

—
ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
POLITÉCNICO
DO PORTO

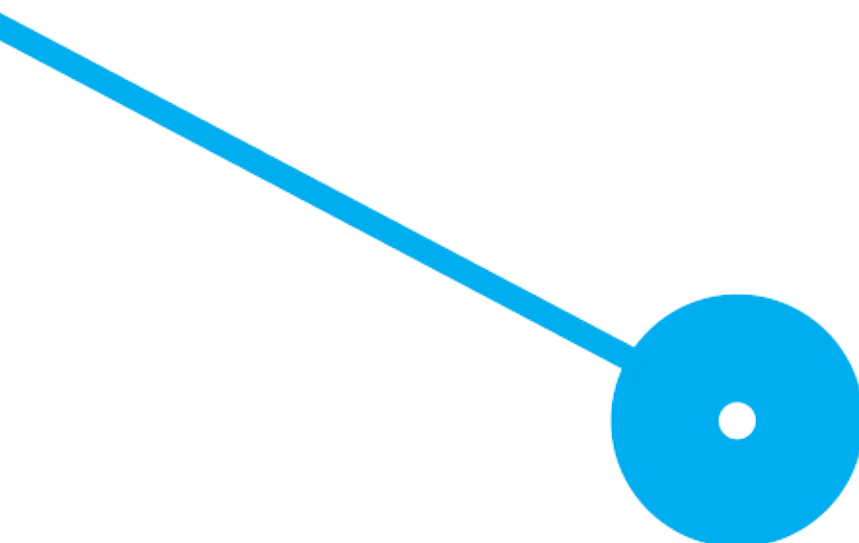
P.PORTO

L —
LICENCIATURA
ENGENHARIA INFORMÁTICA

Plataforma de gestão de treinos para atletas de alta competição

Daniel Sousa

Março de 2021



[PÁGINA PROPOSITADAMENTE DEIXADA EM BRANCO]

—
ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
POLITÉCNICO
DO PORTO

P.PORTO

L —
LICENCIATURA
ENGENHARIA INFORMÁTICA

Plataforma de gestão de treinos para atletas de alta competição

Daniel Sousa

Professor Ricardo Jorge Santos

Este trabalho não inclui as críticas e sugestões feitas pelo Júri

[PÁGINA PROPOSITADAMENTE DEIXADA EM BRANCO]

Conteúdo

Lista de Figuras	ii
Lista de Tabelas	iii
Lista de Excertos de Código	iv
Abreviaturas	v
Glossário	vii
Resumo	1
Apresentação do Autor	1
Apresentação da Entidade de Acolhimento	2
Convenções e Nomenclatura	3
1 Contextualização e Motivação	5
1.1 Introdução	5
1.2 Objetivos	5
1.3 Organização	5
2 Tecnologias	6
2.1 TypeScript	6
2.1.1 Instalação	7
2.1.2 Configuração	8
2.2 React	8
2.2.1 Criação do Projeto	8
2.2.1.1 Opções Adicionais	10
2.2.2 Estrutura de Pastas	10
2.2.3 Execução do Projeto	11

3	Ambiente de Desenvolvimento	13
4	Controlo de Versões	14
4.1	<i>Issues</i>	14
4.2	<i>Merge Requests</i>	14
4.3	<i>Board</i>	14
	Referências Bibliográficas	14
	Anexos	15
	TypeScript — Configurações	16
	Ficheiros React	18
	Visual Studio Code — Configurações	21

Lista de Figuras

2.1	TypeScript — logo	6
2.2	React — logo	8
2.3	Página inicial do React após a execução do projeto	10
2.4	React — possível estrutura de pastas	10
2.5	React — estrutura de pastas e ficheiros gerados pelo create-react-app	11
2.6	Projeto React executado com sucesso	12

