

NextJS

Introdução

1 Introdução

A página oficial do NextJS, que pode ser encontrada a seguir, o define como um "Framework React para produção".

<https://nextjs.org/>

Vale destacar os seguintes pontos da documentação.

- Hybrid static & server rendering
- TypeScript support
- Smart bundling
- Route pre-fetching

Como veremos, o NextJS é um arcabouço que permite a criação de uma aplicação React com interfaces gráficas, sendo, portanto, um arcabouço **Front End**. Veremos também que ele realiza renderização do lado do servidor e define endpoints em cuja implementação podemos implementar regras de negócio, acessar uma base de dados, tal qual uma aplicação **Back End**. O NextJS é, portanto, considerado um arcabouço **Full Stack**.

1.1 (Server-Side Rendering) Considere uma aplicação React comum. Se inspecionar seu código fonte, você deverá se deparar algo assim.

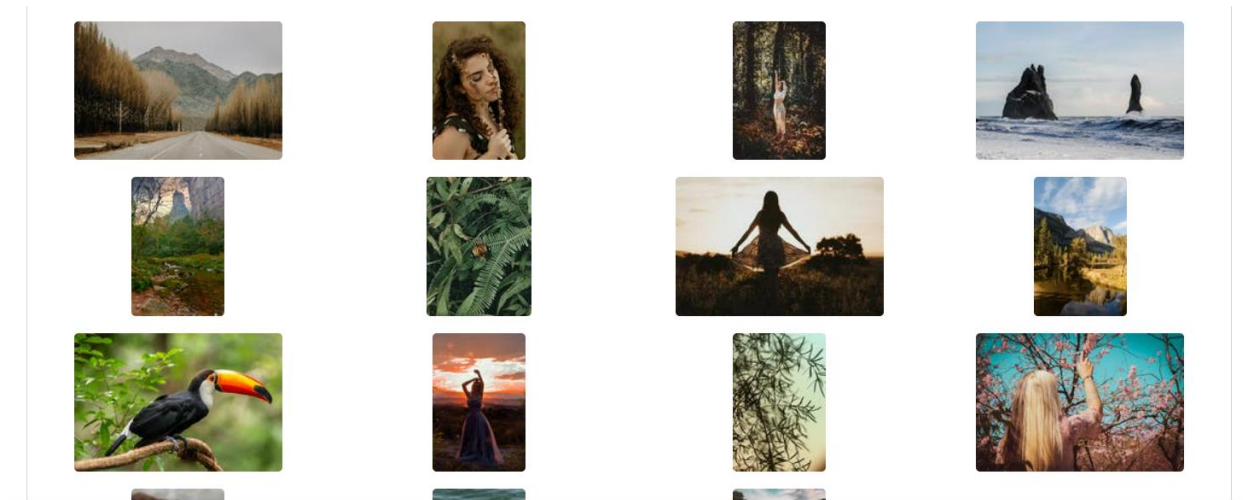
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="/pessoal-maua-busca-usuario-maratona-frontend/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="/pessoal-maua-busca-usuario-maratona-frontend/logo192.png" />
    <!--
      manifest.json provides metadata used when your web app is installed on a
      user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/
    -->
    <link rel="manifest" href="/pessoal-maua-busca-usuario-maratona-frontend/manifest.json" />
    <!--
      Notice the use of /pessoal-maua-busca-usuario-maratona-frontend in the tags above.
      It will be replaced with the URL of the 'public' folder during the build.
      Only files inside the 'public' folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "/pessoal-maua-busca-usuario-maratona-frontend/favicon.ico" will
      work correctly both with client-side routing and a non-root public URL.
      Learn how to configure a non-root public URL by running `npm run build`.
    -->
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.
      To create a production bundle, use `npm run build` or `yarn build`.
    -->
    <script src="/pessoal-maua-busca-usuario-maratona-frontend/static/js/bundle.js"></script><script src="/pessoal-maua-busca-usuario-maratona-frontend/static/js/vendors-main.chunk.js"></script>
  </body>
</html>
```

Temos praticamente uma única div com o famigerado id igual a "root". Além disso, temos diversos arquivos Javascript que o navegador obtém do servidor quando o usuário visita a página. Neste cenário, como se dá a construção da página? O código Javascript é responsável por construir e ajustar a árvore DOM a cada interação do usuário. Um ponto fundamental a se observar é o seguinte: esse código Javascript roda do lado do cliente. Ou seja, no navegador do usuário. Evidentemente, isso é muito natural. E tem algumas **consequências evidentes**. Considere a seguinte página de exemplo.



(Tempo de espera) A exibição do conteúdo da página depende da disponibilidade das figuras. Se elas forem obtidas por meio de requisições HTTP, o usuário provavelmente terá de esperar alguns milissegundos até que possa vê-las.

