



---

**Faculdade de Tecnologia de Americana “Ministro Ralph Biasi”**  
**Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas**

**Leonardo de Souza Bezerra**

**O impacto da abordagem de renderização na otimização para motores de busca**

**Americana, SP**

**2024**

**Leonardo de Souza Bezerra**

**O impacto da abordagem de renderização na otimização para motores de busca**

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas na área de concentração em Desenvolvimento de sistemas.

**Orientador(a): Prof.<sup>(a)</sup> Me. Thiago Salhab Alves**

Este trabalho corresponde à versão final do Trabalho de Conclusão de Curso apresentado por Leonardo de Souza Bezerra e orientado pelo Prof.<sup>(a)</sup> Me. Thiago Salhab Alves

**Americana, SP**

**2024**

FICHA CATALOGRÁFICA – Biblioteca Fatec Americana  
Ministro Ralph Biasi- CEETEPS Dados Internacionais de  
Catalogação-na-fonte

BEZERRA, Leonardo de Souza

O impacto da abordagem de renderização na otimização para motores de busca. / Leonardo de Souza Bezerra – Americana, 2024.

71f.

Monografia (Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) - - Faculdade de Tecnologia de Americana Ministro Ralph Biasi – Centro Estadual de Educação Tecnológica Paula Souza

Orientador: Prof. Ms. Thiago Salhab Alves

1. Desenvolvimento de software 2. Linguagem de programação 3. WEB – rede de computadores. I. BEZERRA, Leonardo de Souza II. ALVES, Thiago Salhab III. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana Ministro Ralph Biasi

CDU: 681.3.05

681.3.061

681.519WEB

Elaborada pelo autor por meio de sistema automático gerador de ficha catalográfica da Fatec de Americana Ministro Ralph Biasi.

**Leonardo de Souza Bezerra**

**O impacto da abordagem de renderização na otimização para motores de busca**

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas pelo Centro Paula Souza – FATEC Faculdade de Tecnologia de Americana Ministro Ralph Biasi.

Área de concentração: Desenvolvimento de Sistemas.

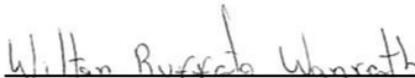
Americana, 20 de junho de 2024.

**Banca Examinadora:**



---

Thiago Salhab Alves  
Mestre  
Fatec Americana Ministro Ralph Biasi



---

Wilton Rufatto Wonrath  
Especialista  
Fatec Americana Ministro Ralph Biasi



---

Eduardo Antônio Vicentini  
Mestre  
Fatec Americana Ministro Ralph Biasi

## RESUMO

Comumente são utilizadas duas abordagens para renderizar o conteúdo de um site: renderização do lado do servidor (SSR) e renderização do lado do cliente (CSR). Ambas as abordagens possuem suas vantagens e desvantagens, sendo uma dessas otimização para motores de busca (SEO). Um site ter SEO ruim faz com que esse não apareça no topo das buscas feitas em mecanismos de busca, algo que pode ser muito prejudicial para aqueles que desenvolveram o site. Diversas fontes mostram que a abordagem CSR possui SEO pior do que a abordagem SSR e os principais motivos disso são: falta de compatibilidade com motores de busca, visto que alguns motores podem não ser capazes de processar conteúdo renderizado no lado do cliente e alguns problemas relacionados ao desempenho do site, visto que essa abordagem pode afetar a velocidade de carregamento inicial. Neste trabalho foram feitos testes de ambos esses aspectos, compatibilidade e desempenho, para determinar se a abordagem CSR, de fato, possui um impacto considerável em SEO. Através do teste de compatibilidade foi possível determinar que o mecanismo Google, o mais utilizado no mundo, é capaz de processar conteúdo renderizado no lado do cliente, mas não foi possível confirmar isso para outros mecanismos. Sendo assim, o aspecto da compatibilidade só se torna um possível problema caso se deseje indexar um site que use a abordagem CSR em algum mecanismo que não seja o Google. Já através do teste de desempenho, foi possível determinar que ambas as abordagens possuem vantagens e desvantagens. Com CSR, o site possui pior estabilidade visual e pior velocidade de carregamento do conteúdo, ao menos, em dispositivos menos potentes, tal como dispositivos móveis. Com SSR, o site possui pior exibição gradual do conteúdo e maior tempo em que a linha principal de execução do navegador fica bloqueada, ao menos, em dispositivos menos potentes, além de pior velocidade de resposta do servidor a primeira requisição do cliente caso o site possua muitos dados dinâmicos. Sendo assim, o desempenho pode se tornar um empecilho para SEO, tanto para a abordagem SSR quanto CSR, caso não sejam tomadas medidas que melhorem essas características.

**Palavras-Chave:** Renderização; Servidor; Cliente; Otimização; Motor de busca.

## ABSTRACT

*Two approaches are commonly used to render a websites content: server-side rendering (SSR) and client-side rendering (CSR). Both approaches have their advantages and disadvantages, one of which is Search Engine Optimization (SEO). A website with poor SEO means that it does not appear at the top of searches made on search engines, something that can be very harmful for those who developed the website. Several sources show that the CSR approach has worse SEO than the SSR approach and the main reasons for this are: lack of compatibility with search engines, as some engines may not be able to process content rendered on the client side and some problems related to site performance, as this approach may affect initial loading speed. In this project, tests were carried out on both these aspects, compatibility and performance, to determine whether the CSR approach, in fact, has a considerably negative impact on SEO. Through the compatibility test, it was possible to determine that Google, the most used search engine in the world, is capable of processing content rendered on the client side, but it was not possible to confirm this for other engines. Therefore, the compatibility aspect only becomes a possible problem if you want to index a site that uses the CSR approach in an engine other than Google. Through the performance test, it was possible to determine that both approaches have advantages and disadvantages. With CSR, the website has worse visual stability and worse content loading speed, at least on less powerful devices, such as mobile devices. With SSR, the site has a worse gradual display of content and a longer time in which the browser's main thread of execution is blocked, at least on less powerful devices, in addition to worse server response speed to the first client request if the site have a lot of dynamic data. Therefore, performance can become an obstacle for SEO, both for the SSR and CSR approaches, if measures are not taken to improve these characteristics.*

**Keywords:** *Rendering; Server; Client; Optimization; Search engine.*

## LISTA DE FIGURAS

Figura 1 – Exemplo de código HTML. ....	15
Figura 2 – Exemplo de regra CSS. ....	15
Figura 3 – Sequência de renderização do lado do servidor. ....	21
Figura 4 – Sequência de renderização do lado do cliente. ....	23
Figura 5 – Tela da ferramenta de software Lighthouse. ....	33
Figura 6 – Exemplo de relatório de desempenho no modo <i>navigation</i> do Lighthouse. ....	34
Figura 7 – Exemplo de relatório de desempenho no modo <i>timespan</i> do Lighthouse. ....	34
Figura 8 – Opções de configurações adicionais do Lighthouse. ....	35
Figura 9 – Exemplo de TTFB com aba <i>network</i> do Chrome DevTools. ....	35
Figura 10 – Diferentes opções de velocidade de rede na ferramenta Chrome DevTools. ....	36
Figura 11 – Opções de configurações adicionais do Chrome DevTools. ....	37
Figura 12 – Arquivos do site do teste de compatibilidade. ....	41
Figura 13 – Página inicial do site do teste de compatibilidade. ....	42
Figura 14 – Página <i>/web</i> do site do teste de compatibilidade. ....	43
Figura 15 – Página <i>/seo</i> do site do teste de compatibilidade. ....	43
Figura 16 – Página <i>/render</i> do site do teste de compatibilidade. ....	44
Figura 17 – Arquivos dos sites do teste de desempenho. ....	45
Figura 18 – Elemento de texto nos sites do teste de desempenho. ....	46
Figura 19 – Elemento de imagens nos sites do teste de desempenho. ....	47
Figura 20 – Elementos dinâmicos-interativos nos sites do teste de desempenho. ....	48
Figura 21 – Exemplo de dado da rota <i>/posts</i> da API JSONPlaceholder. ....	48
Figura 22 – Elemento de comentários nos sites do teste de desempenho. ....	49
Figura 23 – Exemplo de dado da rota <i>/comments</i> da API JSONPlaceholder. ....	49
Figura 24 – Elemento de dados dinâmicos nos sites do teste de desempenho. ....	50
Figura 25 – Exemplo de dado da rota <i>/photos</i> da API JSONPlaceholder. ....	50
Figura 26 – Quantidade de dados por rota na API JSONPlaceholder. ....	51
Figura 27 – Elemento de formulário nos sites do teste de desempenho. ....	51

## LISTA DE TABELAS

Tabela 1 – Mecanismos de busca mais utilizados em maio de 2024.....	29
Tabela 2 – Requisitos funcionais do site do teste de compatibilidade.....	31
Tabela 3 – Requisitos funcionais dos sites do teste de desempenho. ....	37
Tabela 4 – Resultado da pesquisa pelo site do teste de compatibilidade.....	53
Tabela 5 – Resultados das pesquisas 1, 2 e 3 do teste de compatibilidade.....	54
Tabela 6 – Resultados das pesquisas 4, 5, 6 e 7 do teste de compatibilidade.....	55
Tabela 7 – Valores bons e ruins para as métricas <i>Web Vitals</i> .....	57
Tabela 8 – Resultado das métricas do Lighthouse no cenário pequeno. ....	58
Tabela 9 – Resultado das métricas do Chrome DevTools no cenário pequeno. ....	58
Tabela 10 – Resultado das métricas do Lighthouse no cenário médio. ....	59
Tabela 11 – Resultado das métricas do Chrome DevTools no cenário médio.....	59
Tabela 12 – Resultado das métricas do Lighthouse no cenário grande. ....	60
Tabela 13 – Resultado das métricas do Chrome DevTools no cenário grande.....	61
Tabela 14 – Média dos resultados das métricas do Lighthouse nos três cenários. ....	61
Tabela 15 – Média dos resultados das métricas do Chrome DevTools nos três cenários.....	62

## LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
CLS	Cumulative Layout Shift
CRP	Critical Rendering Path
CSR	Client Side Rendering
CSS	Cascading Style Sheets
CTR	Clickthrough rate
DOM	Document Object Model
FCP	First Contentful Paint
FID	First Input Delay
HTML	Hypertext Markup Language
HTTP	Hypertext transfer protocol
INP	Interaction to Next Paint
JS	JavaScript
JSON	JavaScript Object Notation
LCP	Largest Content Paint
SEO	Search Engine Optimization
SI	Speed Index
SPA	Single Page Application
SSG	Static Site Generation
SSR	Server Side Rendering
TBT	Total Blocking Time
TTFB	Time to First Byte

TTI Time to Interactive

URL Uniform Resource Locator

## SUMÁRIO

1	INTRODUÇÃO.....	12
2	WEB.....	14
2.1	LINGUAGENS DA WEB.....	14
2.1.1	HTML.....	15
2.1.2	CSS.....	15
2.1.3	JavaScript.....	16
2.2	HTTP.....	16
2.3	SERVIDOR WEB.....	17
2.4	NAVEGADOR WEB.....	17
3	MOTORES DE BUSCA.....	19
3.1	SEO.....	19
4	RENDERIZAÇÃO DE PÁGINAS WEB.....	21
4.1	SSR.....	21
4.1.1	SSG.....	22
4.1.2	Hydration.....	23
4.2	CSR.....	23
4.3	COMO A RENDERIZAÇÃO PODE IMPACTAR SEO.....	24
4.3.1	Compatibilidade com motores de busca.....	25
4.3.2	Desempenho do site.....	25
4.3.3	Outros possíveis fatores.....	27
5	MATERIAIS E MÉTODOS.....	29
5.1	TESTE DE COMPATIBILIDADE.....	29
5.1.1	Requisitos do site.....	31
5.2	TESTE DE DESEMPENHO.....	32
5.2.1	Requisitos dos sites.....	37

5.3	TECNOLOGIAS .....	37
5.3.1	Tecnologias de desenvolvimento.....	38
5.3.2	Tecnologias de teste.....	39
6	PROTÓTIPOS.....	41
6.1	SITE DO TESTE DE COMPATIBILIDADE.....	41
6.2	SITES DO TESTE DE DESEMPEÑO .....	44
7	RESULTADOS E DISCUSSÕES .....	53
7.1	RESULTADOS DO TESTE DE COMPATIBILIDADE.....	53
7.1.1	Resultados com os endereços do site.....	53
7.1.2	Resultados com o conteúdo do site .....	55
7.2	RESULTADOS DO TESTE DE DESEMPEÑO .....	56
7.2.1	Resultados no cenário pequeno .....	57
7.2.2	Resultados no cenário médio.....	59
7.2.3	Resultados no cenário grande .....	60
7.2.4	Resultados gerais .....	61
8	CONSIDERAÇÕES FINAIS .....	63
	REFERÊNCIAS .....	65
	APÊNDICE A – REPOSITÓRIO PARA OS CÓDIGOS DOS SITES DESENVOLVIDOS E TODOS OS RELATÓRIOS GERADOS NO TESTE DE DESEMPEÑO .....	70

## 1 INTRODUÇÃO

No que diz respeito a maneira como o conteúdo de um site pode ser exibido, duas abordagens são comumente utilizadas: renderização do lado do servidor (SSR) e renderização do lado do cliente (CSR). Em SSR, o servidor envia o documento referente ao site com todo, ou ao menos a maior parte do seu conteúdo já inserido nesse documento para o cliente, enquanto em CSR, o servidor não envia o documento do site completo, mas apenas sua versão básica e sem conteúdo, em conjunto com código, geralmente JavaScript (JS), que será processado pelo cliente e irá inserir o conteúdo na página. Ambas as abordagens possuem suas vantagens e desvantagens, sendo uma delas *Search Engine Optimization* (SEO) ou otimização para motores de busca (Vega, 2017). Um site ter SEO ruim significa que este não será exibido no topo das buscas feitas pelos usuários em motores de busca, algo que pode ser muito prejudicial para aqueles que mantem o site.

Sabe-se que a abordagem CSR pode apresentar resultados ruins de SEO e os principais motivos disso são a falta de compatibilidade com motores de busca, já que esses podem ter problemas para processar o código JS do site (Beke, 2018, p. 29), além de alguns aspectos voltados para o desempenho do site, já que a abordagem CSR pode tornar o carregamento inicial da página mais lento (Vega, 2017). Apesar disso, foram encontrados poucos estudos que abordam o aspecto da compatibilidade levando em conta o estado atual dos mecanismos de busca disponíveis na Web, além de poucos estudos que utilizam o conjunto de métricas moderno para medição de performance em sites, as *Web Vitals* (Walton, 2023a).

O presente trabalho possui como justificativa a pretensão de contribuir para essa área do conhecimento, buscando obter resultados que auxiliem o entendimento geral sobre as abordagens de renderização SSR, CSR e o impacto dessas abordagens em SEO.

O objetivo geral deste trabalho é determinar o impacto que a abordagem CSR possui em SEO, através de testes de compatibilidade com motores de busca e testes de desempenho, comparando essa abordagem com a abordagem SSR utilizando o conjunto de métricas *Web Vitals*.

Em relação aos objetivos específicos deste trabalho, estes são:

- Realizar testes de compatibilidade com motores de busca na abordagem CSR;

- Realizar testes de desempenho, utilizando o conjunto de métricas *Web Vitals*, nas abordagens SSR e CSR;
- Analisar os resultados obtidos e determinar como a abordagem de renderização impacta SEO.

O capítulo 2 deste trabalho abordará a Web, explicando as principais tecnologias que a compõe; O capítulo 3 abordará os motores de busca, explicando como estes funcionam e o que é SEO; O capítulo 4 abordará a renderização de páginas web, explicando sobre as abordagens SSR e CSR e como essas podem impactar SEO; O capítulo 5 abordará os materiais e métodos deste trabalho, elaborando os testes que foram feitos e as tecnologias que foram utilizadas; O capítulo 6 exibirá os protótipos que foram desenvolvidos para a realização dos testes; O capítulo 7 exibirá os resultados e discussões dos testes realizados; O capítulo 8 apresentará as considerações finais deste trabalho.

## 2 WEB

Responsável por interconectar dispositivos e pessoas ao redor do mundo todo, a internet revolucionou o mundo do computador e da comunicação. Esta foi projetada para ser ou servir uma infraestrutura geral na qual novas aplicações poderiam ser desenvolvidas em cima. Aplicações com os mais diversos fins, tais como: correio eletrônico, transferência de arquivos, comunicação por voz, entre diversos outros (Leiner et al., 1997, p. 102-104).

Entretanto, dentre as aplicações desenvolvidas para a internet, a que mais chamou a atenção do público geral foi a Web. O provável motivo disto é o fato de a Web funcionar por demanda, isto é, os usuários podem requisitar o conteúdo que quiserem, quando quiserem, que estes o irão receber, diferente das transmissões de rádio ou televisão, em que o usuário deveria sintonizar no momento específico que o provedor disponibilizasse o conteúdo de seu interesse (Kurose; Ross, 2009, p. 72).

Lins (2013, p. 24) define a Web como um espaço de rede em que as informações são documentos de hipertexto, ou seja, documentos que apresentam conteúdos e referências para outros conteúdos. Essas referências podem apontar:

- Para outro local no mesmo documento, chamados de páginas;
- Para outras páginas localizadas no mesmo local de rede, chamados de sites;
- Para outras páginas de outros sites.

Kurose e Ross (2009, p. 48) dizem que essa tecnologia foi criada entre 1989 e 1991, época em que o programador suíço Tim Berners Lee e seus companheiros haviam desenvolvido as versões iniciais de quatro componentes essenciais para a Web: *Hypertext markup language* (HTML), *Hypertext transfer protocol* (HTTP), um servidor web e um navegador web. Esses componentes são cruciais para o entendimento da Web e serão elaborados a seguir.

### 2.1 LINGUAGENS DA WEB

Além da linguagem de marcação HTML, existem outras duas linguagens muito importantes no contexto da Web, que são: *Cascading Style Sheets* (CSS) e JavaScript (JS). Essas linguagens serão explicadas a seguir.

### 2.1.1 HTML

HTML é a linguagem de marcação que se encontra no núcleo da Web. Originalmente utilizada apenas para descrever documentos científicos de forma semântica, mas eventualmente se adaptou para descrever outros tipos de documentos e até aplicações. Com o uso de *tags* é possível delimitar o início e o fim de elementos HTML, que podem conter dentro de si textos ou até mesmo outros elementos HTML (WHATWG, 2024). Um exemplo de código HTML pode ser visto na figura 1.

Figura 1 – Exemplo de código HTML.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Sample page</title>
  </head>
  <body>
    <h1>Sample page</h1>
    <p>This is a <a href="demo.html">simple</a> sample.</p>
    <!-- this is a comment -->
  </body>
</html>
```

Fonte: (WHATWG, 2024)

### 2.1.2 CSS

CSS é a linguagem usada para desenvolver folhas de estilo que permitem descrever a forma como documentos estruturados, tal como documentos HTML, são renderizados. Através de regras de CSS é possível definir aspectos como cor, fonte, posicionamento e muito mais. Geralmente regras CSS possuem um seletor, que define qual elemento será estilizado e uma declaração, que define como o elemento será estilizado (Atkins Jr. e Sapin, 2021). O formato de uma regra CSS pode ser visto na figura 2.

Figura 2 – Exemplo de regra CSS.