Lista de Tabelas

2.1	Principais diferenças entre TypeScript e JavaScript	6
-----	---	---

Lista de Excertos de Código

1	Demonstração de excerto de código	4
2	Excerto de código com validação TypeScript	7
3	Scripts para a execução do projeto em React	9
4	TypeScript — Ficheiro tsconfig.json	17
5	Ficheiro index.html de um projeto React	19
6	Ficheiro index.jsx de um projeto React	19
7	Ficheiro index.css de um projeto React	19
8	Ficheiro app.jsx de um projeto React	20
9	Configurações utilizadas no Visual Studio Code	23

Abreviaturas

LEI Licenciatura em Engenharia Informática

ESTG Escola Superior de Tecnologia e Gestão

JSON *JavaScript Object Notation*

YAML *YAML Ain't Markup Language*

SQL *Structured Query Language*

HTML *Hyper Text Markup Language*

CSS *Cascading Style Sheets*

SASS *Syntactically Awesome Style Sheets*

JS *JavaScript*

TS *TypeScript*

NoSQL *No SQL –Not Only SQL*

NVM *Node Version Manager*

NPM *Node Package Manager*

JWT *JSON Web Token*

UI *User Interface*

UX *User Experience*

HTTP *HyperText Transfer Protocol*

CDN *Content Delivery Network*

CMS *Content Management System*

CRUD *Create, Read, Update, Delete*

DOM *Document Object Model*

MVC *Model-View-Controller*

REST *Representational State Transfer*

API *Application Programming Interface*

URL *Uniform Resource Locator*

DB *Database*

CI *Continuous Integration*

CD *Continuous Delivery*

IDE *Integrated Development Environment*

SVG *Scalable Vector Graphics*

CORS *Cross-Origin Resource Sharing*

SRP *Single Responsibility Principle*

Web *World Wide Web*

PWA *Progressive Web App*

RGPD *Regulamento Geral sobre a Proteção de Dados*

Glossário

Roles Forma de distinguir os diversos tipos de utilizadores de uma aplicação, contendo como tal diferentes tipos de permissões e ações possíveis de realizar

Responsive / Responsivo Conjunto de técnicas aplicadas a um *layout* de forma a este se adaptar a qualquer tamanho de ecrã, independentemente do dispositivo

Layout Forma como são organizadas ou distribuídas as diferentes partes de algo: *layout de armazém*, *layout do teclado*., por [Lexico](#)

Mockups protótipo de um projeto ou dispositivo, tendo como principal objetivo representar as principais funcionalidades do projeto/dispositivo. Utilizado frequentemente em projetos de desenvolvimento web para obter *feedback* do cliente

Front-end Parte vocacionada ao utilizador final, focada na *interface* visualizada, bem como a interação com o sistema. Essencialmente são usadas as linguagens/tecnologias **HTML**, **CSS** e **JavaScript**

Back-end Parte vocacionada na implementação, lógica e regras de negócio, não contendo *interface*. Nesta componente podem ser utilizadas linguagens como:

- C#;
- PHP;
- Java;
- Python;
- ...

Lazy Loading Consiste na técnica de adiar o carregamento de determinado componente ou class até este ser necessário.

Sprite Consiste numa imagem que contém múltiplas imagens, bastante utilizado para armazenar todos os ícones de uma aplicação num único ficheiro.

Packages São nada mais nada menos do que módulos/dependências que podem ser adicionados e utilizados ao longo de um determinado projeto.

Scripts TODO

Template TODO

Build TODO

Resumo

Palavras-chave: *React, Desenvolvimento Web, Front-end*

Apresentação do Autor

Apresentação da Entidade de Acolhimento

Convenções e Nomenclatura

Ao longo deste relatório, optou-se por seguir um conjunto de convenções de forma a facilitar a interpretação do texto, exemplos e excertos de código apresentados.

Desta forma textos em *itálico* terão como objetivo representar estrangeirismos, já textos em **negrito** terão como objetivo realçar termos com maior relevância ou mesmo nomes de empresas, marcas, etc..

