Linguagens Script

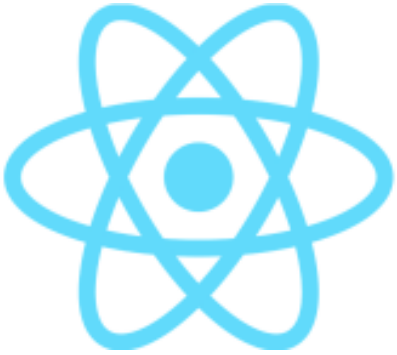
<React>

Departamento de Engenharia Informática e de Sistemas

Cristiana Areias < cris@isec.pt >

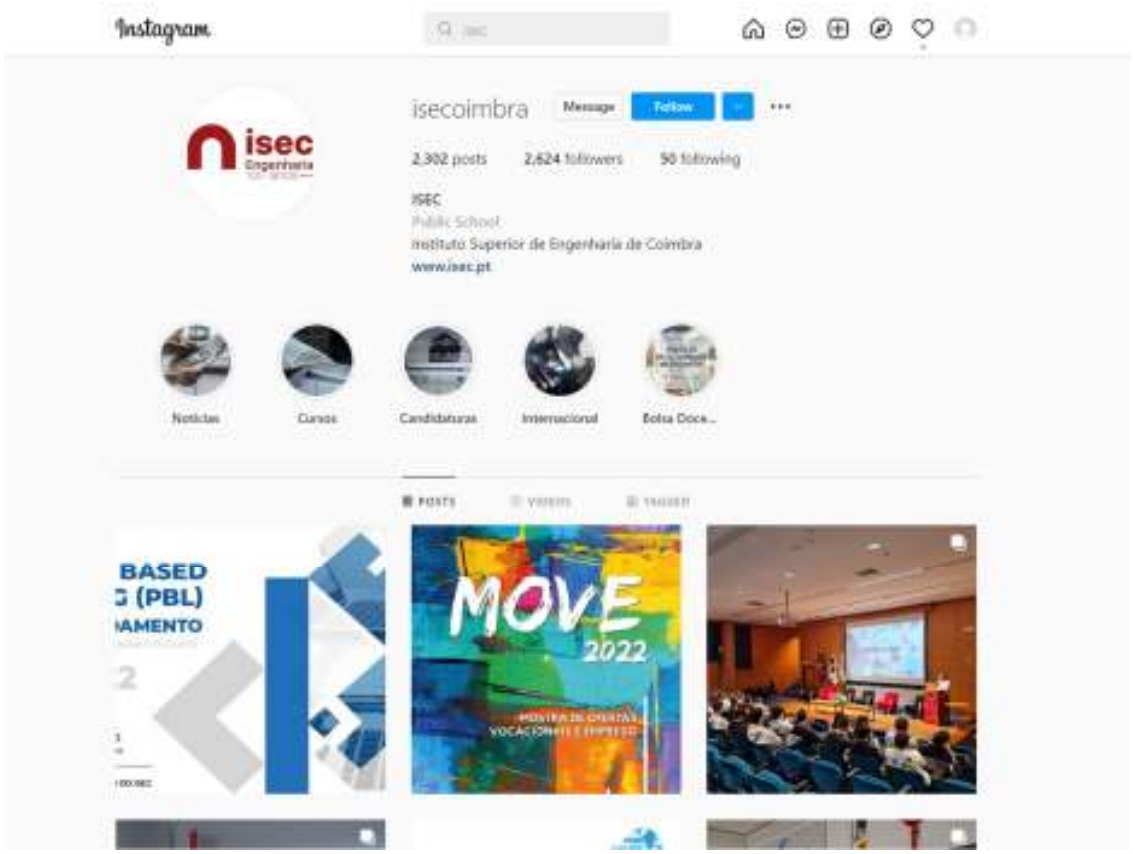
2022/2023

Porquê ReactJS?