```
p > a {
  color: blue;
  text-decoration: underline;
}
```

Fonte: (Atkins Jr. e Sapin, 2021)

### 2.1.3 JavaScript

JS faz parte da tríade de tecnologias que todos os desenvolvedores Web devem conhecer, junto com HTML e CSS. É uma linguagem de alto nível, dinâmica, interpretada e não tipada. É presente na maioria dos sites modernos e a maioria dos navegadores modernos incluem interpretadores JS (Flanagan, 2012, p. 1).

Sendo uma linguagem de programação, esta oferece diversas funcionalidades úteis, como: armazenamento em variáveis, operações lógicas, condições, loops entre muitas outras. Mas além das funcionalidades básicas de uma linguagem de programação, JS também pode ser integrada com o lado do cliente na Web, através de scripts que se relacionam com documentos HTML e CSS, proporcionando um maior controle sobre esses aspectos para o desenvolvedor (Flanagan, 2012, p. 4 e seg). “Um dos motivos da ascensão de scripting é que este permite interfaces de usuário ricas e responsivas para aplicações web” (Ollila; Mäkitalo; Mikkonen, 2021, p. 790, tradução nossa).

Tratando de JS, é importante mencionar também os frameworks JS, que facilitaram o desenvolvimento de aplicações web mais dinâmicas e interativas. Frameworks consistem em bibliotecas que definem uma maneira na qual um software será desenvolvido, oferecendo assim previsibilidade e homogeneidade durante o desenvolvimento deste. Os frameworks JS são comumente conhecidos como *Client-side frameworks* e, dentre os diversos que existem, aqueles que são mais utilizados são: Ember, Angular, Vue e React (MDN Web Docs, 2024a).

## 2.2 HTTP

HTTP é o principal protocolo para realizar transferência de informação na Web. Esse disponibiliza uma interface que permite interações com recursos através de mensagens que manipulam ou transferem representações desses recursos. Cada mensagem pode ser uma requisição ou uma resposta, que são feitas por clientes e servidores respectivamente. Um cliente deve comunicar suas intenções e rotear suas mensagens para o servidor, que escuta essas requisições e envia respostas ao cliente, que irá examinar se as suas intenções foram atendidas. Essas intenções são feitas através dos métodos HTTP, tais como GET e POST, que são utilizadas para recuperação e processamento de alguma informação, respectivamente, além de diversos outros métodos (Fielding; Nottingham; Reschke, 2022).

### 2.3 SERVIDOR WEB

Yeager e McGrath (1996, p. 20) descrevem os principais aspectos de um servidor web:

- Plataforma: se trata do hardware, sistema operacional e software de rede que fazem parte do servidor;
- Software: o software que efetivamente receberá e decodificará as requisições, e, a partir destas, enviará respostas;
- Informação: aquilo que o servidor efetivamente envia.

Um servidor deve ficar ativo de forma contínua, esperando por requisições de usuários na internet. Quando uma requisição é recebida, é estabelecido uma conexão entre “portas”, que são uma abstração que proporcionam um jeito simples e genérico de criar e usar conexões de rede. Estas portas determinam exatamente o tipo de recurso que está sendo requisitado, e, através destas, o recurso é enviado ao cliente (Yeager; McGrath, 1996, p. 20).

### 2.4 NAVEGADOR WEB

Com a conceptualização da Web, ficou claro a necessidade de um software que facilitasse o acesso aos endereços dos servidores web e exibisse os conteúdos requisitados de forma intuitiva e fácil. Um navegador web tem justamente esse propósito.

Sua tecnologia e seu design permitiam mostrar a página de conteúdo do sítio de modo agradável e navegar entre as informações por meio das referências, os hyperlinks, campos nos quais o usuário poderia clicar com um mouse para deslocar-se a outras páginas ou sítios (Lins, 2013, p. 24)

Para o usuário, o ato de acessar uma página web se trata de um simples ato de digitar um endereço no navegador e acessá-lo. Entretanto, para o navegador, existe uma série de ações que devem ser realizadas a fim de exibir o conteúdo requisitado.

Após a conexão entre cliente e servidor ter sido estabelecida, conforme o navegador recebe os dados referente ao documento solicitado, é iniciado um processo chamado de *Critical Rendering Path* (CRP), que se trata da sequência de passos que o navegador deve executar para converter todo o código HTML, CSS e JS nos pixels da tela que representam o site. Uma das

etapas do CRP é a criação da árvore *Document Object Model* (DOM), que é um conceito importante no contexto dos navegadores (MDN Web Docs, 2024b).

DOM se trata de uma representação de um documento estruturado, tal como documentos HTML, no formato de uma hierarquia de objetos chamados de *nodes*. DOM também fornece um conjunto de interfaces para a manipulação desses objetos, que são compatíveis com uma grande gama de linguagens de programação (Hors et al., 2004). É através dessas interfaces que linguagens, como JS, podem alterar os documentos que representam uma página web.

### 3 MOTORES DE BUSCA

Cendón (2001, p. 39) diz que nos primórdios da Web surgiu a preocupação de criar ferramentas de busca que fossem capazes de localizar recursos informacionais nessa. Dentre as soluções propostas, existe o motor de busca. Seymour, Frantsvog e Kumar (2011, p. 55) definem que motores de busca operam em três etapas: *web crawling*, indexação (*indexing*) e busca (*search*).

Na etapa de *web crawling* são utilizados robôs, também conhecidos como aranhas (*spiders*), agentes, viajantes (*wanderers*), rastejadores (*crawlers*) ou vermes (*worms*), que realizam buscas regulares pela internet a fim de obter a maior quantidade possível de documentos para serem integrados no banco de dados do motor de busca. Diversos desses robôs podem operar em paralelo e estes efetuam suas buscas de forma sistemática, através dos *links* presentes nas páginas que visitam (Cendón, 2001, p. 41).

Na etapa de indexação, os documentos vasculhados pelos robôs são encaminhados para indexadores, que extraem as informações relevantes do documento, tais como: *Uniform Resource Locators* (URLs), títulos, resumos, tamanho e palavras. Então essas informações são armazenadas em um banco de dados (Cendón, 2001, p. 41). De maneira mais detalhada, as informações relevantes extraídas do site são analisadas para determinar como a página deve ser indexada no banco de dados, para que posteriormente possa ser rapidamente encontrada no processo de busca (Seymour; Frantsvog; Kumar, 2011, p. 55).

Por fim, na etapa de busca, que geralmente é feita através de uma interface, é possível que usuários formulem uma consulta com o uso de palavras chaves ou até mesmo de linguagem natural, que será utilizada pelo motor de busca propriamente dito, que se trata do programa que irá localizar os itens na base de dados que devem constituir a resposta procurada pelo usuário (Cendón, 2001, p. 41).

#### 3.1 SEO

Um detalhe de suma importância em motores de busca é a ordenação dos resultados obtidos, visto que, um motor que prioriza a exibição de resultados melhores será considerado superior a aqueles que não o fazem (Cendón, 2001, p. 44).

Para garantir um padrão de qualidade nos sites que são exibidos nas listas de resultados, os motores de busca avaliam certos aspectos desses sites a fim de determinar suas pontuações de qualidade, que por fim irão determinar suas posições nas listas. Esses aspectos podem ser, por exemplo: conteúdo, usabilidade, acessibilidade, estrutura da página, entre diversos outros. A exata forma como esses aspectos influenciam na posição final de um site é algo que apenas o criador do algoritmo específico utilizado pelo motor de busca sabe, mas uma coisa é certa: quanto maior for a pontuação de qualidade de um site, melhor serão seus resultados nos motores de busca (Ledford, 2008, p. 10-11).

