Licenciatura
Multimédia

Imagem retirada do website
https://unsplash.com/photos/MuJHwDHbXUk

Programação III

Frederico Fonseca
fffonseca@ismt.pt

INSTITUTO SUPERIOR
MIGUEL TORGA

Sumário

- Introdução ao React
- Componentes

Texto de apoio:

Morgan, J. (2021). **How To Code in React.js** [E-book].

Consultado em https://www.digitalocean.com/community/books/how-to-code-in-react-js-ebook

React

- É uma biblioteca JavaScript (JS) declarativa, eficiente e flexível para a criação de interfaces gráficas (Facebook, 2011);
  - É usada para lidar com a **camada de visualização** (UI) para **aplicações web e mobile**, permitindo também criar componentes de UI reutilizáveis;
- É *open source*, utilizada para construir UI, nomeadamente para aplicações do tipo SPA (*Single-Page Application*);
- Faz a abstração do **DOM**, oferecendo um motor de manipulação mais simples - o chamado **Virtual DOM**;

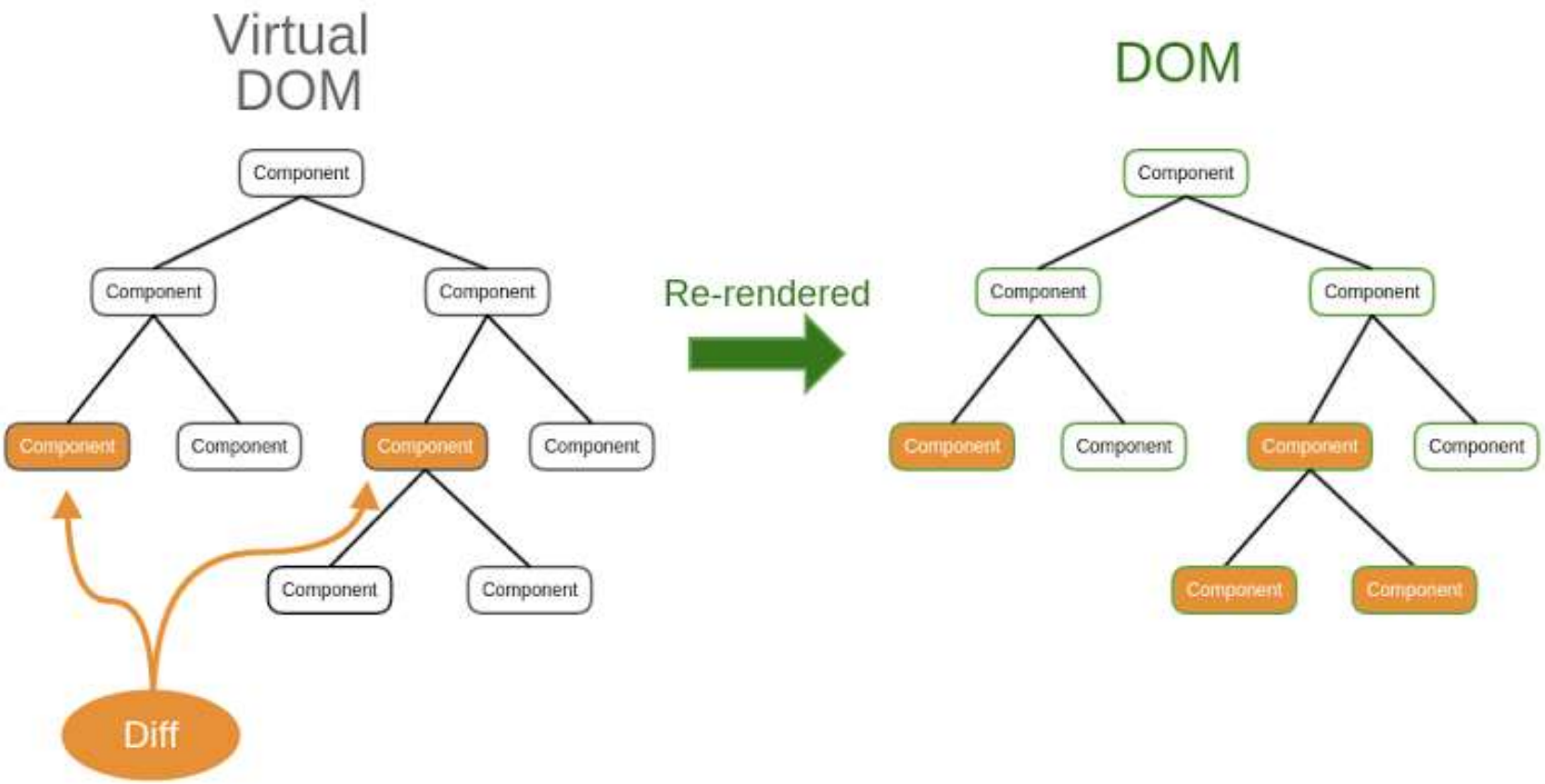
Compatibilidade

- Suporte para ES6 ou superior (ES6+);
  - ES6 também pode ser chamado de **ECMAScript 6** ou **ES2015**;
  - De uma forma muito resumida, esta versão da linguagem resolve problemas antigos do JS, bem como permite a construção de aplicações mais complexas de uma forma mais simples, prática e escalável;