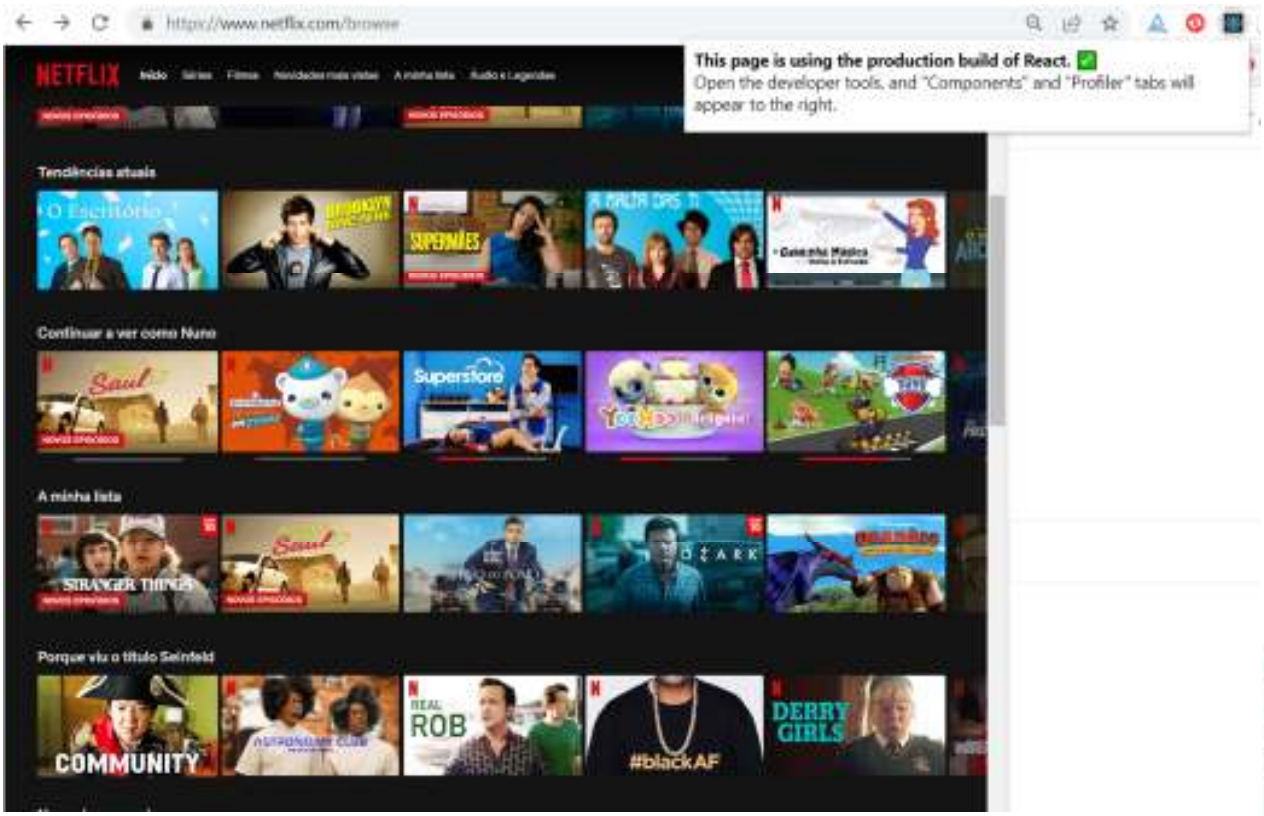


Aplicações React > Facebook

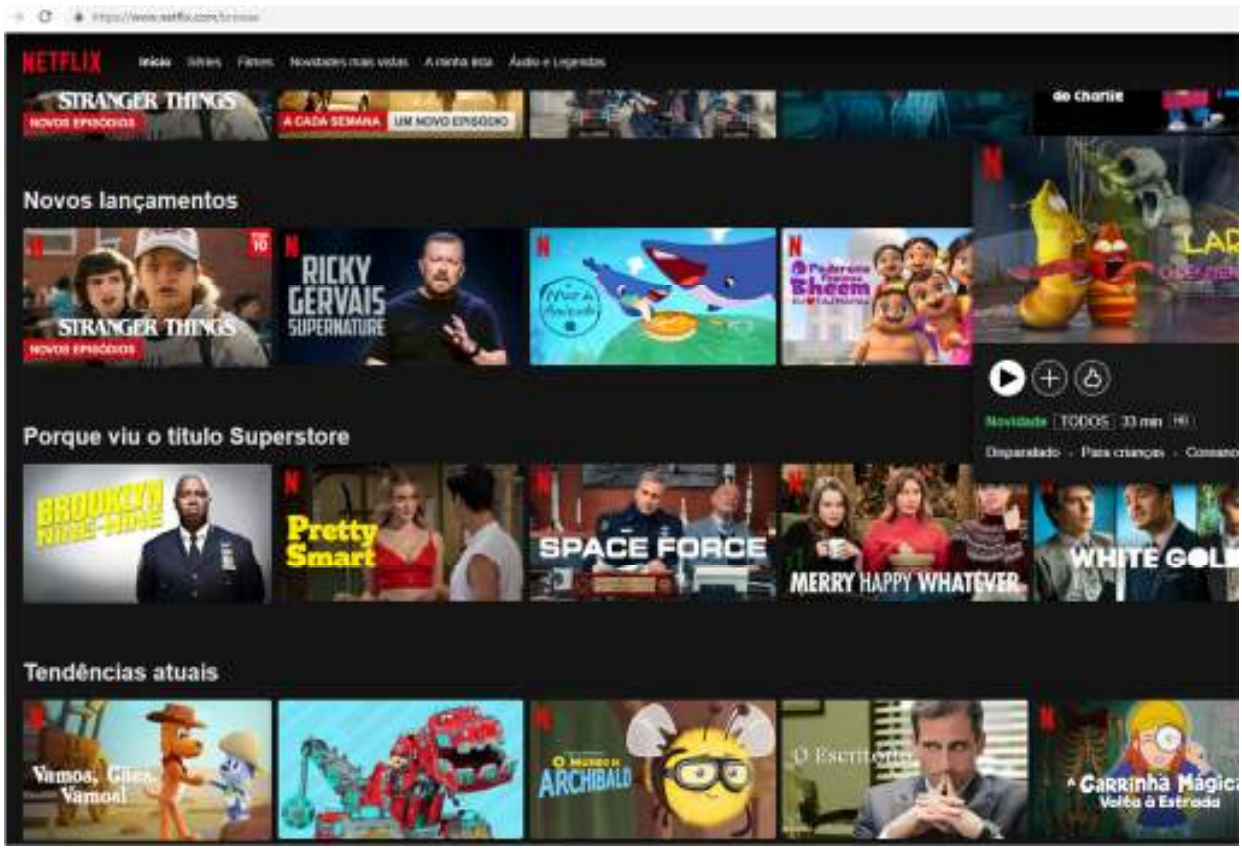
Aplicações React > Instagram



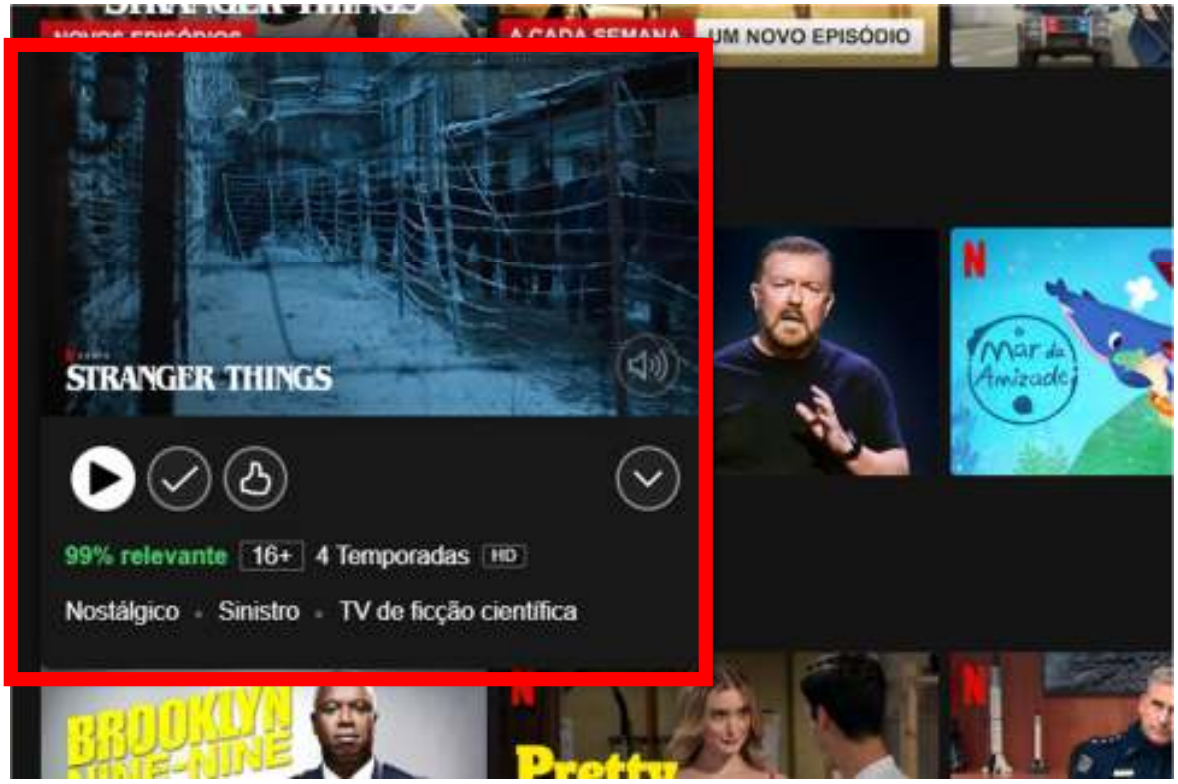