É justamente para isto que serve SEO. Se trata de realizar otimizações necessárias para que um site tenha os melhores resultados nos motores de busca, o que é algo muito importante, visto que, segundo alguns dados de Dean (2023):

- O primeiro resultado em uma pesquisa no Google possui uma Taxa de Cliques (CTR) média de 27.6%;
- O primeiro resultado orgânico de uma pesquisa possui 10 vezes mais chance de receber um clique quando comparado ao décimo resultado;
- Mover 1 posição para cima em motores de busca pode resultar em um aumento de CTR de até 2.8%,

Esses dados mostram que é importante um site obter boas posições em motores de busca, visto que aqueles que obtém possuem muito mais chances de atrair usuários na Web, e para isso é necessário utilizar boas práticas de SEO.

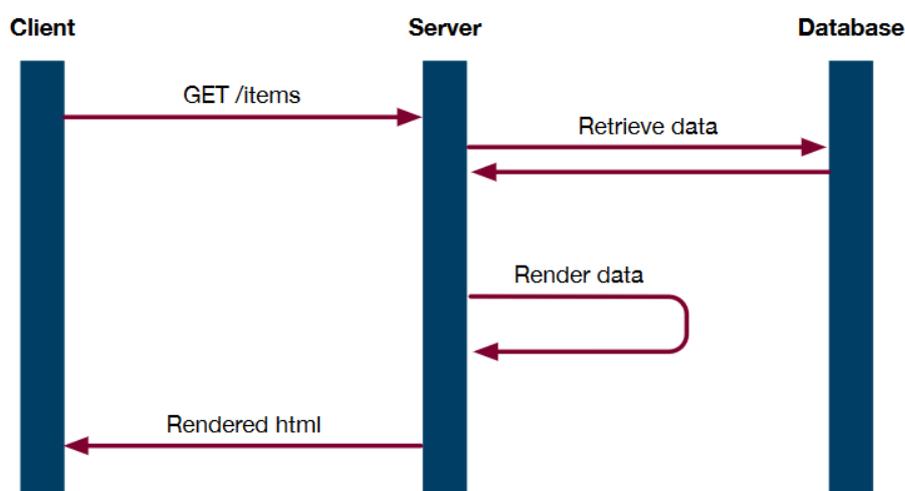
## 4 RENDERIZAÇÃO DE PÁGINAS WEB

Para dar prosseguimento ao estudo proposto neste trabalho, é importante especificar as abordagens de renderização de sites: renderização do lado do servidor (SSR) e renderização do lado do cliente (CSR). Prosseguindo, será tratado como essas abordagens podem impactar SEO.

### 4.1 SSR

Sydorkina (2023) explica que em SSR o código da página é gerado no servidor e então é enviado ao cliente, geralmente um navegador web, cujo único trabalho será executar esse código e exibir a página. A figura 3 demonstra a sequência de passos executados para a renderização do lado do servidor.

Figura 3 – Sequência de renderização do lado do servidor.



Fonte: (Beke, 2018, p. 15)

Nessa figura, percebe-se que o cliente efetua uma requisição *GET* para a rota */items*, fazendo com que o servidor transfira uma representação do recurso desta rota para o cliente. O servidor, então, interage com um banco de dados e utiliza dos dados deste para renderizar, isto é, construir um documento HTML que será enviado ao cliente.

Vega (2017) determina algumas vantagens e desvantagens para a abordagem SSR. Dentre as vantagens, destacam-se:

- Melhor SEO;
- Carregamento inicial da página mais rápido.

Já dentre as desvantagens, destacam-se:

- Maior número de requisições são feitas;
- Uma renderização mais lenta no geral;
- Páginas são recarregadas por inteiras;
- Interações menos “ricas” com o site.

Ademais, quando se trata de SSR, existem dois conceitos que são importantes de serem abordados: *Static Site Generation* (SSG) e *Hydration*. Esses conceitos serão abordados a seguir.

#### 4.1.1 SSG

SSG se trata de uma abordagem muito semelhante a SSR, entretanto, com uma diferença importante. Em SSR a página é renderizada em tempo de requisição, ou seja, todas as vezes que um usuário acessa o site, o servidor efetua a renderização da página antes de enviá-la ao cliente. Já em SSG, o servidor efetua a renderização da página em tempo de *build*, ou seja, o servidor efetua a renderização da página apenas uma vez e já a deixa pronta para ser enviada quando requisitada. O caso de uso para cada uma das abordagens é claro: SSR deve ser usado quando a página possui informações dinâmicas, ou seja, que se alteram entre os usuários, enquanto SSG deve ser usado quando a página possui informações estáticas, ou seja, que não se alteram entre os usuários (Roland, 2023).

Em ambos os casos o servidor foi responsável por renderizar o conteúdo da página, mas ainda assim é importante distinguir essas nomenclaturas, visto que quando se usa o termo SSR se espera que o servidor esteja efetuando a renderização da página para cada requisição recebida, e quando se utiliza o termo SSG se espera que o servidor já tenha a página renderizada antes mesmo do usuário requisitá-la.

### 4.1.2 Hydration

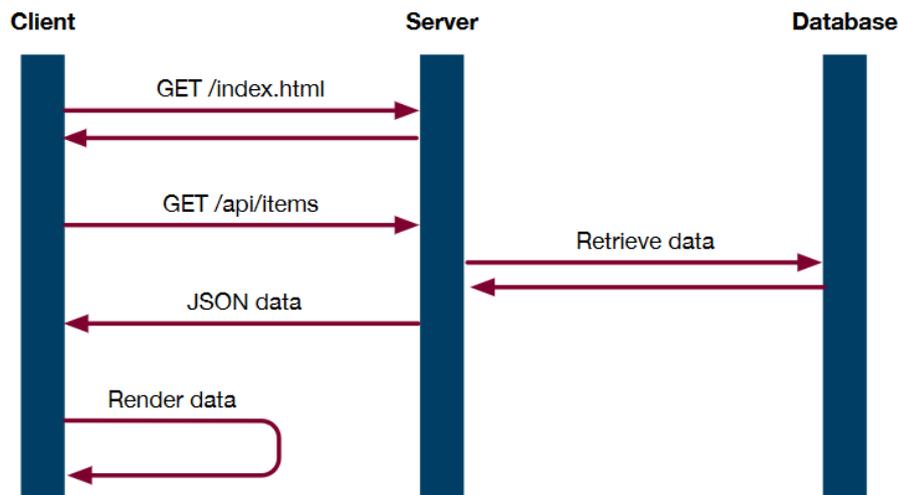
De acordo com a documentação de Next.js (Next.js, 2024a, tradução nossa), “Hydration é o processo de anexar ouvintes de evento no DOM, para fazer com que o HTML estático se torne interativo”.

Como foi mencionado por Vega (2017), uma das desvantagens da abordagem SSR é que essa proporciona interações menos “ricas” no site, e *hydration* é uma técnica que pode mitigar isso, visto que essa possibilita a renderização do HTML inicial da página no servidor, fornecendo os benefícios de SSR, e a interatividade quando a página chega ao cliente, oferecendo benefícios de CSR, o qual será explicado a seguir.

## 4.2 CSR

Vega (2017) explica que a renderização do lado do cliente se trata de renderizar o site utilizando de código JS presente na máquina do cliente. Ou seja, quando o usuário acessa um site, este recebe um documento HTML que serve como base e um arquivo JS que, ao ser executado pelo navegador, irá renderizar os componentes referentes ao site. A figura 4 demonstra a sequência de passos executados para a renderização do lado do cliente.

Figura 4 – Sequência de renderização do lado do cliente.



Fonte: (Beke, 2018, p. 16)

Nota-se que o usuário efetua uma requisição `GET` para a rota `/index.html`, fazendo com que o servidor transfira uma representação do recurso desta rota para o cliente. Este se refere ao

documento base HTML, que na maioria das vezes não possui nenhum conteúdo, entretanto, possui código JS que será capaz de interagir com *Application Programming Interfaces* (APIs). Essas APIs podem ser usadas para resgatar informações do servidor, que serão devolvidas ao cliente no formato *JavaScript Object Notation* (JSON) para serem processadas e utilizadas.

Os dados JSON resgatados podem ser utilizados como conteúdo para serem inseridos na página, isto é, no documento HTML. Esta inserção de dados é possibilitada através da manipulação da DOM, permitindo realizar mudanças na interface do usuário de maneira incremental. Este conjunto de tecnologias é chamado de *Dynamic HTML* (Ollila; Mäkitalo; Mikkonen, 2021, p. 790).

Vega (2017) determina algumas vantagens e desvantagens para CSR. Dentre as vantagens, destacam-se:

- Interação mais “rica” com o site;
- Renderização mais rápida após o carregamento inicial.

Já dentre as desvantagens, destacam-se:

- SEO fraca, caso não implementado corretamente;
- Carregamento inicial mais lento.

### **4.3 COMO A RENDERIZAÇÃO PODE IMPACTAR SEO**

Como foi determinado pelas vantagens e desvantagens de Vega (2017), além deste fato também ser mencionado por Sydorkina (2023) e Beke (2018, p. 28-29), a abordagem CSR apresenta SEO ruim em comparação a abordagem SSR, ao menos, caso essa não seja implementada corretamente. Existem dois principais motivos aparentes do porquê isto acontece: problemas de compatibilidade com motores de busca e questões de desempenho. Esses fatores serão a serem elaborados a seguir, além de uma subseção para analisar se existe algum outro fator para ser levado em consideração.

### 4.3.1 Compatibilidade com motores de busca

O motivo mais aparente, e possivelmente principal para a abordagem CSR impactar SEO negativamente é o a compatibilidade com motores de busca. Beke (2018, p. 29) menciona como que os robôs, comumente chamados de *crawlers*, de certos motores de busca podem não ser capazes de renderizar código JS, dessa forma, prejudicando a indexação do conteúdo de sites CSR. Esse ponto é ainda complementado por Sydorkina (2023), que diz que que sites SSR são mais “amigáveis” para SEO, visto que estes já apresentam suas informações no formato de texto assim que chegam ao cliente, os tornando legíveis para motores de busca. Sendo assim, “compatibilidade com motores de busca” pode ser classificado como um dos fatores que podem ser impactados pela escolha entre SSR e CSR e que podem influenciar SEO.