(Mecanismos de busca não verão o conteúdo completo) Além disso, considere o que acontece quando os mecanismos de busca visitam a nossa página para indexá-la. Como nossa árvore DOM somente é construída do lado do cliente, os mecanismos de busca deixarão de ver parte de seu conteúdo. Possivelmente verão apenas a div de id igual a root.

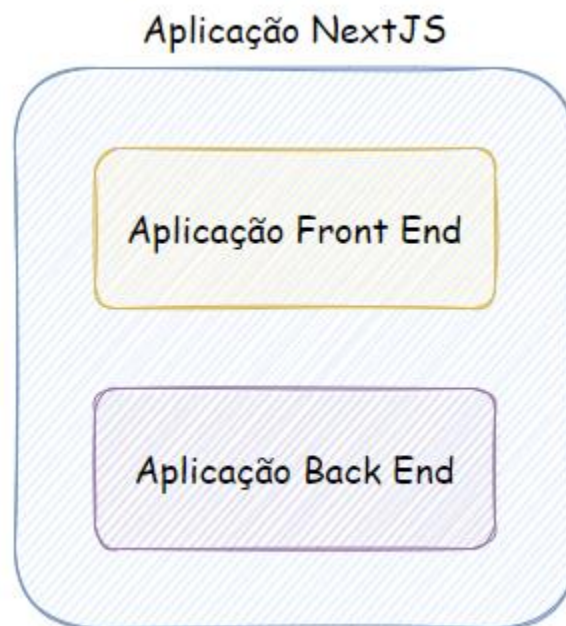
É neste cenário que entra o conceito de Server-Side Rendering. A ideia é renderizar - ou seja, montar o HTML - previamente, do lado do servidor. Quando a página for entregue ao cliente - o navegador -, ela já estará pronta, composta pelos elementos HTML que caracterizam a sua DOM. A renderização do lado do servidor pode, portanto, trazer melhor experiência para o usuário no que diz respeito ao tempo de carregamento de uma página bem como auxiliar na SEO (Search Engine Optimization).

OBS: Vale observar que o próprio React possui mecanismos que viabilizam a renderização do lado do servidor. Entretanto, o NextJS traz um nível de abstração ainda mais alto do que aquele oferecido pelo React.

1.2 (Roteamento baseado em arquivos) Lembra do que é uma Single Page Application? A ideia é muito simples. Uma aplicação que exibe conteúdo geralmente na parte central da tela e links de navegação laterais e superiores. Conforme o usuário clica nos links, o conteúdo na parte central é atualizado sem que uma nova requisição *síncrona* seja enviada pelo navegador ao servidor. Isto é, requisições podem ser enviadas, porém, via Ajax. Uma vez que o conteúdo esteja pronto, ele é exibido na parte central, sem que aconteça aquele famoso "piscar" da tela. Em geral, quando o usuário clica em um link, ele acessa uma nova URL da aplicação. **Roteamento é o mecanismo que nos permite especificar o conteúdo a ser exibido em função da URL visitada pelo usuário.**

Embora o próprio React possua mecanismos para roteamento, o NextJS oferece esta funcionalidade baseada simplesmente na organização de arquivos em pastas certas, o que tende a reduzir a quantidade de código a ser escrito.

1.3 (Desenvolvimento Full Stack) Uma aplicação NextJS viabiliza a escrita de código Front End e Back End. Veja esta figura.



2 Desenvolvimento

Nesta seção, desenvolveremos nossa primeira aplicação NextJS. Será nosso **"Hello World NextJS!"**

2.1 (NodeJS) Certifique-se de que você possui uma versão recente do NodeJS instalada no seu computador. Procure fazer a instalação do NodeJS usando um Node Version Manager.

2.2 (Workspace) Crie uma pasta que terá como finalidade abrigar diversas subpastas. Cada uma delas será um projeto NextJS. Lembre-se de evitar nomes com espaço em branco, caracteres especiais e de não utilizar diretórios sob restrições impostas pelo sistema operacional. Para usuários do Windows, o diretório pode ser algo como **C:\Users\rodri\Documents\workspaces\pessoal\nextjs**.

2.3 (Criando um projeto) Abra um terminal e navegue até o seu workspace com