Aplicações React > NETFLIX



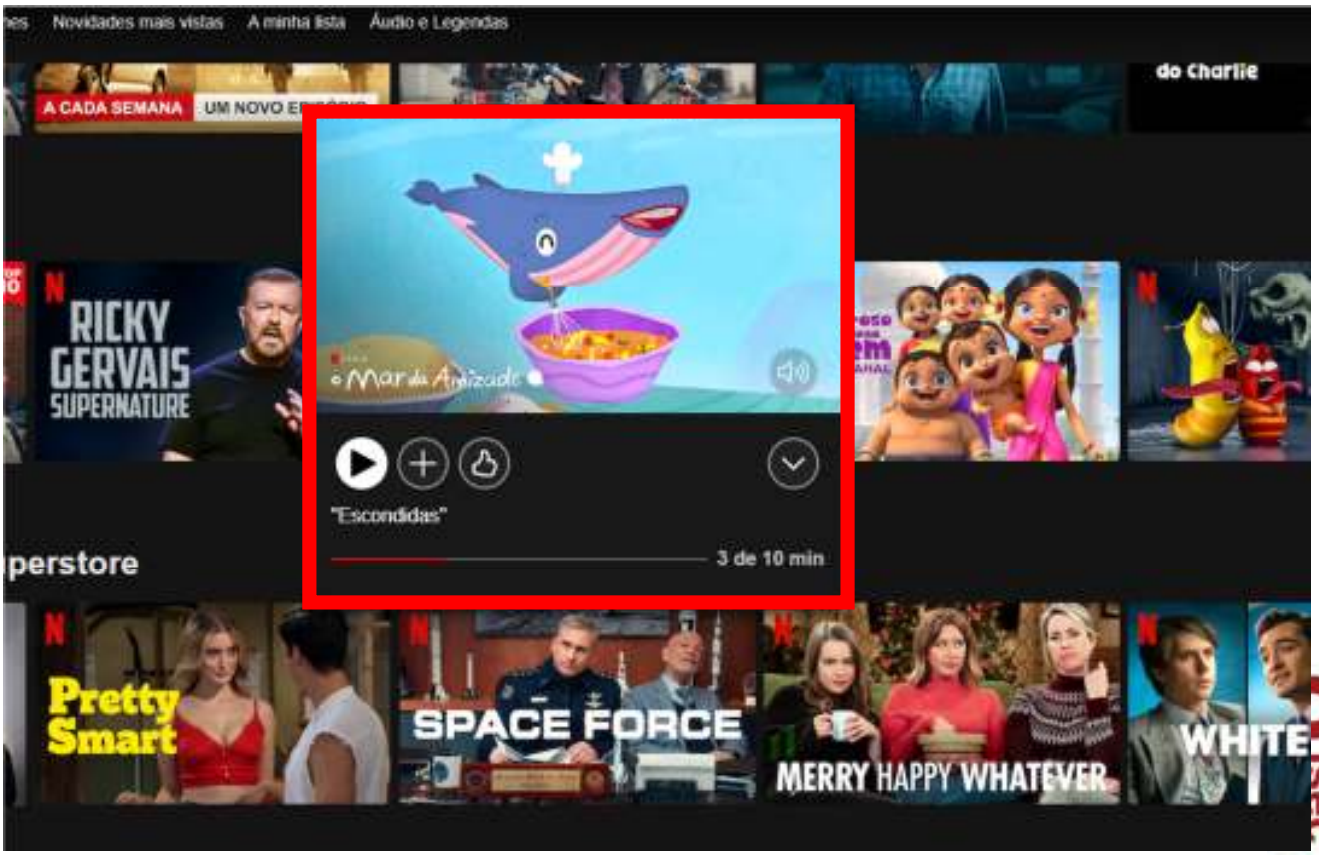
Aplicações React > NETFLIX



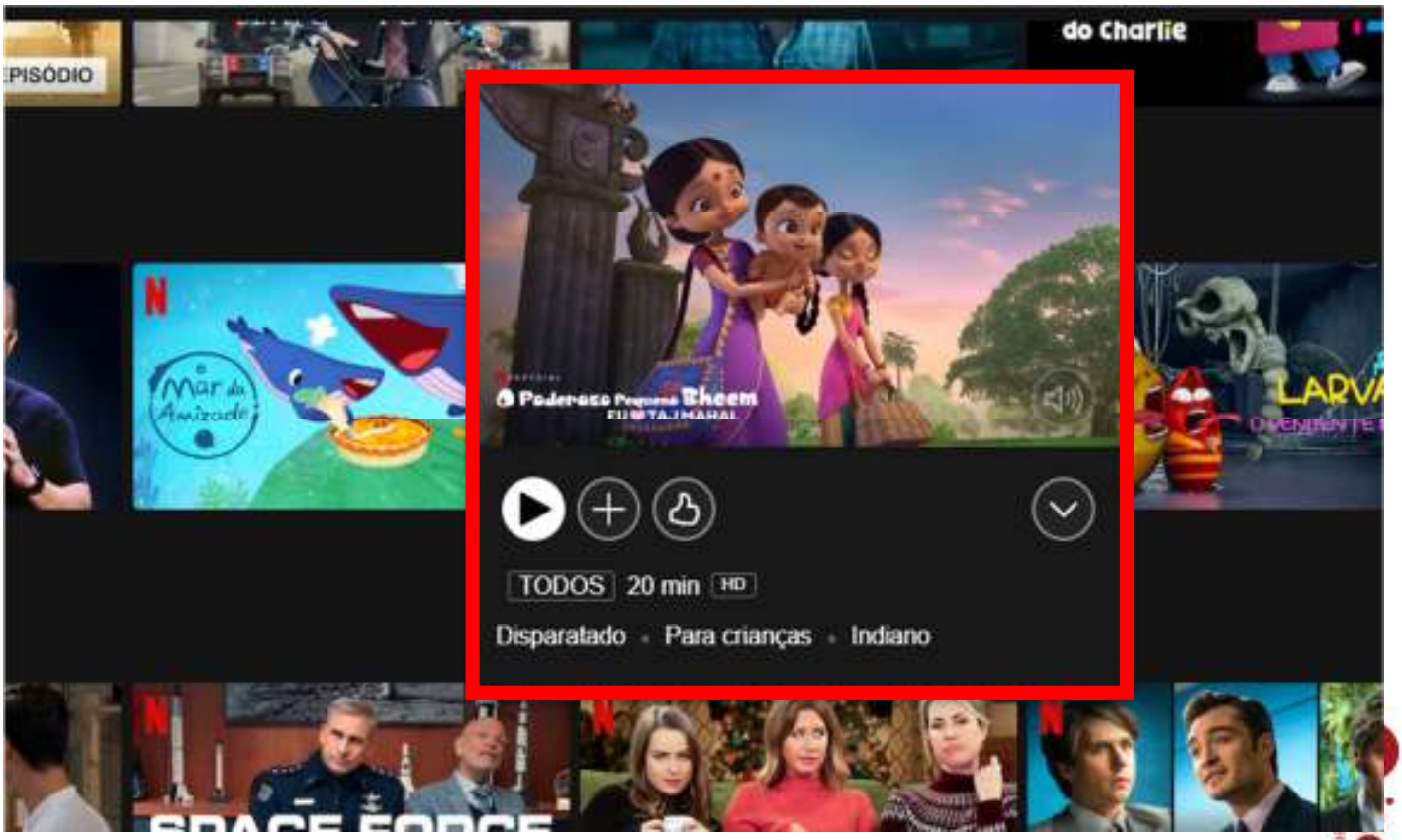