### 4.3.2 Desempenho do site

Como foi mencionado por Vega (2017), sites CSR possuem um carregamento inicial mais lento devido ao fato de que o navegador precisa fazer o *download* de todo o arquivo JS para apenas então exibir o conteúdo da página, entretanto, também é menciona que sites CSR apresentam um carregamento mais rápido após o carregamento inicial. Esses aspectos, que são voltados para desempenho, também são relevantes para SEO, ao menos, após o surgimento das *Web Vitals*. “As Métricas da Web são uma iniciativa do Google para oferecer orientações unificadas quanto aos indicadores de qualidade essenciais para oferecer uma ótima experiência do usuário na Web” (Walton, 2023a).

Firdausi (2024) diz que esse conjunto de métricas, primordialmente, era usado como um indicador de ranqueamento apenas para dispositivos móveis, entretanto, eventualmente, passou a ser usado para o ranqueamento em *desktops* também, tornando-se um conjunto de métricas fundamental para SEO. As três principais métricas, denominadas *Core Web Vitals* são:

- *Largest Content Paint* (LCP): mede quando o maior elemento de uma página é renderizado. Esse elemento pode ser, por exemplo: uma imagem, vídeo ou texto. Um tempo de 2.5s ou menos é o ideal para essa métrica, um tempo entre 2.5s e 4s significa que precisa de melhorias e um tempo maior do que 4s é ruim (Walton e Pollard, 2024).
- *Interaction to Next Paint* (INP): mede a capacidade de resposta de uma página. Esta leva em consideração o tempo entre uma interação do usuário e alguma atualização visual na tela, por exemplo: o usuário clica em um botão de fazer login e então aparece

na tela uma mensagem de carregamento, sucesso ou erro. Mais especificamente, INP mede o maior tempo entre ação e atualização visual na tela dentre todas as interações presentes na vida útil de um site. Um tempo de 200ms ou menos é o ideal para essa métrica, um tempo entre 200ms e 500ms significa que precisa de melhorias e um tempo maior do que 500ms é ruim (Wagner, 2024).

- *Cumulative Layout Shift* (CLS): mede a estabilidade visual de uma página. Movimentações inesperadas de layout podem fazer com que um usuário acabe, por exemplo, clicando em um botão indesejado ou se perca em sua leitura. Um valor de 0.1 ou menos é o ideal para essa métrica, um valor entre 0.1 e 0.25 significa que precisa de melhorias e um valor maior do que 0.25 é ruim. Nota-se que essa métrica não é medida em tempo e sim por um sistema de pontuação que é determinado pelo impacto de elementos instáveis na página e pela distância percorrida por esses (Mihajlija e Walton, 2024).

É importante mencionar que essas são as três principais métricas e são, de fato, as mais relevantes para oferecer uma boa experiência ao usuário. Entretanto também existem outras métricas suplementares (Walton, 2023a). Diversas dessas outras métricas também são mencionadas por Firdausi (2024):

- *First Input Delay* (FID): mede o tempo entre uma ação do usuário com a página até o momento em que o navegador processa os manipuladores de eventos em resposta a essa ação. Um tempo de 100ms ou menos é o ideal para essa métrica, um tempo entre 100ms e 300ms significa que precisa de melhorias e um tempo maior do que 300ms é ruim (Walton, 2023b);
- *Speed Index* (SI): SI mede a velocidade com que um conteúdo é exibido durante o carregamento da página. Um tempo de 3.4s ou menos é o ideal para essa métrica, um tempo entre 3.4s e 5.8s significa que precisa de melhorias e um tempo maior do que 5.8s é ruim (Lighthouse, 2019a);
- *First Contentful Paint* (FCP): mede o tempo que demora para que qualquer conteúdo seja exibido pelo navegador. Um tempo de 1.8s ou menos é o ideal para essa métrica, um tempo entre 1.8s e 3.0s significa que precisa de melhorias e um tempo maior do que 3.0s é ruim (Walton, 2024);

- *Time to Interactive* (TTI): mede o tempo que uma página demora para carregar todos seus sub-recursos e seja capaz de responder de maneira rápida e confiável as entradas do usuário (Walton, 2023c). Um tempo de 3.8s ou menos é o ideal para essa métrica, um tempo entre 3.9s e 7.3s significa que precisa de melhorias e um tempo maior do que 7.3s é ruim (Lighthouse, 2019b);
- *Total Blocking Time* (TBT): mede o tempo em que um site possui sua interatividade bloqueada por conta de tarefas longas sendo executadas. Existe um conjunto de tarefas que são executadas para que o site funcione corretamente, entretanto, tarefas que duram mais de 50ms são consideradas longas e podem bloquear outras tarefas do site (Walton, 2023d). Um tempo de 200ms ou menos é o ideal para essa métrica, um tempo entre 200ms e 600ms significa que precisa de melhorias e um tempo maior do que 600ms é ruim (Lighthouse, 2019c);
- *Time to First Byte* (TTFB): identifica o tempo entre a solicitação de um recurso e o momento em que o primeiro byte de uma resposta chega ao cliente. Este acontece antes das métricas voltadas para o usuário como FCP ou LCP, mas ainda sim é importante possuir um bom tempo de TTFB para garantir o bom resultado nas outras métricas. Um tempo de 800ms ou menos é o ideal para essa métrica, um tempo entre 800ms e 1800ms significa que precisa de melhorias e um tempo maior do que 1800ms é ruim (Wagner e Pollard, 2024).

Cada uma dessas métricas mede algo relevante para o desempenho de uma página web, e pelo fato de que SSR e CSR podem impactar o desempenho de uma página, ao menos, em aspectos como velocidade de carregamento inicial e capacidade de interatividade da página, é possível que os valores dessas métricas sejam afetados, o que pode impactar SEO. Sendo assim “desempenho do site” pode ser classificado como um dos fatores que pode ser impactado pela escolha entre SSR e CSR e que pode influenciar SEO.

### 4.3.3 Outros possíveis fatores

Para determinar se existem outros fatores que podem impactar SEO e que podem ser afetados pela abordagem de renderização escolhida, é possível utilizar de documentações oficiais de motores de busca a fim de averiguar o que estes consideram como relevantes no ranqueamento

de sites. O Google oferece uma boa noção dos fatores que este leva em consideração para determinar a classificação de um site em uma busca, e estes são (Google, 2024):

- Significado da consulta;
- Relevância do conteúdo;
- Qualidade do conteúdo;
- Usabilidade das páginas web;
- Contexto e configurações.

Percebe-se que a maior parte desses fatores não possuem relação direta com a abordagem de renderização utilizada em um site. Os fatores “relevância do conteúdo” e “qualidade do conteúdo” estão relacionados ao conteúdo em si, ou seja, as informações apresentadas no site. Já os fatores de “significado da consulta” e “contexto e configurações” estão relacionadas ao usuário, ou seja, o que foi pesquisado pelo usuário e quais são as configurações e contexto deste, por exemplo, sua localização geográfica. Resta então o fator de usabilidade das páginas web, que é algo que pode ser medido pelas *Web Vitals*, dessa forma, enquadra-se no aspecto de “desempenho do site”.

Determina-se então que os fatores que podem influenciar SEO e que podem ser impactados pela abordagem de renderização do site, seja essa SSR ou CSR, são: compatibilidade com motores de busca e desempenho do site.