```
cd path/do/seu/workspace
```

Use

```
npx create-next-app meu-primeiro-app
```

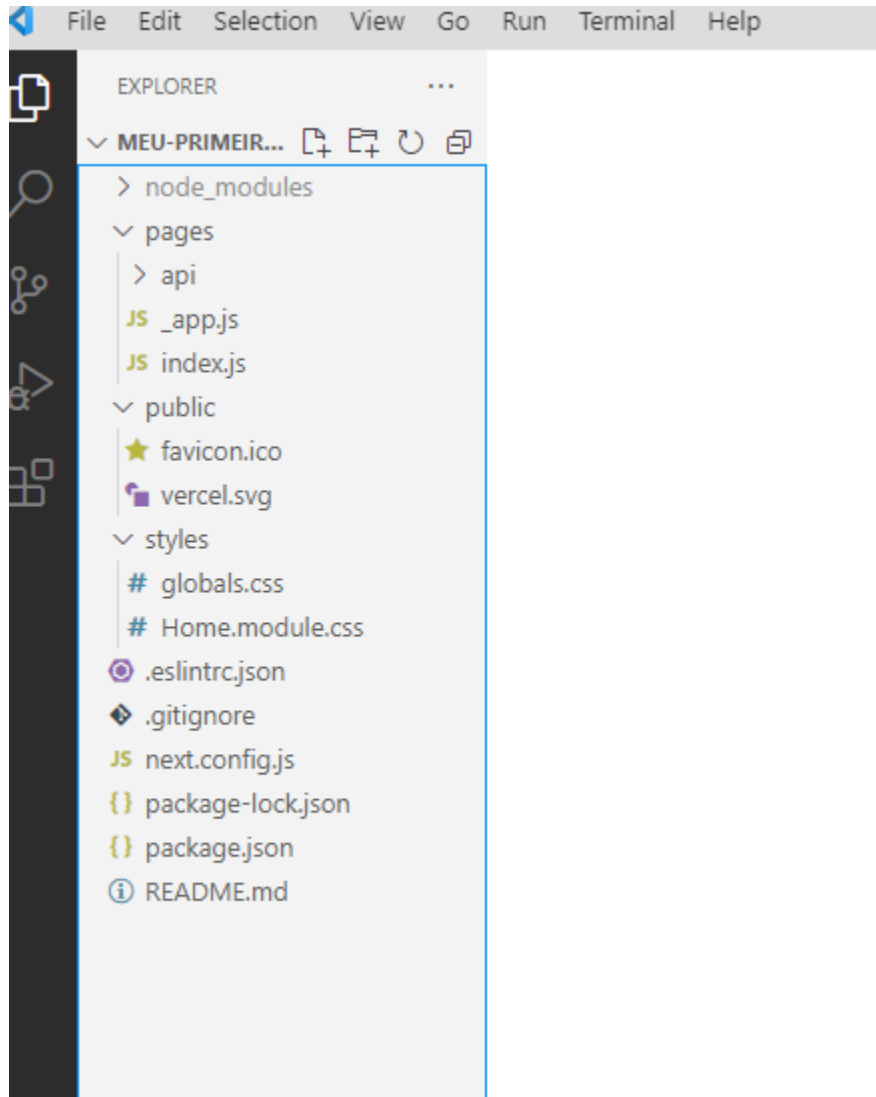
Uma pasta chamada meu-primeiro-app será criada. Use