Aplicações React > NETFLIX



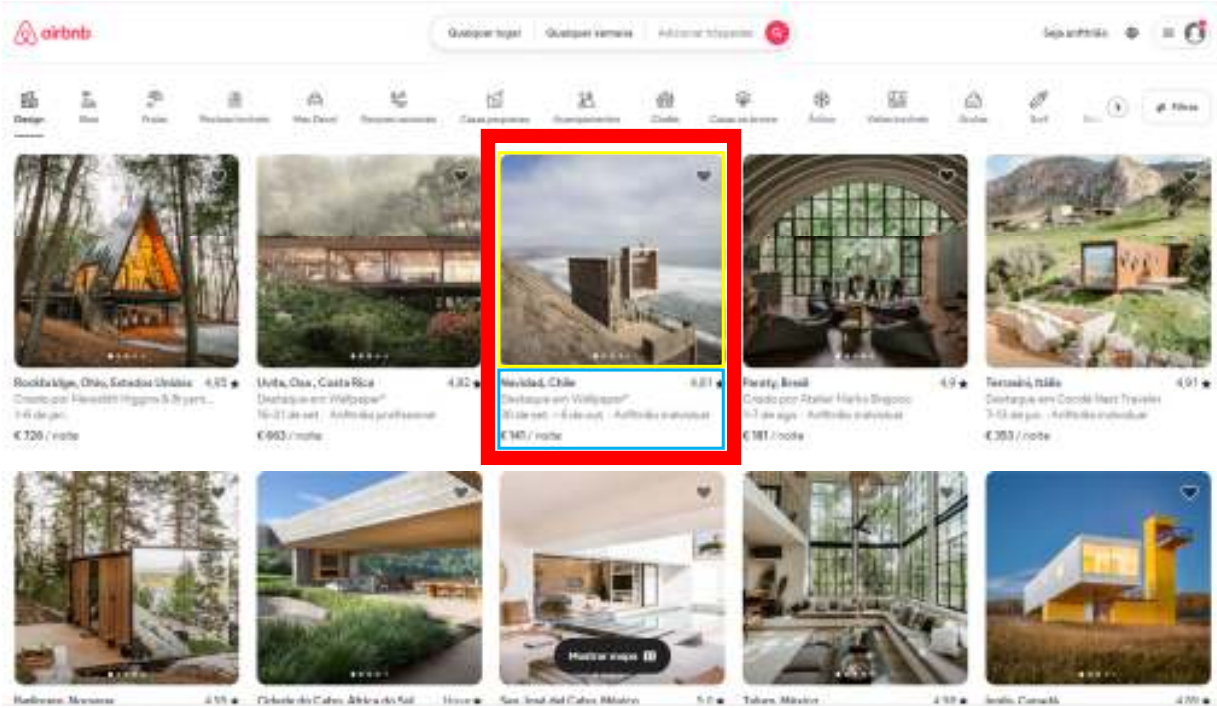
Aplicações React > NETFLIX



Aplicações React > NETFLIX



Aplicações React > Airbnb



Introdução ao ReactJS?