## 5 MATERIAIS E MÉTODOS

Como foi determinado, os fatores “compatibilidade com motores de busca” e “desempenho do site” podem impactar SEO e podem ser influenciados pela escolha entre SSR e CSR. Sendo assim, se torna necessário realizar testes de ambos esses aspectos para avaliar o quão impactante a abordagem de renderização é em SEO. A seguir é elaborado como esses testes serão conduzidos e quais serão as tecnologias utilizadas.

### 5.1 TESTE DE COMPATIBILIDADE

Como é dito por Beke (2018, p. 29) o problema da compatibilidade surge quando um motor não é capaz processar ou indexar conteúdo renderizado através de JS, o que é algo que afeta apenas sites CSR. Sendo assim, para determinar a compatibilidade de um motor de busca com esses tipos de sites será desenvolvido um site CSR, que será indexado em motores de buscas e então será avaliado se este foi indexado corretamente. Para isso, primeiramente, é preciso definir quais motores de busca serão levados em consideração. Existem diversos na Web, mas, naturalmente, existem aqueles que são os mais utilizados. A tabela 1 exhibe os mecanismos de busca mais utilizados em maio de 2024:

Tabela 1 – Mecanismos de busca mais utilizados em maio de 2024.

Mecanismo	Fatia de mercado mundial (%)
Google	90.8
Bing	3.72
Yandex	1.58
Yahoo!	1.19
Baidu	0.92
DuckDuckGo	0.56

Fonte: (StatCounter, 2024)

Dessa forma, ao se determinar a compatibilidade desses mecanismos com CSR será possível obter uma boa noção da compatibilidade geral da Web com essa abordagem. Entretanto, é importante saber que nem todos esses mecanismos disponibilizam ferramentas para solicitar a indexação de um site, e utilizam apenas de robôs automatizados vasculhando a Web até

eventualmente encontrarem um site e indexá-lo, o que pode atrapalhar o teste. A seguir será mencionado para cada mecanismos se esse possui uma ferramenta desse tipo e alguma informação adicional relevante, caso haja.

- Google: possui a ferramenta Google Search Console para solicitar a indexação de sites (Google Search Console, 2024);
- Bing: possui a ferramenta Bing Webmaster Tools para solicitar a indexação de sites (Bing Webmaster Tools, 2024);
- Yandex: possui a ferramenta Yandex Webmaster para solicitar a indexação de sites (Yandex Webmaster, 2024);
- Yahoo!: Não possui nenhuma ferramenta própria para solicitar indexação de sites, mas recomenda o uso do Bing Webmaster Tools para esse propósito (Yahoo Search, 2024);
- Baidu: possui a ferramenta Baidu Webmaster para solicitar a indexação de sites (Baidu Webmaster, 2024). O Baidu é fortemente voltado para o mercado chinês e possui toda sua interface e documentação na língua Chinesa, com pouco suporte para o Português ou para o Inglês;
- DuckDuckGo: Não possui nenhuma ferramenta para solicitar indexação de sites e utiliza majoritariamente do Bing para exibir seus resultados (DuckDuckGo, 2024).

Com essas informações determina-se que as ferramentas que serão usadas para solicitar indexação do site são Google Search Console, Bing Webmaster Tools e Yandex Webmaster, com a expectativa do site aparecer nos mecanismos Google, Bing, Yandex, Yahoo! e DuckDuckGo.

Para testar se o site foi indexado nos mecanismos serão efetuadas pesquisas com o uso do prefixo “site:” seguido pelo endereço do site na barra de pesquisa. Caso o site apareça, significa que está indexado. Caso não apareça, significa que não está indexado. É ideal repetir esse processo para cada página do site para verificar se todas foram indexadas também. Essa pesquisa serve apenas para limitar em quais mecanismos serão feitas buscas por textos que aparecem no site, por exemplo: caso o site apareça apenas no mecanismo Google, as próximas pesquisas se limitarão apenas ao Google, já que não faz sentido pesquisar por textos presentes no site em mecanismos no qual esse nem se quer foi indexado.

Essas pesquisas por textos irão determinar, de fato, a compatibilidade do mecanismo com a abordagem CSR. O site utilizar dessa abordagem significa que todos seus elementos, incluindo

seus textos, foram renderizadas no lado do cliente. O fato de o site aparecer caso sejam pesquisados por textos significa que o motor de busca foi capaz de processar esses textos, refutando a noção de que o motor não possui compatibilidade com conteúdo renderizado no lado do cliente.

É importante perceber as limitações desse teste. Com esse, não é possível afirmar se um determinado mecanismo não possui compatibilidade com a abordagem CSR, visto que mesmo que o site não apareça nesse mecanismo, pode ser apenas por uma questão de tempo. Entretanto, com esse teste é possível afirmar se um determinado mecanismo possui compatibilidade com a abordagem CSR, através da análise de se esse aparece ou não nas pesquisas. Dessa forma, o resultado desse teste será algo como “a abordagem CSR é compatível com, ao menos, tais mecanismos de busca”. Não serão feitas afirmações do tipo “a abordagem CSR não é compatível com tais mecanismos de busca”, visto que esse teste não anula a possibilidade do site, eventualmente, aparecer no mecanismo.

### 5.1.1 Requisitos do site

É ideal definir requisitos para o site que será desenvolvido para esse teste, isto é, definir quais aspectos o site deve ter para que o teste produza resultados mais significativos. A tabela 2 exhibe esses requisitos no formato de requisitos funcionais, descritos por Sommerville (2011, p. 59) como requisitos que descrevem o que um sistema deve fazer.

Tabela 2 – Requisitos funcionais do site do teste de compatibilidade.

Identificação	Requisito funcional	Prioridade
RF001	O site deve utilizar a abordagem CSR.	Essencial
RF002	O site deve ser acessível na Web.	Essencial
RF003	O site deve ter sua indexação solicitada através das ferramentas Google Search Console, Bing Webmaster Tools e Yandex Webmaster.	Essencial
RF004	O site deve ter conteúdo de texto real para ser pesquisado posteriormente.	Importante
RF005	O site deve ter mais de uma página.	Importante
RF006	O site deve ser estilizado com CSS.	Desejável

Fonte: (Próprio autor, 2024)

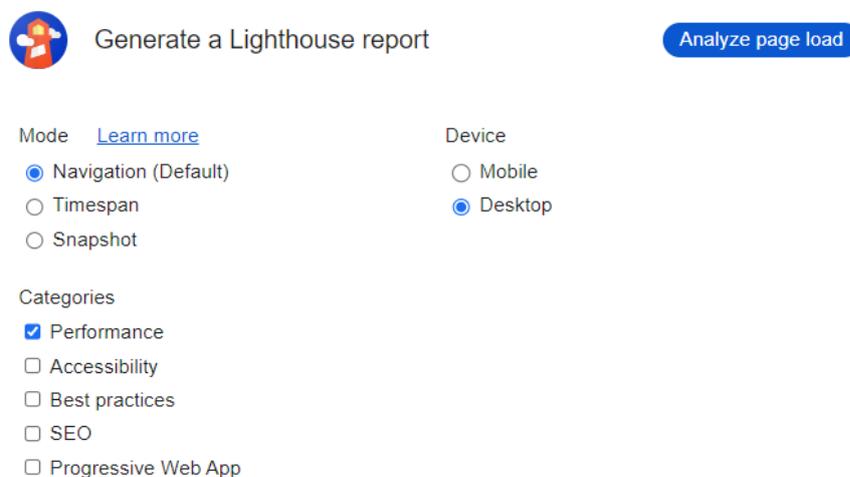
## 5.2 TESTE DE DESEMPENHO

Diferente do aspecto da compatibilidade, que afeta apenas sites CSR, o aspecto do desempenho pode ser tanto beneficiado quanto prejudicado pela escolha dessa abordagem. Como é dito por Vega (2017), sites CSR apresentam um carregamento inicial mais lento, mas uma renderização mais rápida após isso, enquanto sites SSR apresentam carregamento inicial mais rápido, mas uma renderização mais lenta no geral. Por conta disso, esperasse que haja pontos fortes e fracos de desempenho para ambas as abordagens. Sendo assim, serão desenvolvidos dois sites para esse teste, um SSR e outro CSR. Esses sites serão auditados usando as métricas *Web Vitals* e ao final será determinado em quais aspectos essas abordagens podem ser prejudiciais para SEO.

É importante mencionar que testes de *Web Vitals* podem ser feitos em dois ambientes: laboratório e campo. O ambiente de laboratório se trata de um ambiente controlado, onde são utilizadas ferramentas que simulam um carregamento de página consistente, enquanto o ambiente de campo trata das interações que usuários reais tem com o site (Walton, 2019). Essa pesquisa se limita ao ambiente de laboratório, visto que para obter dados relevantes de campo seria necessária uma base de usuários que interaja com os sites. Por conta disso a métrica FID não será mesurada nessa pesquisa visto que essa é uma métrica exclusiva ao ambiente de campo (Walton, 2023b). Também não será mesurado a métrica TTI, visto que, como é dito por Kenny (2024), essa métrica é muito sensível a solicitações de rede atípicas e tarefas longas, sendo melhor utilizar métricas como LCP, SI e TBT em seu lugar.