Já em casos de textos sublinhados, por norma, referem-se a ligações no documento, por exemplo a ligação para uma determinada definição no glossário.

Além disso, sempre que seja pretendido realçar uma nota será utilizado o exemplo abaixo.

Nota

Informação da nota

Para além das notas e, recorrendo ao mesmo esquema, sempre que seja necessário apresentar informações relativas algum erro será utilizado o exemplo que se segue.

⚠ Erro Apresentado

Mensagem ou informações sobre o erro.

Já para apresentar excertos de código ao longo deste relatório, optou-se por utilizar o seguinte esquema:

```
1 // Exemplo de Excerto de Código
2 console.log("Hello World");
```

Excerto de Código 1: Demonstração de excerto de código

No que toca a nomenclatura e, tal como será possível analisar ao longo deste documento, são seguidas as seguintes regras:

- **Componentes React:** nomes em **Pascal Case**, ou seja, a primeira letra do identificador e a primeira letra de cada palavra são escritas em maiúsculas;
- **Interfaces:** seguem novamente o *naming convention* **Pascal Case** e começam pela letra **I**, que representa interface;

Capítulo 1

Contextualização e Motivação

1.1 Introdução

1.2 Objetivos

1.3 Organização

Capítulo 2

Tecnologias

Neste projeto foram utilizadas tecnologias tanto do lado do cliente, *front-end*, como do lado do servidor, *back-end*, apesar que o foco deste relatório é o lado do cliente (*front-end*), é necessário referir que este irá comunicar com o lado do servidor (*back-end*), onde estão armazenadas todas as informações da aplicação.

2.1 TypeScript



Figura 2.1:
TypeScript
— logo

O **TypeScript** é uma das tecnologias que é possível encontrar neste projeto tanto em *front-end* como *back-end*.

O **TypeScript**, segundo a própria **Microsoft** (detentora do **TypeScript**), é nada mais nada menos do que **JavaScript**, porém com a adição de tipos.

"TypeScript extends JavaScript by adding types."

TypeScript	JavaScript
Linguagem orientada a objetos	Linguagem de Scripting
Possui tipagem estática	Não possui tipagem
Suporte a módulos	Sem suporte a módulos
Possui suporte a parâmetros opcionais em funções	Não suporta a definição de parâmetros opcionais em funções

Tabela 2.1: Principais diferenças entre **TypeScript** e **JavaScript**

Devido a esta tipagem que é adicionada, o código torna-se mais facilmente interpretado, facilitando também o processo de *debug*, bem como as validações realizadas no processo de *build*. O excerto de código abaixo, retirado do [website oficial](#), tem como objetivo demonstrar a validação que é realizada pelo **TypeScript**.

```
1 const user = {  
2   firstName: "Angela",  
3   lastName: "Davis",  
4   role: "Professor"  
5 }  
6  
7 console.log(user.name)
```

Excerto de Código 2: Excerto de código com validação TypeScript

No caso, a linha 7 (assinalada com a cor vermelha), irá causar a mensagem de erro abaixo que indica que a propriedade **name** não existe no objeto **user**.

Erro Apresentado

Property 'name' does not exist on type '{ firstName: string; lastName: string; role: string; }'.

2.1.1 Instalação

A instalação do **TypeScript** pode ser realizada das seguintes maneiras:

- Globalmente:
 - Com Yarn: `yarn global add typescript`
 - Com NPM: `npm i -G typescript`
- Por Projeto:
 - Com Yarn: `yarn add -D typescript`
 - Com NPM: `npm i -D typescript`

A maneira mais comum é a instalação por projeto, visto que desta forma sempre que existir um *clone* do projeto e sejam instaladas as dependências¹, o **TypeScript** será também instalado e pronto a ser utilizado.