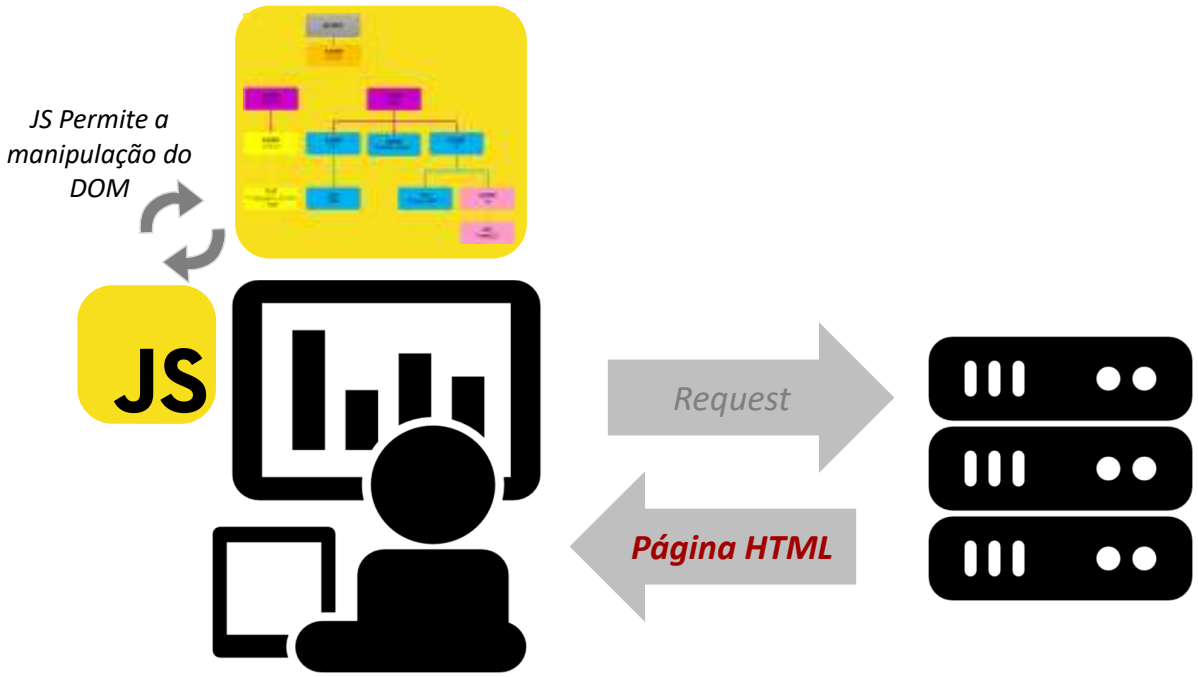
Multi-Page Application (MPA)

- Forma "clássica" :
  - uma alteração de conteúdo implica um novo request ao servidor e envio de uma nova página para ser interpretada pelo browser;
- Cada reload de uma nova página, implica um tempo de espera, o que reduz fluidez e empobrece a experiência de utilização;
- Navegação sólida entre menus;
- Favorece o SEO
  - O conteúdo encontra-se todo disponivel o que permite optimizar a visibilidade e consequente indexação pelos motores de busca;
- Desvantagens
  - Menor fluidez (tempo de espera necessário para o load de uma nova página);
  - Tipicamente existe um acoplamento elevado entre o *front end* e o *back end*;

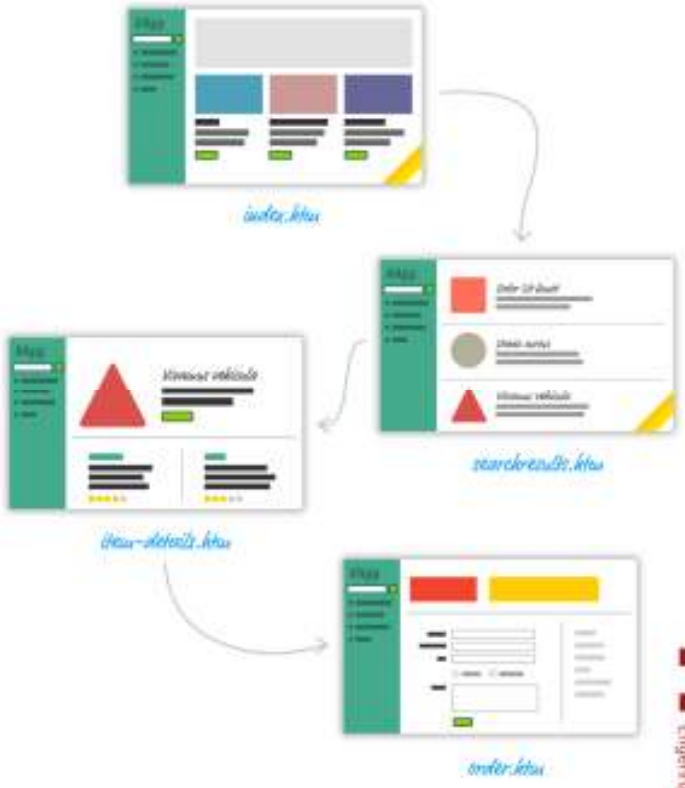
Single Page Application (SPA)

- Um script controla localmente a gestão de conteúdo o que elimina a necessidade de reload completo da da página
  - É feito o download dos recursos HTML+CSS+JS apenas uma vez durante todo o ciclo de interação com o utilizador;
  - Durante o ciclo existe apenas necessidade de transmissão de dados entre cliente e servidor;
  - Os pedidos de markup e dados são totalmente independentes.
- Melhora a experiência do utilizador (maior fluidez, reduz o tempo de espera)
- Os dados podem ser armazenados localmente permitindo utilização offline da aplicação.
- Desvantagem
  - Como o conteúdo não está todo disponivel ao mesmo tempo, pode dificultar o Search Engine Optimization (SEO)

