\Windows\System32\cmd.exe
C:\Users\Public\PWA107Desenv>node -v
v16.16.0
C:\Users\Public\PWA107Desenv>npm -v
npm WARN config global '--global', '--local' are deprecated. Use '--location=global' instead.
6.11.0
C:\Users\Public\PWA107Desenv>npm install -g yarn
npm WARN config global '--global', '--local' are deprecated. Use '--location=global' instead.
npm WARN config global '--global', '--local' are deprecated. Use '--location=global' instead.
changed 1 package, and audited 2 packages in 671ms
Found 0 vulnerabilities
C:\Users\Public\PWA107Desenv>
```

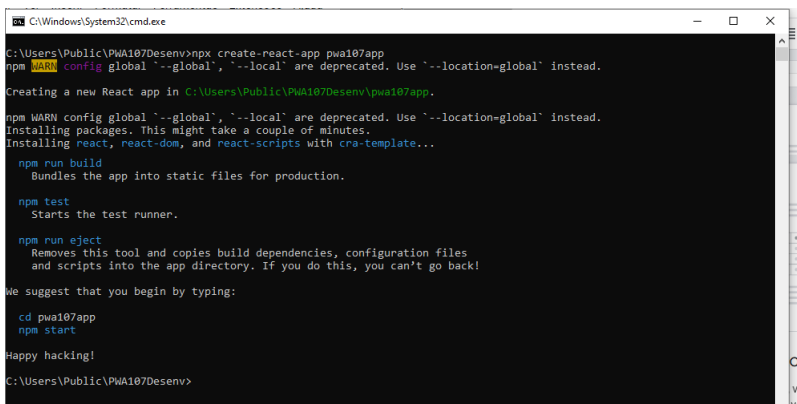
Figure 2.1: Verificando versão do ambiente

2.2 Criar o projeto

Vamos criar o projeto. Siga os seguintes procedimentos:

- Acione o comand prompt (cmd)
- Vá até a pasta na qual o projeto será desenvolvido e crie o projeto com nome **pwa107app**:
 - ***`npx create-react-app pwa107app`***

Veja o resultado da criação do projeto à figura 2.2.



```
C:\Windows\System32\cmd.exe
C:\Users\Public\pwa107Desenv>npm create-react-app pwa107app
npm WARN config global '--global', '--local' are deprecated. Use '--location=global' instead.
Creating a new React app in C:\Users\Public\pwa107Desenv\pwa107app.
npm WARN config global '--global', '--local' are deprecated. Use '--location=global' instead.
Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...
npm run build
  Bundles the app into static files for production.
npm test
  Starts the test runner.
npm run eject
  Removes this tool and copies build dependencies, configuration files
  and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

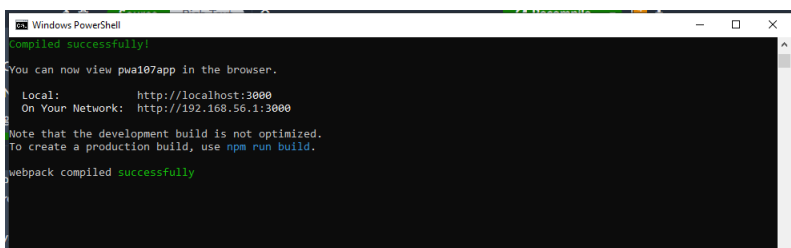
  cd pwa107app
  npm start

Happy hacking!
C:\Users\Public\pwa107Desenv>
```

Figure 2.2: A criar o projeto

A pasta, com o nome de seu projeto foi criada: "**pwa107app**", então entre nesta e inicie o seu app, com o comando: *npm start*.

Como resultado deste comando, o seu **app** agora está no ar, na **porta 3000**, conforme demonstrado à figura 2.3.



```
Windows PowerShell
Compiled successfully!

You can now view pwa107app in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.56.1:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

Figure 2.3: Start do App

veja também que o App agora a tela gerada, no seu navegador, conforme demonstrado à figura 2.4.

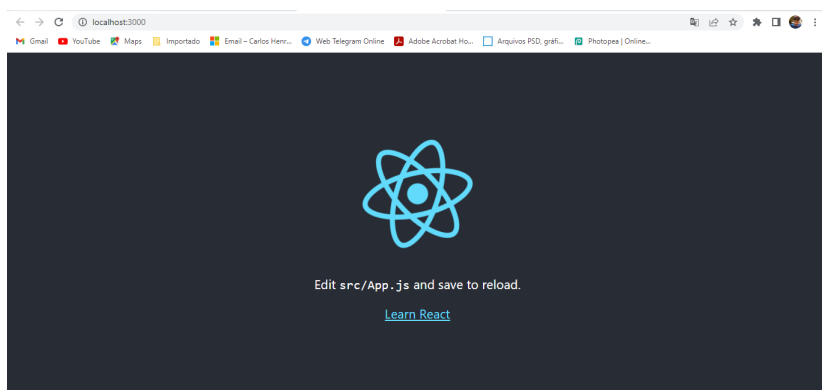


Figure 2.4: App no navegador

2.2.1 A Explorar a estrutura do projeto

Para explorar a estrutura de pastas do projeto, utilizamos o VS Code.