Com os sites desenvolvidos, será possível utilizar a ferramenta de software Lighthouse para realizar auditorias e obter os valores das métricas de desempenho. Essa ferramenta já vem por padrão no navegador Google Chrome e sua tela pode ser visualizada na figura 5.

Figura 5 – Tela da ferramenta de software Lighthouse.



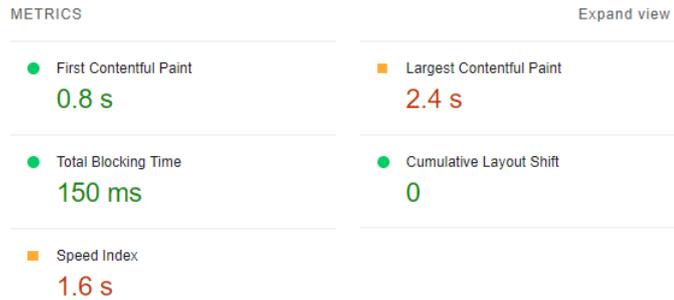
Fonte: (Lighthouse, 2024)

Como é possível ver na imagem, a ferramenta oferece três modos para obtenção de relatórios: *navigation*, *timespan* e *snapshot*. Esses modos serão elaborados a seguir.

- *Navigation*: simula um carregamento de página e efetua suas medições baseado nesse único carregamento (Lighthouse, 2023). Para obter as métricas nesse modo basta pressionar o botão “*Analyze page load*” que a auditoria é feita. Esse modo será usado para obtenção das métricas FCP, TBT, SI, LCP e CLS, que podem ser vistos na figura 6.
- *Timespan*: permite a obtenção de métricas por um período arbitrário (Lighthouse, 2023). Para obter as métricas nesse modo é necessário pressionar os botão “*Start timespan*” e “*End Timespan*”, para iniciar e terminar a medição de métricas, respectivamente. Durante o tempo em que a medição estiver ativa, é possível interagir com o site e os valores gerados no relatório final são referentes a página durante essas interações. Esse modo será usado para obtenção das métrica INP, TBT e CLS, que podem ser vistas na figura 7. As métricas TBT e CLS obtidas nesse modo são diferentes das obtidas no modo *navigation*, visto que aqui é levado em consideração aspectos do site que estão além do carregamento inicial da página.
- *Snapshot*: permite a obtenção de métricas da página em um estado específico (Lighthouse, 2023). Esse modo não se encontra particularmente útil para essa pesquisa e não será usado.

Ademais, dentre as cinco opções disponíveis no campo “*Categories*” da ferramenta apenas a “*Performance*” será selecionada na realização dos testes e serão feitos testes tanto no ambiente “*Mobile*” quanto “*Desktop*”, disponíveis na seção de “*Device*” da ferramenta.

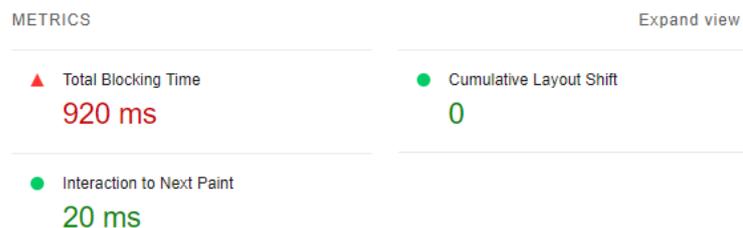
Figura 6 – Exemplo de relatório de desempenho no modo *navigation* do Lighthouse.



Fonte: (Lighthouse, 2024)

A métrica LCP considera, dentre outros, elementos de imagens e elementos de bloco com texto em sua medição (Walton e Pollard, 2024). Sendo assim, é essencial que os sites do teste possuam tais elementos. Esse mesmo conceito se aplica a métrica FCP (Walton, 2024). Para a métrica TBT é essencial apenas que o site possua processos JS, pois essa métrica mede o tempo que certos processos JS estão bloqueando a linha principal de execução do site (Walton, 2023d). Para as métricas CLS e SI, que medem estabilidade visual e velocidade com que os elementos de um site começam a aparecer na tela, é importante que o site possua elementos dinâmicos, que podem ser inseridos a partir de uma chamada de API, por exemplo. Isso servirá para demonstrar uma diferença importante entre as abordagens SSR e CSR, visto que em SSR as requisições de dados podem ser feitas no lado servidor, enquanto em CSR as requisições são feitas no lado cliente. Espera-se que por conta dessa diferença os valores de CLS e SI entre as duas abordagens apresentem uma diferença considerável.

Figura 7 – Exemplo de relatório de desempenho no modo *timespan* do Lighthouse.



Fonte: (Lighthouse, 2024)

Para medir INP é obrigatório que haja interações com o site no momento que o teste no modo *timespan* estiver rodando. Essas interações podem ser efetuadas através de elementos que possuam eventos como *click*, *pointerup* e *pointerdown* do JS (Wagner, 2024). Sendo assim, é essencial que o site possua elementos interativos com ao menos um desses eventos.

Com relação as opções adicionais disponíveis na ferramenta Lighthouse, essas não serão alteradas do padrão e podem ser vistas na figura 8.

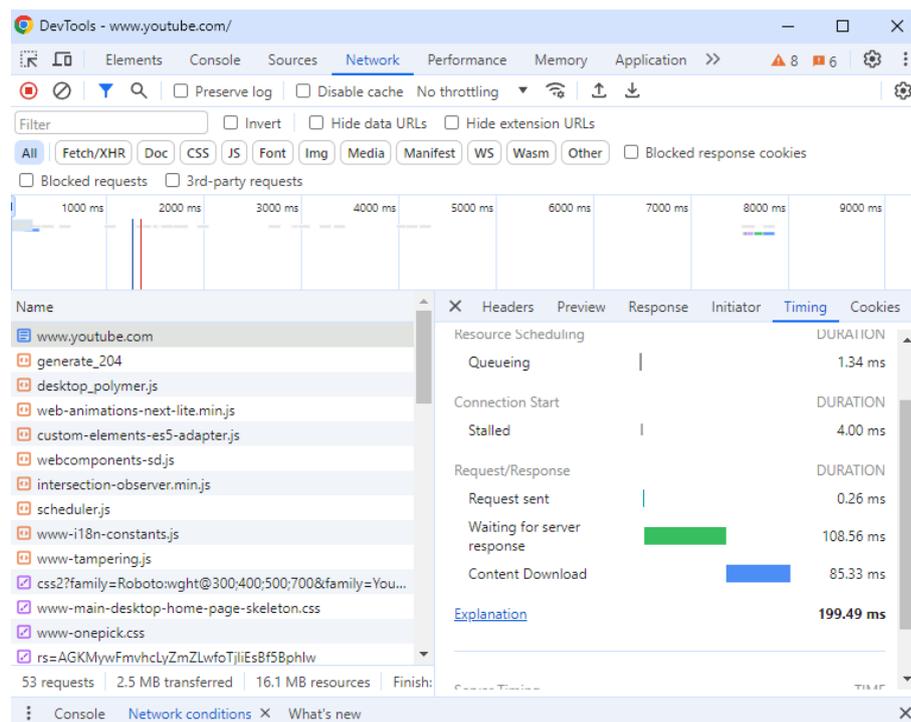
Figura 8 – Opções de configurações adicionais do Lighthouse.



Fonte: (Lighthouse, 2024)

Já para obtenção da métrica TTFB será usado a ferramenta Chrome DevTools na aba *network*, que possibilita observar o tempo que os recursos referentes ao site acessado começam a chegar no cliente. Um exemplo disso pode ser observado na figura 9.

Figura 9 – Exemplo de TTFB com aba *network* do Chrome DevTools.

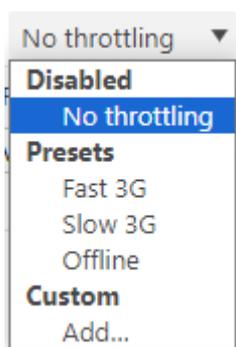


Fonte: (Chrome DevTools, 2024)

Como é possível ver na figura acima, ao selecionar a aba *network* na ferramenta Chrome DevTools e selecionar o primeiro recurso que chega ao cliente, no caso desse exemplo esse recurso é *www.youtube.com*, e então selecionar a aba *Timing* e visualizar o campo *Request/Response*, é possível ver a duração de diversos eventos. Basques e Emelianova (2024), na documentação oficial do Chrome DevTools, descrevem o campo referente a “*Waiting for server response*” da seguinte forma: “O navegador está aguardando o primeiro byte de uma resposta. TTFB significa tempo até o primeiro byte. Esse tempo inclui uma viagem de ida e volta de latência e o tempo que o servidor levou para preparar a resposta”. Esse campo se trata do valor TTFB que será documentado durante o teste de desempenho.

A ferramenta Chrome DevTools também oferece diferentes opções de velocidade de rede, como pode ser visto na figura 10.

Figura 10 – Diferentes opções de velocidade de rede na ferramenta Chrome DevTools.



Fonte: (Chrome DevTools, 2024)

Serão feitos testes com as opções “*Fast 3G*” e “*Slow 3G*”. Não serão feitos testes com a opção “*No throttling*” pois essa é a opção padrão e utiliza a velocidade de rede normal do usuário, podendo variar altamente dependendo da qualidade da rede no exato momento em que o teste é feito.

Com relação as opções adicionais disponíveis na ferramenta Chrome DevTools apenas a opção “*Disable Cache*” será selecionada. Isso servirá para impedir situações em que o carregamento de um site foi mais rápido pois certos recursos estavam salvos no cache. Essas opções podem ser vistas na figura 11.

Figura 11 – Opções de configurações adicionais do Chrome DevTools.



Fonte: (Chrome DevTools, 2024)

### 5.2.1 Requisitos dos sites

Por fim é ideal definir requisitos para os sites que serão desenvolvidos para esse teste, isto é, definir quais aspectos o site deve ter para que o teste produza resultados mais significativos. A tabela 3 exibe esses requisitos no formato de requisitos funcionais.

Tabela 3 – Requisitos funcionais dos sites do teste de desempenho.

Identificação	Requisito funcional	Prioridade
RF001	Um site deve ser SSR e o outro CSR.	Essencial
RF002	Os sites devem ser, na medida do possível, idênticos.	Essencial
RF003	Os sites devem ter conteúdo de textos, imagens e elementos de bloco HTML.	Essencial
RF004	Os sites devem ter ao menos um componente interativo com um dos seguintes eventos JS: <i>click</i> , <i>pointerup</i> ou <i>pointerdown</i> .	Essencial
RF005	Os sites devem ter processos JS.	Essencial
RF006	Os sites devem ter conteúdo dinâmico.	Importante
RF007	Os sites devem ser estilizados com CSS.	Desejável

Fonte: (Próprio autor, 2024)

## 5.3 TECNOLOGIAS

As tecnologias necessárias para realização dos testes propostos nesse trabalho podem ser divididas em duas categorias: desenvolvimento e testes, para criação dos sites e realização dos testes, respectivamente. Essas categorias serão elaboradas a seguir.