Multi Page Application

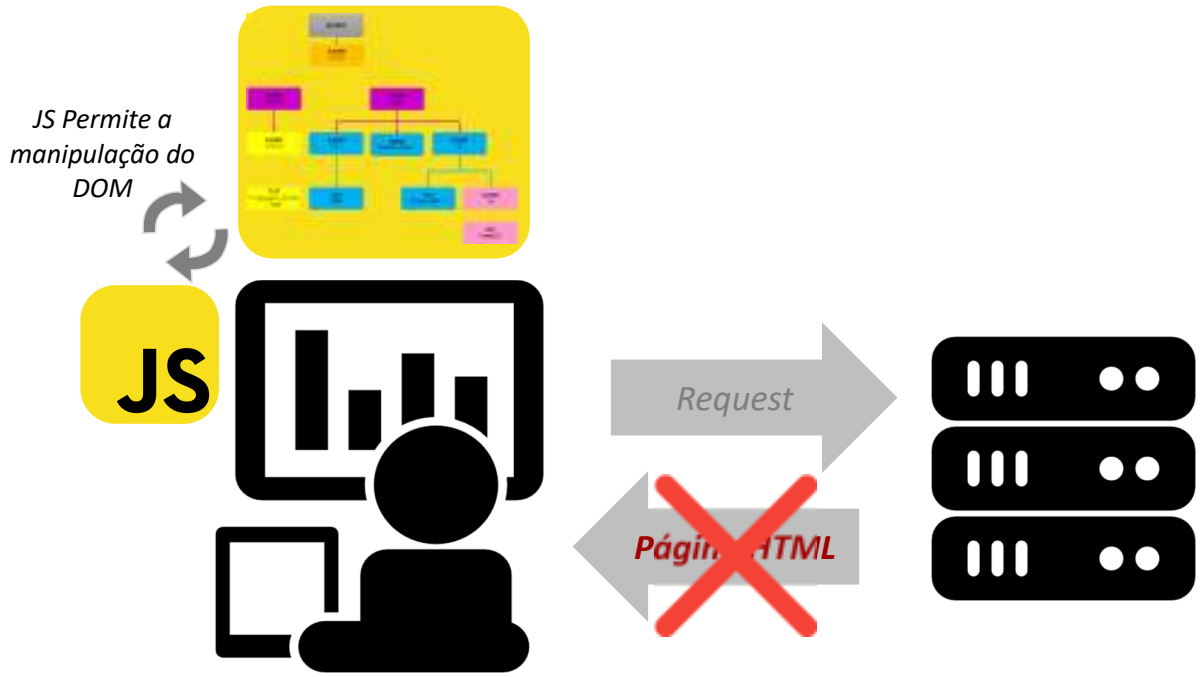


Multi Page Applications (MPA)

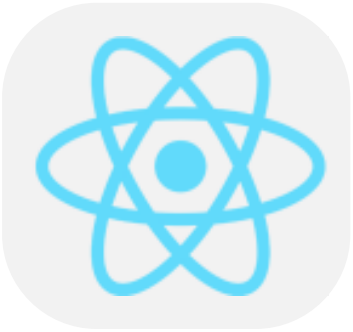


https://www.kirupa.com/react/

Single Page Applications



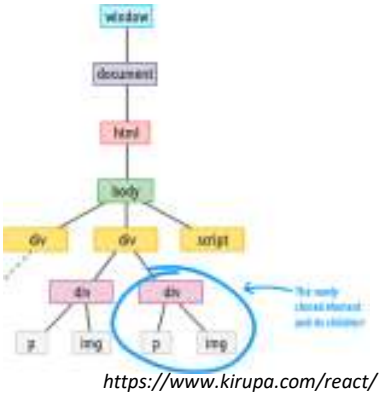
Single Page Application (SPA)



https://www.kirupa.com/react/

SPA vs MPA > Desafios

- Multi-page application
  - O page reload impede a manutenção de elementos, os anteriores são destruídos e é novamente construída na sua totalidade uma nova página
- Single Page Application
  - Dados devem ser sincronizados com o interface com o utilizador (UI)
    - Se é feito o carregamento de novo conteúdo, quais são os elementos a manter? quais é que vão ser eliminados?
    - Esta gestão pode tornar-se confusa
  - Manipulação do DOM é mais lenta
    - adicionar elementos, alterar, eliminar ….



https://www.kirupa.com/react/

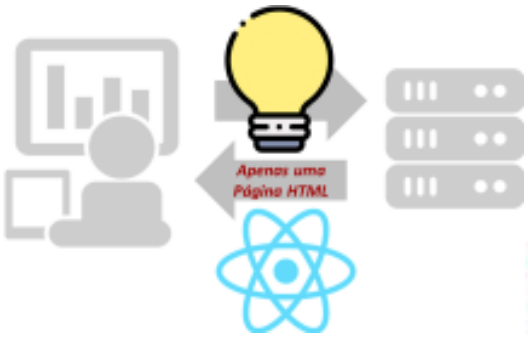
O que é ReactJS?

O que é React

- **Biblioteca JS para criação de UI** *(client-side)*;
  - Dá libertar ao programador, no qual tem responsabilidade da estrutura do projecto, instalação de outras bibliotecas (de terceiros) para incluir determinadas funcionalidades, como HTTP routing, entre outros.
- Utilizado para construção **de SPAs** *(Single Page Applications);*
- Segue um estilo de programação declarativa e uma abordagem baseada em **componentes**;
- Permite a criação de aplicações complexas com grande flexibilidade e velocidade:
- Recorre ao JSX para renderização HTML;
- Aplica o **Virtual DOM** para realizar as alterações do DOM;
- Não é framework (como o Angular ou o Vue);
- Projecto open-source criado pelo Facebook;
- Corresponde à camada **view**, numa aplicação que recorre ao padrão MVC (*Model View Controller*);

React > Vantagens

- Melhora o Controlo do UI (elementos visíveis, conteúdo, …)
- Melhora a manipulação do DOM (fluidez, …)
  - O Virtual DOM permite a renderização do lado do servidor mais rápida;
- Popularidade, recursos disponiveis, comunidade, ….
- Permite uma abordagem declarativa, focada em componentes com estados próprios;
- Arquitectura simplificada;
- Ambiente intiutivo;
- JSX – *markup* dinâmico;
- Desempenho e facilidade de implementação de testes.

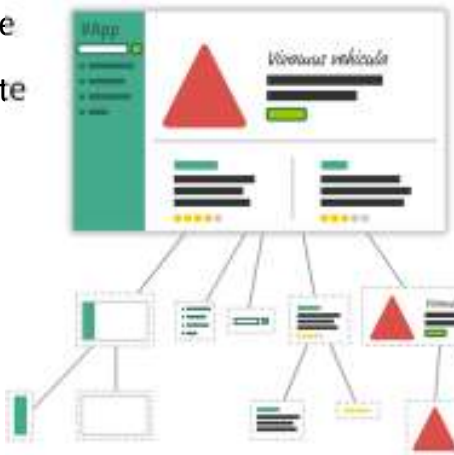


React > Características

- Gestão (automática) do estado do Interface com o utilizador
  - Apenas tem de ser definido o estado final do UI
  - O React implementa esta gestão de forma automática e transparente para o utilizador.
- Manipulação do DOM
  - React implementa o **conceito de Virtual DOM** o que acelera as alterações no DOM
  - A alteração/modificação do **Virtual DOM** é extremamente rápida:
    - React compara o "virtual DOM" com o "real DOM", identificando as alterações que é necessário fazer, minimizando dessa forma o número de alterações do DOM. **(reconciliation process)**
      - Os elementos iguais entre o "virtual DOM" e o "real DOM" permanecem inalterados sem necessidade de qualquer ação adicional.

React > Características

- Interface com o Utilizador : **Combinação de Componentes**
  - O React promove a divisão dos componentes em componentes mais pequenos até se atingir um nível elementar (componente que não faz sentido dividir).
  - Assim, é incentivada a criação de componentes cada vez menores (elementares) que posteriormente são combinados para criar elementos visuais mais complexos que por sua vez são combinados para criar a totalidade do UI.