O uso de **TypeScript** pode implicar, em alguns casos, a instalação dos tipos (**@types**), por exemplo, no caso do **React** é necessário instalar os tipos recorrendo a `yarn add -D @types/react` ou `npm i -D @types/react`.

Nota

Como é possível analisar nos comandos de instalação do **TypeScript** por projeto, como na instalação dos tipos (**@types**), é usada a opção **-D** (tanto no uso do **Yarn** como do **NPM**), isto deve-se porque o **TypeScript** apenas será utilizado em desenvolvimento, uma vez que feito o *build* do projeto todo o código **TypeScript** é transformado em **JavaScript**.

¹Recorrendo a `yarn install` ou `npm install`.

2.1.2 Configuração

O **TypeScript** permite que o programador realizar determinadas configurações no seu projeto recorrendo a um ficheiro, no caso o ficheiro `tsconfig.json`,² neste ficheiro, tal como é possível encontrar em anexo, é possível definir desde configurações relacionadas com a estrutura de pastas do projeto, bem como definir *paths* para ajudar a manter as importações realizadas mais “enxutas”.

No exemplo em anexo é possível analisar que foi criado um *path* para a pasta **components**, desta forma sempre que seja realizada a importação de um componente é possível utilizar o *path* `@components/` seguido do nome do componente.

2.2 React

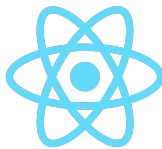


Figura 2.2:
React
— logo

Existem quem considere que o **React** é uma *framework* de **JavaScript**, porém e, ao mesmo tempo, há quem a considere como uma biblioteca de **JavaScript** baseada em componentes, sendo este o termo correto.

Os principais objetivos desta biblioteca são essencialmente:

- Fácil Aprendizagem;
- Rápidez;
- Escalável.

2.2.1 Criação do Projeto

A criação de um projeto **React** pode ser realizada de duas formas, manualmente ou recorrendo ao **create-react-app**, porém será possível analisar abaixo como proceder à criação de ambas as formas.

Para a criação de um projeto **React** manualmente é necessário adicionar todos os packages ao ficheiro `package.json`, para isso os passos a seguir são:

1. Criação da pasta para o projeto;
2. Aceder à pasta criada anteriormente via terminal e executar o comando `npm init -y` ou `yarn init -y` (caso seja utilizado **Yarn** como *package manager*);
3. Adicionar todos os packages necessários, sendo eles (por norma):
 - **React** — `npm i react` ou `yarn add react`;
 - **React Dom** — `npm i react-dom` ou `yarn add react-dom`;
 - **React Scripts** — `npm i react-scripts` ou `yarn add react-scripts`.
4. Após a instalação dos packages é necessário proceder à criação dos scripts, para isso é necessário adicionar o seguinte código no ficheiro `package.json`:

²[Documentação Oficial](#)

```
1 "scripts": {
2   "start": "react-scripts start",
3   "build": "react-scripts build",
4   "test": "react-scripts test",
5   "eject": "react-scripts eject"
6 },
```

Excerto de Código 3: *Scripts para a execução do projeto em React*

5. Posto isto é necessário criar todos os ficheiros necessários para a aplicação funcionar. Sendo eles:

- `index.html` (na pasta `public`)
- `index.css` (na pasta `src`)
- `index.jsx` (na pasta `src`)
- `App.jsx` (na pasta `src`)
- `App.css` (na pasta `src`)

Nota

Estes ficheiros é possível encontrar em anexo, mais precisamente em [ficheiros React](#). É ainda importante referir que estes ficheiros são apenas a base para colocar um projeto **React** a funcionar.

Porém como é possível analisar este processo é um pouco mais trabalhoso e implica que o programador saiba quais as dependências que necessita, para isso é possível usar o **create-react-app** que é o método recomendado pelo **React**³ para criar um projeto.