Vantagens

- Algumas das **vantagens** do *react* são:
  - Curva de aprendizagem rápida;
  - Sintaxe amigável (ES6+ ou JSX);
  - Possibilidade de reaproveitamento de componentes (logo, de código);
  - Otimização da manipulação do DOM *(Virtual DOM)*, alterando somente o necessário;
  - Comunidade muito ativa;
  - Entre outros…

Virtual DOM

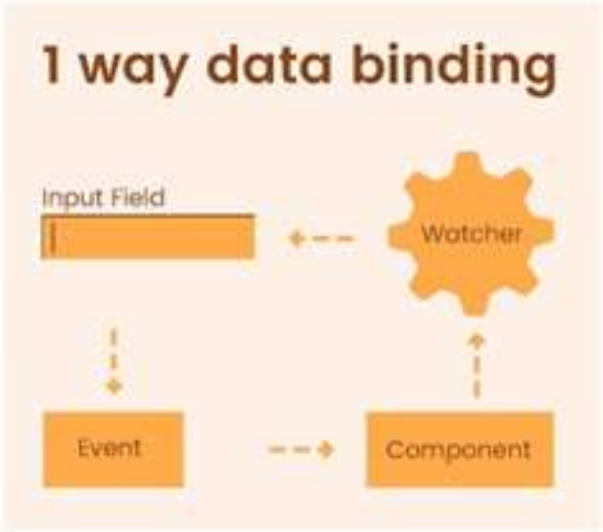

Virtual DOM
Component
Component
Component
Component
Component
Component
Component
Component
Component
Diff
Re-rendered
DOM
Component
Component
Component
Component
Component
Component
Component
Component
Component

Virtual DOM

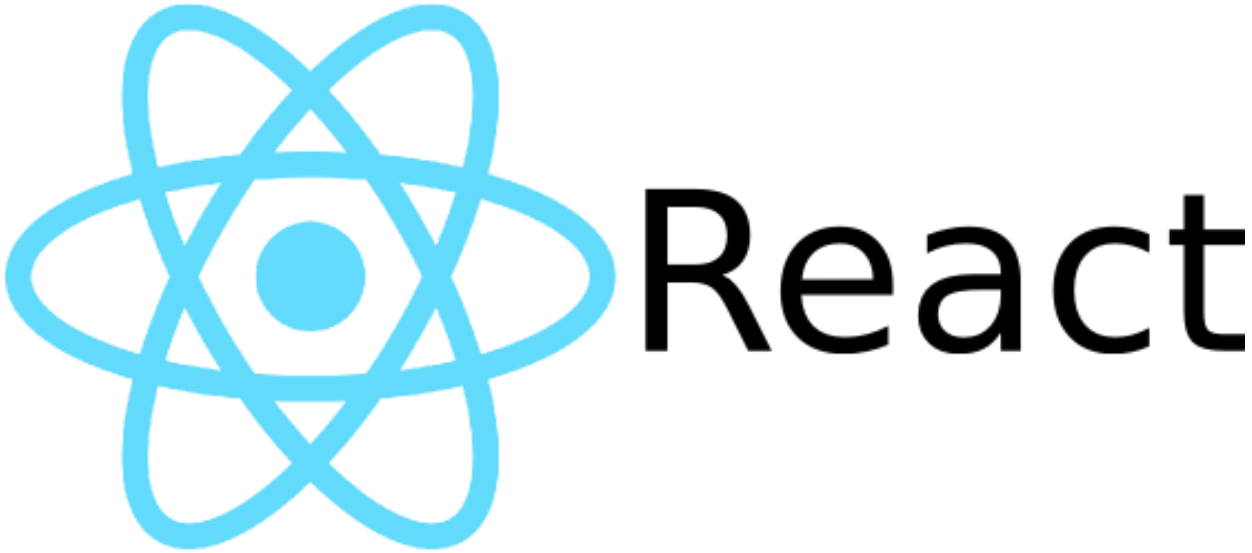
- O **Virtual DOM** tem a capacidade de processar seletivamente subárvores de nós após a sua alteração de estado;
  - É esta "virtualização" do DOM que permite a sua **renderização no servidor** e a utilização de *views* no lado do servidor;
- Faz a menor quantidade possível de manipulações DOM para manter os componentes atualizados;
- Implementa o **fluxo de dados reativo** de uma via (one-way), facilitando o *binding* dos dados;