Iremos explorar a estrutura default para entendermos como as partes se conectam. O projeto foi criado com a estrutura de pastas demonstrada à figura 2.5.

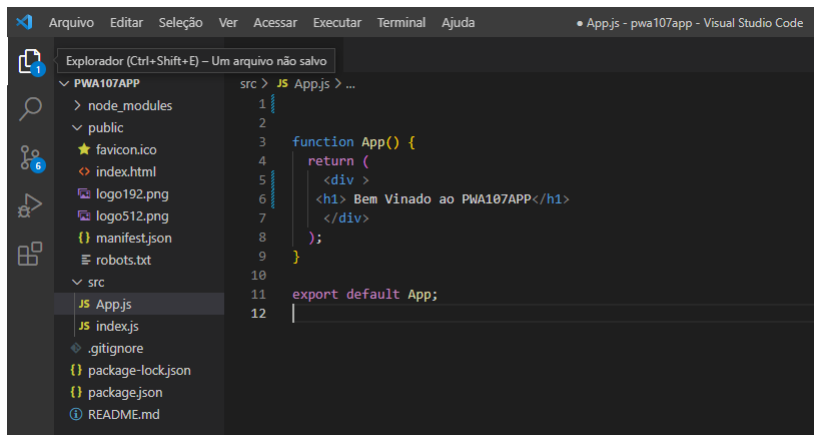


Figure 2.5: Estrutura do projeto

Abaixo consta a relação de pastas criadas, com seus respectivos objetivos:

- **node_modules**: Contém todas as dependências do projeto (Pasta criada pelo NPM).
- **package.json**: Contém as configurações do projeto com as dependências necessárias para rodar o projeto.
- **public**: Onde ficar, de fato, o código que irá para produção exemplo: index.html e favicon.ico. Nela ficam guardados os arquivos iniciais da aplicação, como o index.html que é interpretado pelo navegador, imagens e um arquivo JSON com os parâmetros de configuração do site.
- **src**: local para colocar os códigos a serem desenvolvidos. Nesta pasta temos código fonte da aplicação: Aqui ficarão todos os arquivos JavaScript e CSS que irão compor a nossa aplicação.

2.2.2 O arquivo "package.json": Scripts e dependências

O conteúdo do **package.json** lista as dependências do projeto e possui alguns aliases para os scripts envolvidos no *build*, conforme demonstrado à listagem 2.1.:

```
1 {  
2   "name": "pwa107app",  
3   "version": "0.1.0",  
4   "private": true,  
5   "dependencies": {  
6     "react": "^18.2.0",  
7     "react-dom": "^18.2.0",  
8     "react-scripts": "5.0.1"  
9   },
```

```
10 "scripts": {  
11   "start": "react-scripts start",  
12   "build": "react-scripts build",  
13   "eject": "react-scripts eject"  
14 },
```

Listing 2.1: Parte do conteúdo do arquivo package.json

Observe que temos os módulos **react** e **react-dom** declarados como dependência, e o **react-scripts** declarado como uma dependência de desenvolvimento. O **react-scripts** é o módulo que encapsula todos os scripts e configs do *build*.

Observe também os a declaração de alguns scripts:

- **start**: Inicia o build no modo de desenvolvimento;
- **build**: Executa o build do projeto otimizado para produção;
- **test**: Executa os testes do projeto;
- **eject**: Traz para dentro do nosso projeto, toda a configuração que o react-scripts abstrai.

O arquivo "public/index.html"

Observe o conteúdo do arquivo index.html, conforme demonstrado à listagem 2.2.:

```
1 <!DOCTYPE html>  
2 <html lang="en">  
3   <head>  
4     <meta charset="utf-8" />  
5     <link rel="icon" href="%PUBLIC_URL%/favicon.  
6       ico" />  
7     <meta name="viewport" content="width=device-  
8       width, initial-scale=1" />  
9     <meta name="theme-color" content="#000000" />  
10    <meta  
11      name="description"
```



```
10     content="Web site created using create-react
11         -app"
12     />
13     <link rel="apple-touch-icon" href="%PUBLIC_URL%
14         %/logo192.png" />
15
16     <link rel="manifest" href="%PUBLIC_URL%/
17         manifest.json" />
18
19     <title>App PWA107APP</title>
20 </head>
21 <body>
22     <noscript>You need to enable JavaScript to run
23         this app.</noscript>
24     <div id="root"></div>
25 </body>
26 </html>
```

Listing 2.2: Conteúdo do arquivo public/index.html

Note que este arquivo vem com a marcação mínima necessária para iniciar a aplicação. Importante observar que as *tags* de estilos e Scripts serão injetadas automaticamente no *build*.