React > Características

- Elementos gráficos definidos exclusivamente em Javascript
  - JSX (*JavaScript Syntax Extension*)
    - Sintaxe semelhante ao HTML
    - Apesar das semelhanças o JSX possui um conjunto de caracteristicas próprias do JavaScript

```
export default class Componente
extends React.Component {
  render() {
    return (
      <header>
        <h1>Linguagens Script</h1>
        <h2>JavaScript | React </h2>
      </header>
    );
  }
}
```

```
export const Componente = () => {

  return (
    <header>
      <h1>Linguagens Script</h1>
      <h2>JavaScript | React </h2>
    </header>
  );

}
```

```
<Componente title="LS" />
```

React

Instalação e Setup

Linguagens Script @ LEI / LEI-PL / LEI-CE

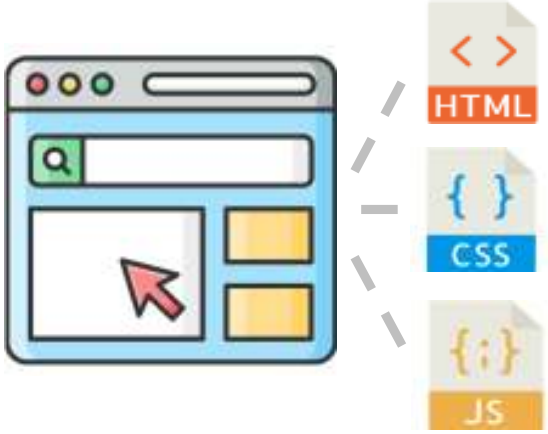
Departamento de Engenharia Informática e de Sistemas

Cristiana Areias < cris@isec.pt >

2022/2023

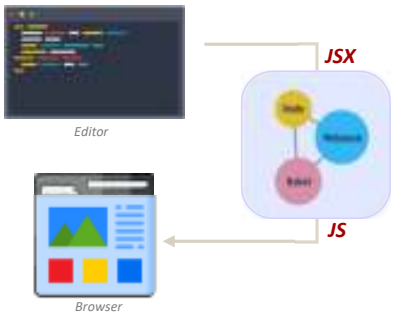
React (setup)

- O browser, independentemente da tecnologia aplicada para criação da aplicação, consegue interpretar apenas HTML, CSS e JavaScript
  - Para uma React app ser interpretada pelo browser é necessário efetuar a sua tradução para as tecnologias que o browser consegue interpretar (p.ex: JSX -> JS)



Instalação / Projeto

Transpiling direto no browser do código JSX



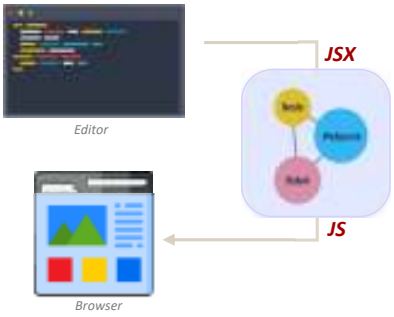
Ambiente de Desenvolvimento responsável pelo transpiling do JSX

Instalação / Projeto

Transpiling direto no browser do código JSX

- Desvantagens:
  - Baixa Performance.
  - Para além de todas as tarefas que o browser é responsável tem também de converter o JSX para JavaScript. O que pode consumir muito tempo.
  - É aceitável em projetos muito limitados mas em contexto real não é aceitável.

Instalação / Projeto



Ambiente de Desenvolvimento responsável pelo transpiling do JSX

- Vantagens
  - Elevada performance (muito mais eficiente);
  - A app final incorpora a conversão JSX-JS;
  - O browser é responsável apenas pela interpretação do JS (geralmente já optimizado);
  - Aplicado em contexto real
- Desvantagens:
  - maior complexidade na configuração do ambiente de desenvolvimento

React (*setup*)

- **Ficheiro HTML estático:**
  - O browser converte automaticamente para JavaScript em *runtime*
    - Esta opção é mais limitada e muito menos eficaz do que a opção seguinte;
    - A única vantagem é que não exige a instalação do ambiente de desenvolvimento.
- **create-react-app**
  - Instalar um ambiente de desenvolvimento que envolve um conjunto de ferramentas
    - Sempre que é feita uma *build* da aplicação, o JSX é automaticamente convertido para JS, o qual funciona como um ficheiro de JavaScript puro (*plain JS*).
    - Esta é uma das opções mais correta, mais eficaz e utilizada nas aplicações modernas e de elevada dimensão / complexidade (explorar mais tarde)

React (setup)

- Ficheiro HTML estático

```
<script src="https://unpkg.com/react@17/umd/react.development.js" crossorigin></script> *
<script src="https://unpkg.com/react-dom@17/umd/react-dom.development.js" crossorigin></scri**
<script src="https://unpkg.com/@babel/standalone/babel.min.js"></script> ***
```

* Ligação CDN que permite a utilização do React, fazendo a ligação à API do React;

** ReactDOM é uma biblioteca que permite que os interfaces implementados em React, possam funcionar diretamente no browser.

*** Babel é necessário para fazer o tanspiling (converter código fonte numa linguagem (JSX) para código fonte em outra linguagem (JS)

React (setup)


```
<script src="https://unpkg.com/react@17/umd/react.development.js" crossorigin></script>
<script src="https://unpkg.com/react-dom@17/umd/react-dom.development.js" crossorigin></script>
<script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
```

- O valor do atributo **type** tem que ser estabelecido como "text/babel" para pemitir o *transpiling* do código

```
<script type="text/babel">
   const containerRoot = document.getElementById("root");
   ReactDOM.render(<React.StrictMode>
                      <h1> Linguagens Script </h1>
                   </React.StrictMode>
   ,containerRoot);
</script>
```

ReactDOM.render()

O método tem dois argumentos:

- O JSX a ser disponibilizado
- A localização no DOM onde vai ser feito o render do JSX

Existem outras bilbiotecas para utilizar interfaces React em outros ambientes (ReactDomServer, ReactNative, ….)

```
<body>
  <div id="root"></div>
  <script type="text/babel" src="js/index.js"></script>
</body>
```

A localização onde o ReactDOM.render() vai renderizar os conteúdos pode ser facilmente alterada para qualquer localização na DOM Tree.