```
cd meu-primeiro-app
```

no terminal e

```
code .
```

para abrir uma instância do VS Code vinculada a ela. A expectativa é que você tenha obtido uma aplicação com esta estrutura.



Conforme desenvolvemos, aprenderemos sobre cada uma das pastas e arquivos.

2.4 (Executando o aplicativo) No VS Code, clique Terminal >> New Terminal. Use

`npm run dev`

para colocar o aplicativo em execução. Visite

`localhost:3000`

numa aba de seu navegador preferido para visualizar algo assim.

Welcome to **Next.js!**

Get started by editing `pages/index.js`

Documentation →

Find in-depth information about Next.js features and API.

Learn →

Learn about Next.js in an interactive course with quizzes!

Examples →

Discover and deploy boilerplate example Next.js projects.

Deploy →

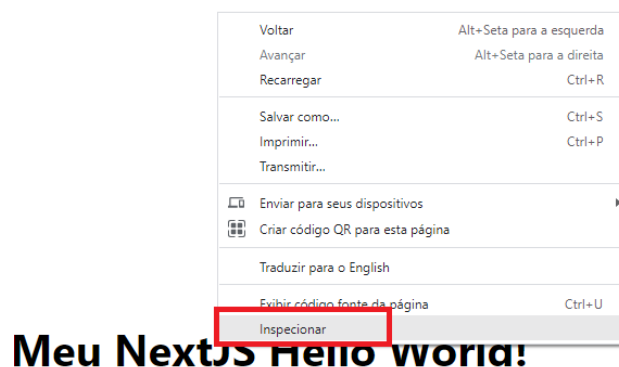
Instantly deploy your Next.js site to a public URL with Vercel.

Observe que este conteúdo é definido por um componente React comum. Ele pode ser encontrado no arquivo **pages/index.js**. Abra o arquivo e inspecione seu conteúdo. A seguir, mantenha neste mesmo arquivo somente o seguinte conteúdo.

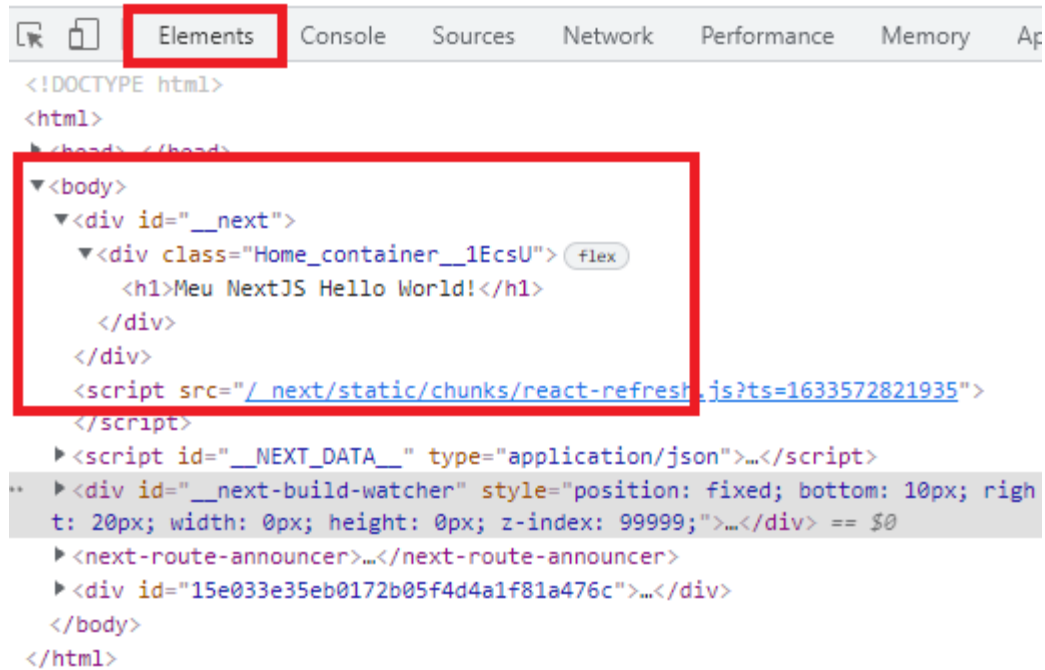
```
import styles from '../styles/Home.module.css'

export default function Home() {
  return (
    <div className={styles.container}>
      <h1>Meu NextJS Hello World!</h1>
    </div>
  )
}
```

Salve o arquivo e visualize o resultado no navegador, que deve ter sido atualizado automaticamente. Clique com o direito na página - ainda no navegador - e escolha "Inspecionar", como exibido a seguir.



Na aba elements, deve ser possível verificar a existência do elemento **h1**. Veja.



```
<!DOCTYPE html>
<html>
  <head> </head>
  <body>
    <div id="__next">
      <div class="Home_container__1EcsU" flex>
        <h1>Meu NextJS Hello World!</h1>
      </div>
    </div>
    <script src="/_next/static/chunks/react-refresh.js?ts=1633572821935">
    </script>
    <script id="__NEXT_DATA__" type="application/json">...</script>
    <div id="__next-build-watcher" style="position: fixed; bottom: 10px; right: 20px; width: 0px; height: 0px; z-index: 99999;">...</div> == $0
    <next-route-announcer>...</next-route-announcer>
    <div id="15e033e35eb0172b05f4d4a1f81a476c">...</div>
  </body>
</html>
```

Referências

Next.js by Vercel - The React Framework. 2021. Disponível em <<https://nextjs.org>>. Acesso em outubro de 2021.