Virtual DOM

- O chamado *one-way data binding*:
  - É necessário invocar a função *setState()* para alterar o valor;
  - Desta forma temos muito mais previsibilidade e segurança na nossa aplicação;



- Podemos criar aplicações eb onde se possa alterar elementos ou os dados exibidos, sem ser necessário recarregar a página;
  - Um exemplo são os **likes** do Facebook onde o número de *likes* aumenta e diminui sem fazer "refresh" à página;



Requisitos

- Podemos utilizar o React na construção das nossas aplicações de duas maneiras diferentes:
  - Diretamente no HTML (via CDN):
    - A chamada implementação 'stand-alone';

```
<script src="https://unpkg.com/react@16/umd/react.production.min.js"></script>
<script src="https://unpkg.com/react-dom@16/umd/react-dom.production.min.js"></script>
```

  - Utilizando o NPM e Node.js.

Requisitos

- Atualmente esta abordagem não é a mais recomendada;
  - Saltar para o slide seguinte;

- É possível utilizar o módulo NPM do **React** (create-react-app) para criar, de forma automática, uma aplicação em React;
- Para instalar o módulo executamos o comando:

```
npm install create-react-app -g
```

- Para criar o projeto executamos o comando:

```
create-react-app <nome do projeto>
```

(p. ex., `create-react-app reactapp`)

Requisitos

- Atualmente a solução mais recomendada recorre à biblioteca Vite em detrimento do NPM;

- Para criar o projeto executamos o comando:

```
npm create vite@latest <nome do projeto> --template react
```

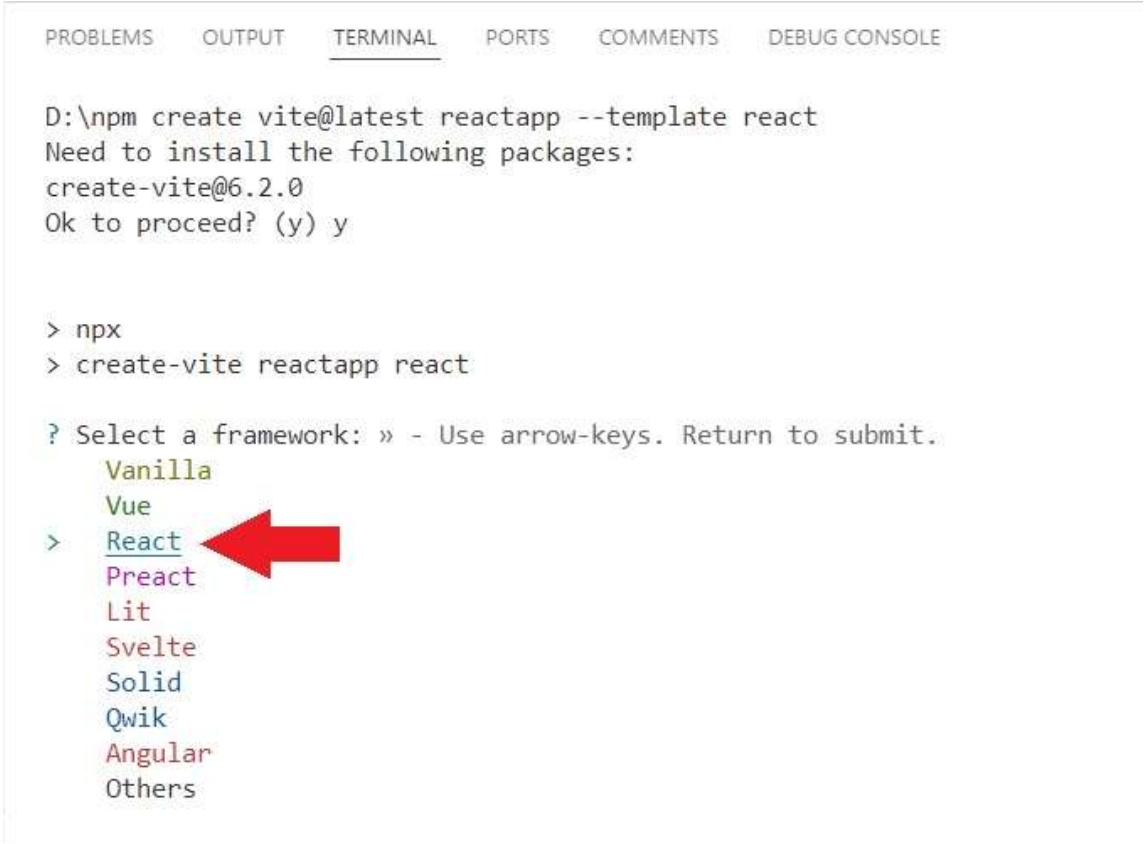
(p. ex., `npm create vite@latest reactapp --template react`)