Os passos para a criação de um projeto seguindo este método são bastante simples e práticos, permitindo ainda ao programador definir se pretende usar ou não algum *template*, como por exemplo **TypeScript**. Os passos que se seguem demonstram a criação de um projeto **React** através desta “ferramenta”:

1. Em primeiro lugar é necessário instalar o **create-react-app**, isto pode ser realizado de duas formas de acordo com o *package manager* utilizado:

- **Com Yarn:** `yarn global add create-react-app`
- **Com NPM:** `npm install -g create-react-app`

2. Após a instalação é agora possível criar agora o projeto, para tal:

- **Com Yarn:** `yarn create react-app <project-name> [<options>]`
- **Com NPX:** `npx create-react-app <project-name> [<options>]`
- **Com NPM:** `npm init react-app <project-name> [<options>]`

³ Documentação: [“Create a new React App”](#)

Com isto é possível aceder à pasta do projeto (sendo a pasta o nome do projeto — `<project-name>`) e verificar que todos os *packages* foram adicionados, bem como os ficheiros base, inclusive o logo do **React** que irá aparecer como animação ao executar o projeto (ver figura abaixo).



Figura 2.3: Página inicial do **React** após a execução do projeto

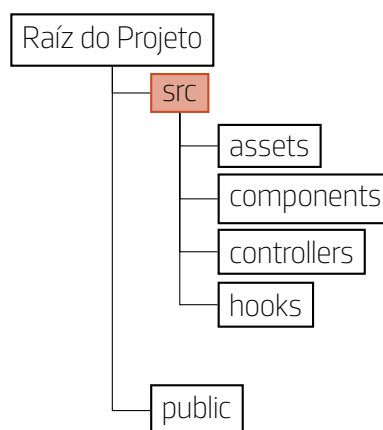
2.2.1.1 Opções Adicionais ⁴

Na criação de um projeto **React** através do **create-react-app** é possível especificar o *template* a usar, não sendo de uso obrigatório. Neste *template* é possível especificar, por exemplo, para ao gerar o projeto **React** gerar com **TypeScript** e não **JavaScript**.

Além do *template* é ainda possível especificar o *package manager* utilizado, recorrendo à opção `--use-npm`, isto para usar o **NPM** como *package manager*⁵.

2.2.2 Estrutura de Pastas

A estrutura de pastas para um projeto **React** pode variar de projeto para projeto, ou da forma como o programador prefere organizar os mais diversos ficheiros do projeto. Porém e, tal como é possível analisar na figura que se segue, é comum encontrar a seguinte estrutura de pastas.



⁴**Nota:** as opções referidas são apresentadas acima como `<options>`.

⁵No caso de possuir o **Yarn** instalado.

Figura 2.4: React — possível estrutura de pastas

Importante referir que a pasta **src/** será a pasta principal, uma vez irá conter todos os componentes, *assets* e outros ficheiros importantes para o projeto.

Nota

É importante relembrar que consoante o uso de **TypeScript** ou **JavaScript** será possível encontrar ficheiros **.tsx**, **.ts**, **.jsx** ou **.js**.

Além da extensão dos ficheiros será possível encontrar na raiz do projeto um ficheiro de configuração, podendo ser:

- **tsconfig.json** — a quando a utilização de **TypeScript**;
- **jsconfig.json** — a quando a utilização de **JavaScript**.

É importante referir que seguindo o método de criação do projeto **React** com o **create-react-app**, a estrutura de pastas e os ficheiros criados inicialmente será a seguinte:

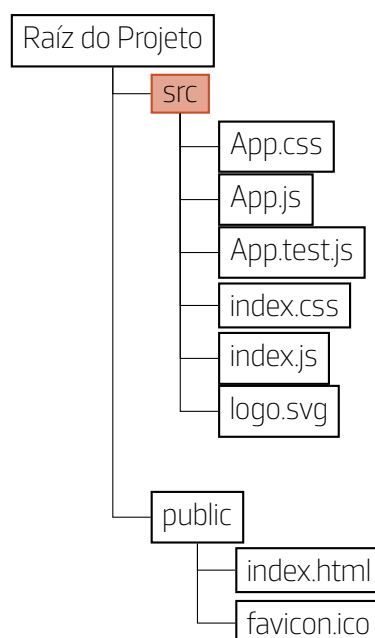


Figura 2.5: React — estrutura de pastas e ficheiros gerados pelo **create-react-app**