Create-React-App

- Uma das formas mais populares;
- *Boilerplate* criado pela equipa do facebook que inclui as principais ferramentas para usar o React, tais como:
  - Webpack;
  - Babel;
  - Eslink;
  - Entre outros elementos;

Boilerplate *– Templates de código que podem ser usados para construção de aplicações e que necessitam de pouca ou nenhuma alteração;*

"Create React App is a new officially supported way to create single-page React applications. It offers a modern build setup with no configuration.."

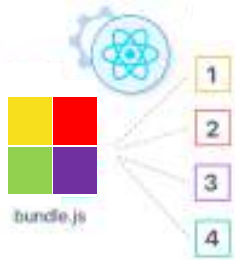
https://reactjs.org/blog/2016/07/22/create-apps-with-no-configuration.html

- **Outras possibilidades:**
  - Vite
  - Snowpack

Create-React-App > Instalação

- *Node.js runtime environment*
  - Aplicações full stack JavaScript;
  - O Node Package Manager (**npm**) é necessário para instalar as dependências de que o React precisa o que vai facilitar todo o processo de configuração/instalação;
  - **npm** (Node.js package manager) é composto por:
    - Linha de commando (**npm client**) e por uma base de dados com packages públicos e privados (***npm registry***).
      - O ***npm registry*** é acedido via ***npm client***
    - Os packages disponíveis podem ser pesquisados no npm website https://www.npmjs.com/

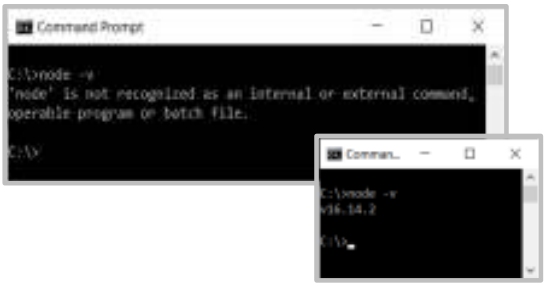
Create-React-App > Instalação



- *Bundler*
  - Constrói os ficheiros a serem interpretados pelo browser
- *Babel*
  - Pré-processador de JS que permite incorporar todas os desenvolvimentos mais recentes do JS;
  - Conversão automática para JS ES5 o que garante a compatibilidade com todos os browsers;
- *Development Server* (DS)
  - Servidor que assegura o funcionamento da app quando efetuado um request a um servidor web.
  - Servidor local que reproduz de forma correta a operação da app em condições reais (request a um servidor web remoto).

Create-React-App

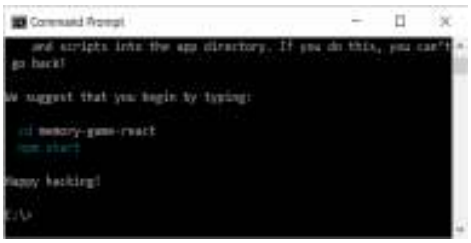
- Instalação do node



- Executar os commandos

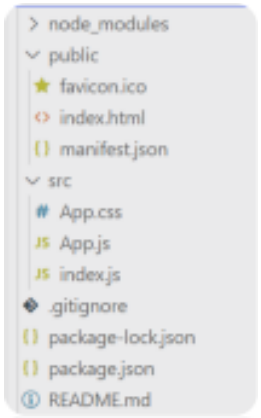
```
>> npm install create-react-app
>> create-react-app  ProjectName
>> npm start
```

__Nota:__ npm start deve ser executado na pasta do projeto



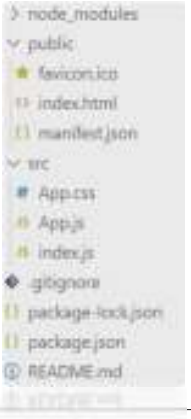
Estrutura de Ficheiros

- **Pasta node_modules**
  - Pasta é onde o npm armazena localmente todos os pacotes instalados;
  - Não se precisa efetuar alterações a esta pasta.
- **index.html**
  - Ficheiro principal o qual é apresentado no browser;
  - Inclui todos os metadados que descrevem o conteúdo da página.
- **Pasta src**
  - Inclui o código fonte.
  - Aqui pode-se incluir todos os componentes em react, módulos, ficheiros de estilização, entre outros.
- **index.js**
  - ponto de entrada para a aplicação React. O método render é invocado para renderizar um elemento React no DOM, e retorna a referencia do componente. Neste caso, o elemento React é o componente App.
- **App.js**
  - o componente principal (root) da aplicação, o qual irá importar outros componentes, filhos.



Estrutura de Ficheiros

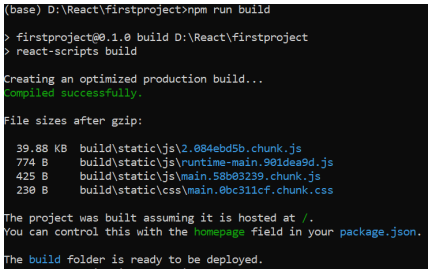
- **manifest.json**
  - Descreve toda a aplicação incluindo a informação necessária para os icons, cores entre outros.
- **.gitignore**
  - ficheiro que descreve quais diretorias serão ignoradas, no contexto do controlo de versões git, como por exemplo, o diretório node_modules será ignorado. Não é obrigatório usar o git no contexto das aulas, mas não será removido para quem desejar explorar o seu uso.
- **package.json**
  - inclui todos os pacotes que o projeto depende, e qual a versão que o projeto pode usar. Inclui ainda um conjunto de script do create_react_app, que permitem, por exemplo, executar a aplicação com npm start.
- **package-lock.json**
  - inclui a versão exata os pacotes utilizados no projeto.
- **README.md**
  - inclui informação geral do projeto, a descrição, como instalar e executar.
- **App.css**
  - Contém estilos CSS que estão a ser usados no ficheiro App.js.



Projeto React > Build

- App's são implementadas em development mode:
  - Código não otpimizado de forma a facilitar o debug;
  - Quando é altura de disponibilizar a aplicação aos utilizadores reais, o objetivo é criar a solução mais compacta possível e que possibilite uma execução o mais rápida possível

>> npm run build



  - Quando o build da applicação é concluido, a app pode ser disponibilizada num servidor web (deployment)
    - O resultado final é um ficheiro HTML, um ficheiro CSS e um ficheiro único *.js, o qual contém toda a lógica da aplicação .