Requisitos

- 1º Passo: executar o Vite com o template React

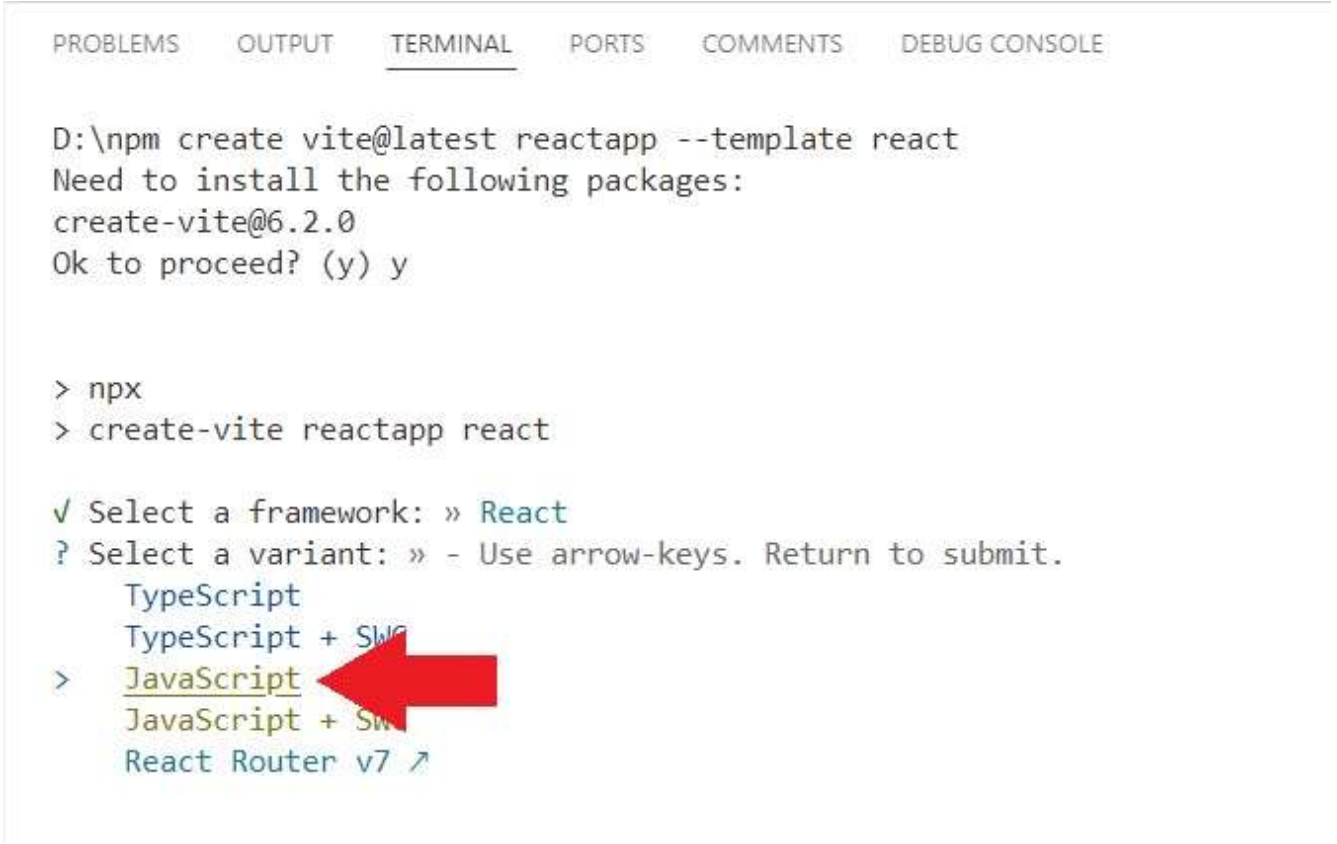
Requisitos

- 2º Passo: escolher a framework React *(utilize as setas do teclado)*



Requisitos

- 3º Passo: escolher a linguagem JavaScript

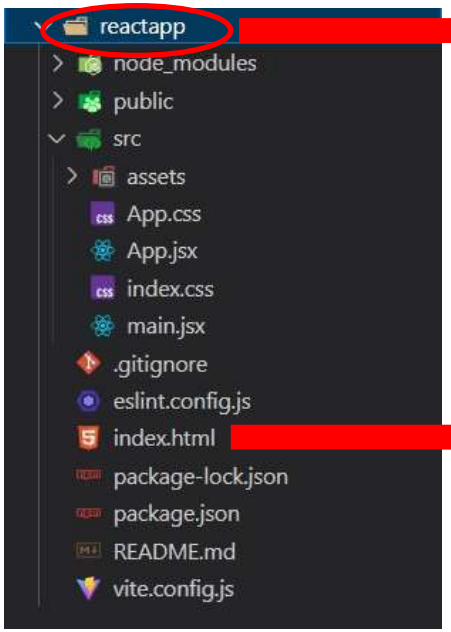


Requisitos

- 4º Passo: entre na diretoria e executa a instalação das dependências

Estrutura

- O Vite cria automaticamente as diretorias necessárias para o projeto em React, como é possível visualizar na figura seguinte:



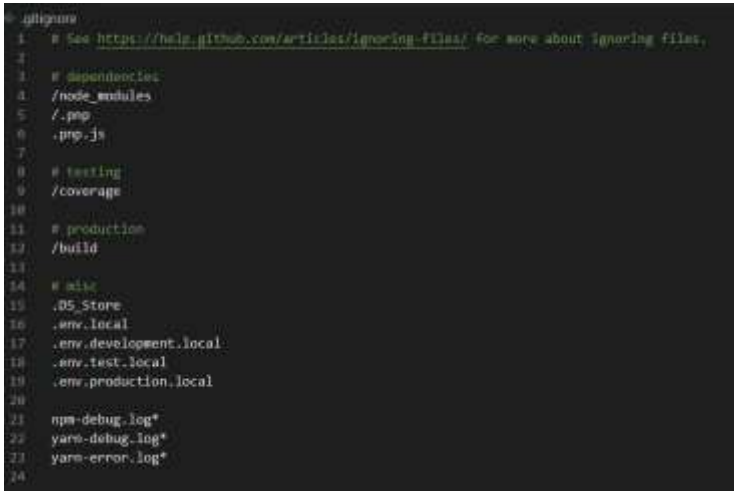
É a pasta principal do projeto React.

O ficheiro index.html é considerado como um ponto de entrada para a aplicação web.

Figura 1 – Diretorias e ficheiros do projeto React criado com o Vite

Estrutura

- '.gitignore' – é o ficheiro que permite identificar a origem GIT para controlar quais são os ficheiros e pastas que serão ignorados quando é feito o *commit*;
  - Embora o comando *create-react-app* crie o ficheiro, o mesmo processo não cria as pastas no repositório GIT;



Estrutura

- .package.json – este ficheiro contém as dependências e os scripts necessários para o projeto;