2.2.3 Execução do Projeto

Após a criação do projeto é agora possível executar o mesmo, para tal é possível utilizar os scripts presentes no ficheiro **package.json**, para isso basta utilizar um dos comandos que se segue de acordo com o *package manager* em uso:

- **Yarn:** **yarn start**;
- **NPM:** **npm start**

Se tudo correr como esperado será apresentado a seguinte mensagem no terminal:

```
Compiled successfully!

You can now view [redacted] in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.8.129:3000

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

█
```

Figura 2.6: Projeto **React** executado com sucesso

Nota

De notar que os comandos apresentados são para executar o projeto em modo de desenvolvimento, caso seja pretendido realizar o *build* para colocar o projeto em produção os comandos a executar são:

- Com Yarn: `yarn build`,
- Com NPM: `npm run build`.

Capítulo 3

Ambiente de Desenvolvimento

Capítulo 4

Controlo de Versões

4.1 *Issues*

4.2 *Merge Requests*

4.3 *Board*

Referências Bibliográficas

- [1] Alex Banks e Eve Porcello. *Learning React. Functional Web Development with React and Redux*. Inglês. O'Reilly, mai. de 2017. ISBN: 978-1-491-95462-1.
- [2] Pavels Jelisejevs. *Create React App: Get React Projects Ready Fast*. Inglês. SitePoint. Nov. de 2020. URL: <https://www.sitepoint.com/create-react-app/> (acedido em 15/03/2021).
- [3] *O que é ReactJS e porque devemos usá-lo?* Português. EDIT. Dez. de 2018. URL: <https://edit.com.pt/blog/o-que-e-reactjs-e-porque-devemos-usa-lo/> (acedido em 15/03/2021).
- [4] *React JS. Notes for Professionals*. Inglês. GoalKicker.com, mai. de 2018. URL: <https://goalkicker.com/ReactJSBook/>.
- [5] Filipe Portela e Ricardo Queirós. *Introdução ao Desenvolvimento Moderno para Web. Do front-end ao back-end: uma visão global!* Português. FCA, 2018. ISBN: 978-972-722-897-3.
- [6] Prahlad Yeri. *under_scores, camelCase and PascalCase - The three naming conventions every programmer should be aware of*. Inglês. Jul. de 2019. URL: <https://dev.to/prahldyeri/underscores-camelcasing-and-pascalcasing-the-three-naming-conventions-every-programmer-should-be-aware-of-3aed> (acedido em 13/03/2021).

Anexos

TypeScript – Configurações

```
1 {
2   "compilerOptions": {
3     "target": "es5",
4     "lib": [
5       "dom",
6       "dom.iterable",
7       "esnext"
8     ],
9     "allowJs": true,
10    "skipLibCheck": true,
11    "esModuleInterop": true,
12    "allowSyntheticDefaultImports": true,
13    "strict": true,
14    "forceConsistentCasingInFileNames": true,
15    "noFallthroughCasesInSwitch": true,
16    "module": "esnext",
17    "moduleResolution": "node",
18    "resolveJsonModule": true,
19    "isolatedModules": true,
20    "noEmit": true,
21    "jsx": "react",
22    "experimentalDecorators": true,
23    "baseUrl": "src",
24    "rootDir": "src",
25    "paths": {
26      "@components/*": [
27        "src/components/*"
28      ]
29    }
30  },
31  "include": [
32    "src"
33  ]
34 }
```

Excerto de Código 4: TypeScript – Ficheiro *tsconfig.json*

Nota

O ficheiro **tsconfig.json** apresentado tem como objetivo apresentar apenas uma possível estrutura de configuração. É recomendado consultar a [documentação oficial](#) relativa a este ficheiro.