Muita atenção á **linha #20**, pois **id="root"** será peça chave para nosso entendimento da renderização.

2.2.3 Componentes raiz: pasta *src*

App.js é o **componente principal** da aplicação. Observe o conteúdo do arquivo **App.js**, conforme demonstrado à listagem 2.3.:

```
1 function App() {
2   return (
3     <div >
4       <h1> Bem Vindo ao PWA107APP </h1>
5     </div>
6   );
7 }
```

```
8  
9 export default App;
```

Listing 2.3: Conteúdo do arquivo src/App.js

Existem duas formas de definir componentes: através de **functions**, ou através de **class**. Em nosso caso, utilizamos a primeira citada.

Agora confira o resultado no navegador demonstrado 2.6.

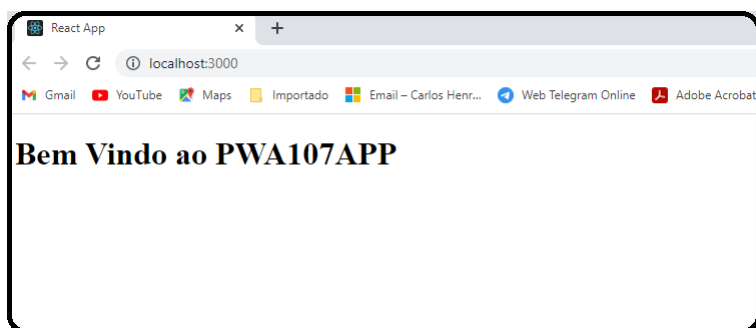


Figure 2.6: Resultado do componente App.js

Um componente deve sempre implementar um método render, que retorna um JSX do que deve ser mostrado na tela, ou null quando não deve mostrar nada.

index.js é responsável pela renderização do código, ou seja, visualização de toda a aplicação, conforme demonstrado à listagem 2.4.:

```
1 import React from 'react';  
2 import ReactDOM from 'react-dom/client';  
3 import App from './App';  
4  
5  
6 const root = ReactDOM.createRoot(document.  
  getElementById('root'));  
7 root.render(  
8   <React.StrictMode>  
9     <App />
```

```
10   </React.StrictMode>  
11 );
```

Listing 2.4: Conteúdo do arquivo `src/index.js`

Portanto, todo o código fonte de uma aplicação deverá ser **gerenciado** pelo **componente principal** `<App>` e **renderizado** pelo arquivo `src/index.js`.

2.3 Abordando App.js e JSX

Vimos que todo o código fonte deverá ser gerenciado pelo componente App e depois sua renderização. Mas, como isto acontece? Vamos, em primeiro plano entender a estrutura básica para iniciarmos uma aplicação com React, e depois entender o JSX:

2.3.1 Estrutura Básica do componente

Configure o arquivo **App.js**, com estrutura básica mínima do component; conforme demonstrado à listagem 2.6.:

```
1 import React from 'react';
2
3 function App() {
4   return (
5     <>
6
7     </>
8   );
9 }
10
11 export default App;
```

Listing 2.5: App.js com a configuração mínima

2.3.2 Inserir JSX

Agora que já temos a nossa estrutura mínima, podemos iniciar o nosso código JSX. O componente App somente poderá retornar um único elemento, ou seja, todo o código JSX deverá estar inserido entre as tags `<>` e `</>`, ou pelas tags `<div></div>..`

```
1 import React from 'react';
2
3 function App() {
```

```
4   return (  
5     <div>  
6       <h1> Bem Vindo ao PWA107APP</h1>  
7     </div>  
8   );  
9 }  
10  
11 export default App;
```

Listing 2.6: App.js com a configuração mínima

Saiba mais sobre o JSX

JSX é uma extensão da sintaxe do JavaScript criada para ajudar a dar forma aos elementos de *"front-end"* de uma aplicação desenvolvida em React. Veja abaixo algumas considerações sobre o JSX quando utilizado junto com o React:

- JSX é muito similar ao HTML e XML, mas não é HTML e nem XML;
- Diferentemente do HTML, o JSX não é interpretado pelo navegador e por isso é necessário a utilização de um "transpilador" (o React instalado no computador via NPM já possui um "transpilador" interno instalado);
- JSX parece HTML, mas é uma mistura de JavaScript e HTML, entretanto, é mais rápido e mais fácil que o JavaScript puro;
- O JSX permite colocar estruturas do tipo HTML dentro do JavaScript.