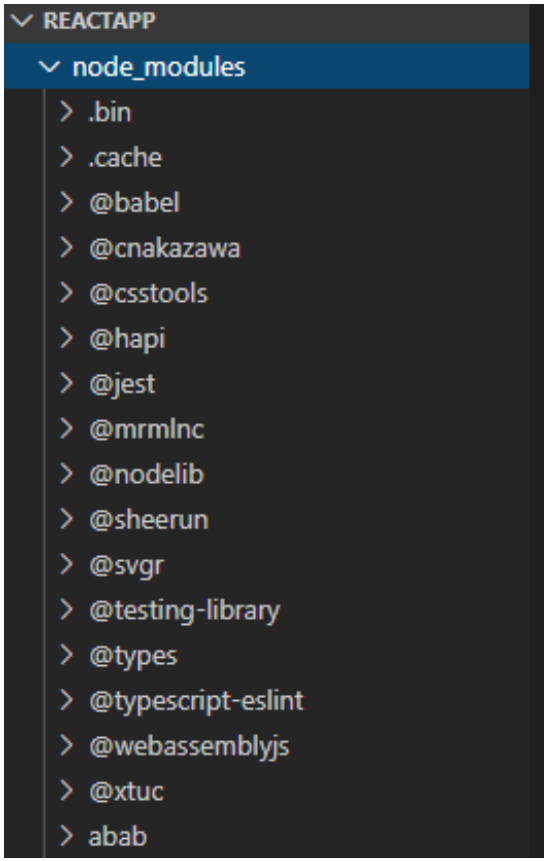
```
{
  "name": "reactapp",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^4.2.4",
    "@testing-library/react": "^9.4.0",
    "@testing-library/user-event": "^7.2.1",
    "react": "^16.12.0",
    "react-dom": "^16.12.0",
    "react-scripts": "3.3.1"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": "react-app"
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

Estrutura

- .package-lock.json - este ficheiro contém a árvore de dependência a ser instalada em *'/node_modules'*;
  - Mantém um histórico de alterações feitas no *package.json*;

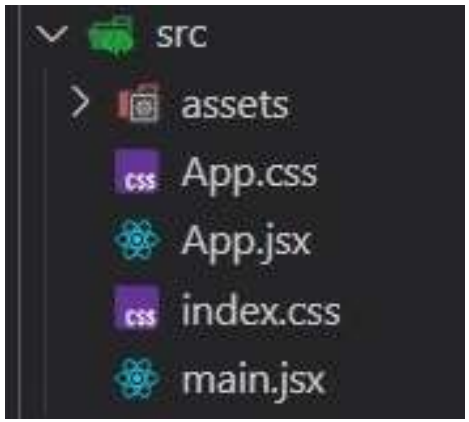
Estrutura

- .node_modules - esta pasta contém todas as dependências e sub-dependências especificadas no ficheiro '*package.json*', usadas pela aplicação React;
- Contém mais de 800 subpastas, sendo que essa pasta é adicionada automaticamente no ficheiro .gitignore;



Estrutura

- .src - Esta é a pasta principal da aplicação React. Contém os ficheiros JSX (ficheiros JS do React) e os respetivos ficheiros CSS;



- Nesta pasta existe a aplicação principal (main.jsx), juntamente com o seu estilo (index.css). O ficheiro App.jsx é a componente principal da aplicação, juntamente com o ficheiro de estilos (App.css).

Componentes

```
<KawaiiPlanet size="200" mood="blissful" color="#72CEFF" />
```

Componentes

- Um **componente** pode ser descrito como um "pedaço" de código isolado que pode ser reutilizado em vários módulos;
- A aplicação *React* contém um componente principal (normalmente a App), no qual outros sub-componentes podem ser incluídos;
  - Por exemplo, numa página, a aplicação pode ser subdividida em 3 partes, o *header*, o *main* e o *footer*;
  - Assim sendo, existe um único componente (app) com 3 sub-componentes (header, main e footer);

Componentes

- Existem **dois tipos** de componentes no React:
  - Componente funcional sem estado
    - *Stateless Funcional Component*
    - Alguns autores referem este componente como sendo de apresentação (*presentational*);
  - Componente de classe com estado
    - *Stateful Class Component*
    - E este componente como sendo de *container* (*stateful*);

Stateless Functional Component

- Os componentes deste tipo normalmente só se preocupam com a apresentação dos dados, portanto não tem estado;
- Estes componentes incluem **propriedades** imutáveis, ou seja, o valor das propriedades não pode ser alterado em execução;

- Este componente é composto por uma **função** vanilla JS simples:

```
function exemplo(props) {
  return <h1>Olá, {props.name}</h1>;
}
```

Exemplo

- Exemplo de um componente sem estado:

```
import React from 'react';

function exemplo() {
  return <h1>Olá Mundo!</h1>;
}
export default exemplo;
```

Exemplo 1a - Exemplo de um componente funcional sem estado utilizando sintaxe vanilla JS

```
import React from 'react';

const exemplo = () => <h1>Olá Mundo!</h1>;

export default exemplo;
```

Exemplo 1b - Exemplo de um componente funcional sem estado utilizando sintaxe JSX

Stateful Class Component

- Os componentes deste tipo são classes que estendem a classe Component da biblioteca React;
- Este componentes devem incluir obrigatoriamente o método render() que retorna os elementos HTML;

Exemplo 2 - Exemplo de um componente de classe com estado utilizando sintaxe JS.