É importante referir que este ficheiro deve encontra-se na raíz do projeto para garantir o seu correto funcionamento.

Ficheiros React

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="utf-8" />
6   <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
7   <meta name="viewport" content="width=device-width, initial-scale=1" />
8   <meta name="theme-color" content="#000000" />
9   <meta name="description" content="Web site created using create-react-app" />
10  <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
11  <!--
12    manifest.json provides metadata used when your web app is installed on a
13    user's mobile device or desktop. See
14    ↪ https://developers.google.com/web/fundamentals/web-app-manifest/
15    -->
16  <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
17  <!--
18    Notice the use of %PUBLIC_URL% in the tags above.
19    It will be replaced with the URL of the `public` folder during the build.
20    Only files inside the `public` folder can be referenced from the HTML.
21
22    Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
23    work correctly both with client-side routing and a non-root public URL.
24    Learn how to configure a non-root public URL by running `npm run build`.
25    -->
26  <title>React App</title>
27</head>
28<body>
29  <noscript>You need to enable JavaScript to run this app.</noscript>
30  <div id="root"></div>
31  <!--
32    This HTML file is a template.
33    If you open it directly in the browser, you will see an empty page.
34
35    You can add webfonts, meta tags, or analytics to this file.
36    The build step will place the bundled scripts into the <body> tag.
37
38    To begin the development, run `npm start` or `yarn start`.
39    To create a production bundle, use `npm run build` or `yarn build`.
40    -->
41</body>
42
43</html>
```

Excerto de Código 5: Ficheiro *index.html* de um projeto React

```
1 import React from 'react';
2 import ReactDOM from 'react-dom';
3 import App from './App';
4 import './index.css';
5
6 ReactDOM.render(
7   <React.StrictMode>
8     <App />
9   </React.StrictMode>,
10  document.getElementById('root')
11 );
```

Excerto de Código 6: Ficheiro *index.jsx* de um projeto React

```
1 body {
2   margin: 0;
3   font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
4     'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
5     sans-serif;
6   -webkit-font-smoothing: antialiased;
7   -moz-osx-font-smoothing: grayscale;
8 }
9
10 code {
11   font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
12     monospace;
13 }
```

Excerto de Código 7: Ficheiro *index.css* de um projeto React

```
1 import React from 'react';
2 import './App.css';
3
4 function App() {
5   return (
6     <div className="App">
7       <header className="App-header">
8         <p>
9           Edit <code>src/App.jsx</code> and save to reload.
```

```
10     </p>
11     <a
12         className="App-link"
13         href="https://reactjs.org"
14         target="_blank"
15         rel="noopener noreferrer"
16     >
17         Learn React
18     </a>
19 </header>
20 </div>
21 );
22 }
23
24 export default App;
```

Excerto de Código 8: Ficheiro *app.jsx* de um projeto *React*

Visual Studio Code — Configurações

```
1 {
2   "[javascript]": {
3     "editor.defaultFormatter": "esbenp.prettier-vscode",
4     "editor.formatOnPaste": true,
5     "editor.formatOnType": true,
6     "editor.tabSize": 4,
7     "editor.detectIndentation": false,
8     "editor.insertSpaces": false,
9     "editor.formatOnSave": true
10  },
11  "javascript.suggest.enabled": true,
12  "javascript.updateImportsOnFileMove.enabled": "never",
13  "javascript.suggest.autoImports": true,
14  "[typescript]": {
15    "editor.formatOnPaste": true,
16    "editor.defaultFormatter": "esbenp.prettier-vscode",
17    "editor.tabSize": 4,
18    "editor.detectIndentation": false,
19    "editor.formatOnType": false,
20    "editor.formatOnSave": true
21  },
22  "[typescriptreact]": {
23    "editor.formatOnPaste": true,
24    "editor.defaultFormatter": "esbenp.prettier-vscode",
25    "editor.tabSize": 4,
26    "editor.detectIndentation": false,
27    "editor.formatOnType": false,
28    "editor.formatOnSave": true
29  },
30  "[javascriptreact]": {
31    "editor.defaultFormatter": "esbenp.prettier-vscode",
32    "editor.formatOnPaste": true,
33    "editor.formatOnType": true,
34    "editor.tabSize": 4,
35    "editor.detectIndentation": false,
36    "editor.insertSpaces": false,
37    "editor.formatOnSave": true
38  },
39  "typescript.suggest.enabled": true,
40  "typescript.autoClosingTags": true,
41  "typescript.preferences.quoteStyle": "single",
42  "typescript.updateImportsOnFileMove.enabled": "never",
43  "typescript.tsserver.log": "verbose",
44  "typescript.suggest.autoImports": true,
45  "eslint.validate": [
```

```

46     "javascript",
47     "typescript"
48 ],
49 "[html]": {
50     "editor.defaultFormatter": "vscode.html-language-features",
51     "editor.formatOnPaste": true,
52     "editor.formatOnType": true
53 },
54 "html.autoClosingTags": true,
55 "html.format.indentInnerHtml": true,
56 "[sass]": {
57     "editor.formatOnSave": false,
58     "editor.formatOnPaste": true,
59     "editor.insertSpaces": true,
60     "editor.detectIndentation": true,
61     "editor.autoIndent": "full",
62     "editor.tabSize": 4,
63     "editor.quickSuggestions": {
64         "other": true,
65         "comments": false,
66         "strings": true
67     }
68 },
69 "[json]": {
70     "editor.defaultFormatter": "vscode.json-language-features",
71     "editor.formatOnPaste": true,
72     "editor.formatOnType": true,
73     "editor.tabSize": 4,
74     "editor.detectIndentation": false,
75     "editor.insertSpaces": false
76 },
77 "emmet.syntaxProfiles": {
78     "javascript": "jsx"
79 },
80 "emmet.includeLanguages": {
81     "javascript": "javascriptreact"
82 },
83 "files.associations": {
84     ".stylelintrc": "json",
85     ".prettierrc": "json"
86 },
87 "editor.wordWrapColumn": 80,
88 "editor.codeActionsOnSave": {
89     "source.fixAll.eslint": true,
90     "source.organizeImports": true
91 },
92 "editor.insertSpaces": false,
93 "editor.autoIndent": "full",

```

```

94  "editor.wordWrap": "on",
95  "editor.autoClosingBrackets": "always",
96  "editor.autoClosingQuotes": "always",
97  "editor.tabSize": 4,
98  "editor.tabCompletion": "on",
99  "editor.minimap.enabled": false,
100 "editor.quickSuggestionsDelay": 0,
101 "editor.snippetSuggestions": "top",
102 "editor.formatOnSave": true,
103 "editor.quickSuggestions": {
104     "other": true,
105     "comments": true,
106     "strings": true
107 }
108 }

```

Excerto de Código 9: Configurações utilizadas no Visual Studio Code

Nota

Para utilizar as Configurações apresentadas devem ser seguidos os passos abaixo:

1. Aceder às configurações do **Visual Studio Code** no formato **JSON**, para isso utilizar a tecla de atalho apresentada abaixo de acordo com o sistema operativo e pesquisar pela opção "Preferences: Open Settings (JSON)";
 - **No macOS:** CMD + SHIFT + P;
 - **No Windows/Linux:** CTRL + SHIFT + P.
2. Copiar as configurações apresentadas e colar no ficheiro **settings.json** (ficheiro que abriu no passo anterior).
 - **Nota:** caso já possua configurações neste ficheiro, basta remover as chavetas iniciais ({}) no código apresentado e colocar as restantes configurações.