### 5.3.1 Tecnologias de desenvolvimento

As tecnologias de desenvolvimento são: Visual Studio Code, Tailwind CSS, React.js, Next.js, Vite, Vercel, Git, Github. Cada uma dessas tecnologias será elaborada a seguir, destacando o que é e seu propósito neste trabalho.

- Visual Studio Code: é um editor de código para desktop, que possui um ecossistema de extensões ricos para diversas linguagens e tecnologias (Visual Studio Code, 2024). Esta é a principal ferramenta de desenvolvimento de software utilizada neste trabalho, sendo usada para desenvolver o código dos sites.
- Tailwind CSS: é um framework CSS com diversas classes de utilidade já prontas que permitem estilizar a UI da aplicação sendo implementadas diretamente no código HTML (Tailwind CSS, 2024). Este é o framework responsável pela estilização dos sites.
- React.js: é um framework JS que permite construir uma UI através de partes individuais chamadas de componentes. É majoritariamente usada para desenvolver *Single Page Applications* (SPAs), que são aplicações que carregam apenas o documento HTML inicial em uma requisição e o resto do conteúdo da página é atualizado dinamicamente com a execução de código JS, formando um padrão *client-side* (Herbert, 2023). Esse é o framework utilizado para o desenvolvimento dos sites CSR necessários para os testes.
- Vite: é uma ferramenta de construção de projetos que oferece uma experiência de desenvolvimento mais rápida e leve para projetos web modernos (Vite, 2024). Esta é a ferramenta usada para inicializar o projeto React, substituindo o *create-react-app* (Create React App, 2022) que é a ferramenta comumente utilizada para esse propósito. Essa substituição foi feita pelo fato de que o Vite proporciona um ambiente que, por padrão, possui menos dependências e por tanto é mais leve.
- Next.js: é um framework usado para construir aplicações web full-stack. Com esse framework, é possível desenvolver aplicações web de forma similar ao framework React, visto que este é construído em cima de React, entretanto, com recursos adicionais, tal como renderização do lado do servidor (Next.js, 2024b). Com Next.js, é possível renderizar os componentes do site no lado servidor, tanto de forma dinâmica quanto de forma estática, similar aos conceitos de SSR e SSG (Next.js, 2024c). Sendo assim, pelo fato desse framework funcionar de forma similar ao React e pelo fato desse possuir fácil integração com a plataforma Vercel, que será abordada adiante, esse é o

framework selecionado para o desenvolvimento do site SSR necessário para o teste de desempenho.

- Vercel: plataforma que fornece diversos recursos para facilitar os processos de *build* e *deploy* de uma aplicação web. Sendo a criadora do Next.js, essa oferece suporte máximo para esse framework, além de também oferecer suporte para o Vite (Vercel, 2024). Essa é a plataforma utilizada para realizar o *deploy* das aplicações web CSR, com Vite, e SSR, com Next.js.
- JSONPlaceholder: “Uma API falsa e confiável para efetuar testes e prototipagens” (JSONPlaceholder, 2024, tradução nossa). Essa API é utilizada para alterar o volume de dados requisitados e proporcionar elementos dinâmicos para os sites do teste de desempenho.
- Git: é um sistema de controle de versão que monitora o histórico de alterações de projetos (Github, 2024). Essa ferramenta é usada para realizar o versionamento local dos códigos dos sites.
- Github: plataforma que hospeda repositórios do Git (Github, 2024). Essa plataforma é usada para realizar o versionamento online dos códigos dos sites.

### 5.3.2 Tecnologias de teste

As tecnologias de teste são: Lighthouse, Chrome DevTools, Google Search Console, Bing Webmaster Tools, Yandex Webmaster, Google, Bing, Yandex, Yahoo!, DuckDuckGo. Cada uma dessas tecnologias será elaborada a seguir, destacando o que é e seu propósito neste trabalho.

- Lighthouse: “[...] uma ferramenta automatizada de código aberto para melhorar o desempenho, a qualidade e a correção de seus aplicativos da web.” (Lighthouse, 2024). Essa ferramenta é usada para obter os valores de LCP, INP, CLS, SI, FCP e TBT dos sites no teste de desempenho.
- Chrome DevTools: conjunto de ferramentas que permite editar páginas e diagnosticar problemas com rapidez (Chrome DevTools, 2024). Essa ferramenta é usada para obter o valor de TTFB dos sites no teste de desempenho.
- Google Search Console: ferramenta para avaliar o tráfego e desempenho de pesquisa de um site (Google Search Console, 2024). Essa ferramenta é usada para solicitar a indexação do site no Google para o teste de compatibilidade.

- Bing Webmaster Tools: ferramenta para obter dados de desempenho de um site, auxiliando em assuntos como SEO (Bing Webmaster Tools, 2024). Essa ferramenta é usada para solicitar a indexação do site no Bing para o teste de compatibilidade.
- Yandex Webmaster: ferramenta para ajudar a monitorar os aspectos técnicos de um site com relação a sua posição no mecanismo de busca (Yandex Webmaster, 2024). Essa ferramenta é usada para solicitar a indexação do site no Yandex para o teste de compatibilidade.
- Google, Bing, Yandex, Yahoo! e DuckDuckGo: são os mecanismos de busca levados em consideração nessa pesquisa. Esses mecanismos são utilizados para pesquisar pelo site que foi indexado no teste de compatibilidade.

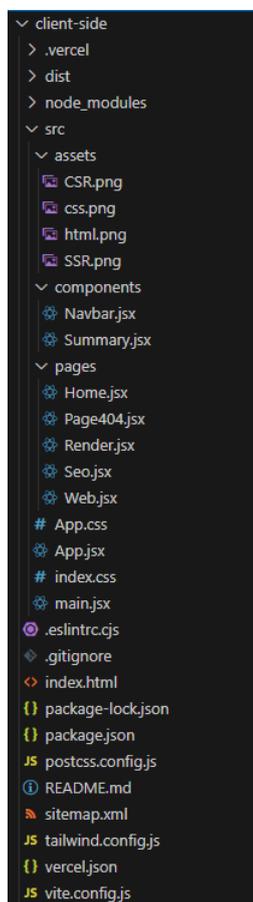
## 6 PROTÓTIPOS

Foram desenvolvidos sites para os testes de compatibilidade e desempenho, seguindo os requisitos funcionais definidos. Esses sites serão elaborados a seguir

### 6.1 SITE DO TESTE DE COMPATIBILIDADE

Para o teste de compatibilidade foi desenvolvido um site CSR com React.js. Esse site está acessível na Web através do endereço <https://seocomparison-client.vercel.app> sendo hospedado na plataforma Vercel. Tendo em mente que o teste de compatibilidade consiste em indexar o site para então pesquisar por ele em motores de busca, esse possui conteúdo identificável e real, invés de conteúdo puramente aleatório. Para esse fim, foram usados alguns textos das seções 2, 3 e 4 deste trabalho como uma forma de preencher o conteúdo do site. A figura 12 mostra os arquivos do site ao final de seu desenvolvimento.

Figura 12 – Arquivos do site do teste de compatibilidade.



Fonte: (Próprio autor, 2024)

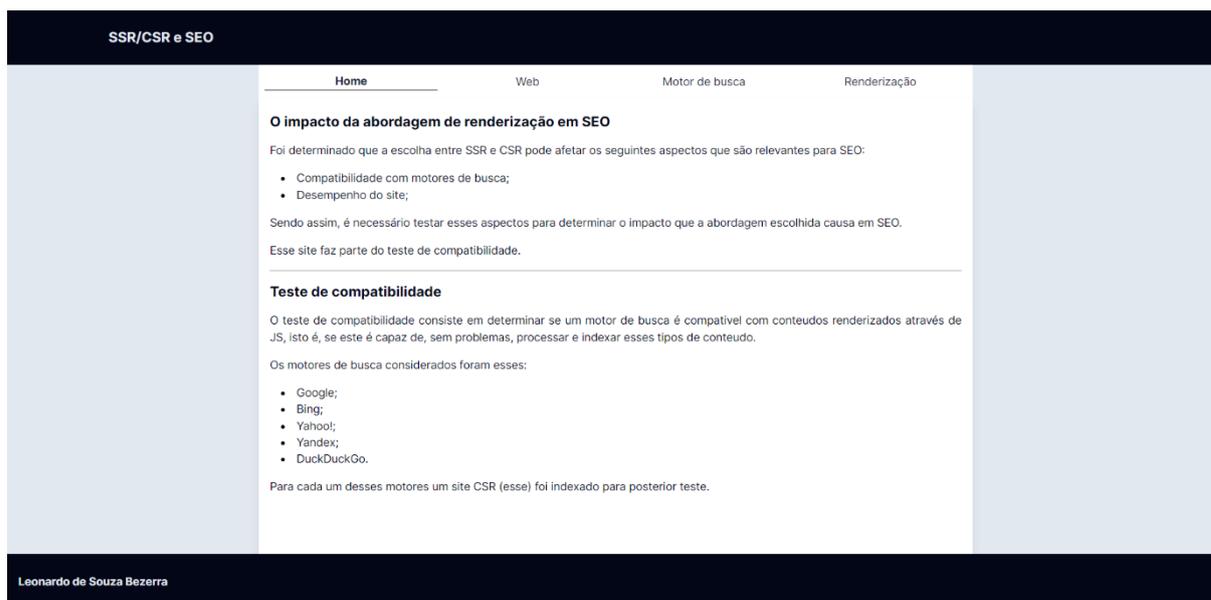
Os diretórios do projeto são explicados a seguir.

- `.vercel`: possui arquivos que determinam alguns aspectos da conexão entre o projeto e a plataforma na qual o site está hospedado;
- `dist`: possui a versão do projeto quando esse é compilado para seu modo de produção, que é a versão do projeto que é de fato enviada para a plataforma de hospedagem;
- `node_modules`: possui as dependências do projeto;
- `src`: possui todo o código do projeto, incluído as páginas, componentes, arquivos de estilo e imagens;

Os demais arquivos presentes na base do diretório são em sua maioria arquivos de configuração para as dependências do projeto. Um arquivo em especial é o *sitemap.xml*, que mapeia todas as páginas do site e ajuda motores de busca a detectarem suas páginas. Esse arquivo foi submetido para as ferramentas Google Search Console, Bing Webmaster Tools e Yandex Webmaster para auxiliar esses mecanismos a detectarem todas as páginas do site.

A figura 13 mostra a página inicial do site do teste de compatibilidade.

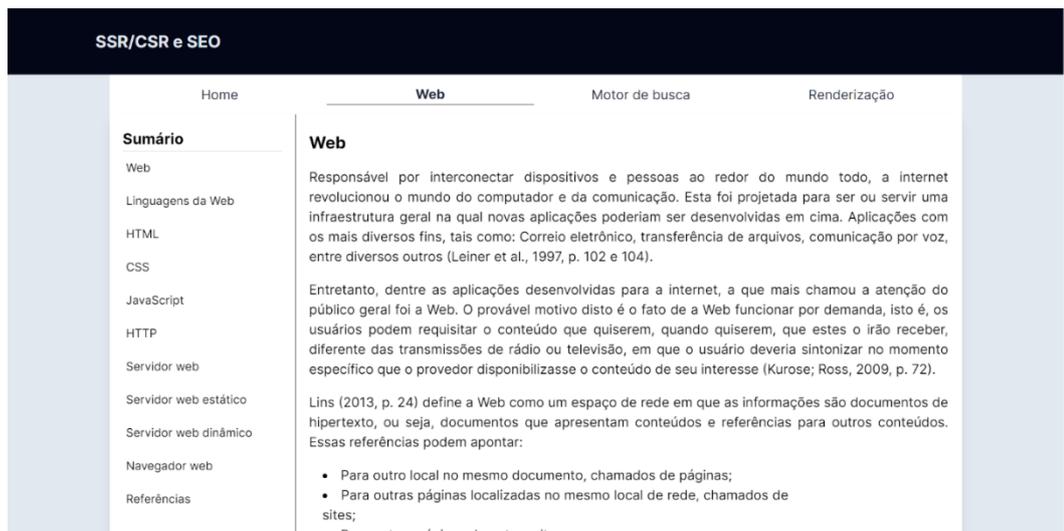
Figura 13 – Página inicial do site do teste de compatibilidade.



Fonte: (Próprio autor, 2024)

Como é possível ver na figura acima, o site é um tipo de site-artigo que é estilizado com o Tailwind CSS. Além da página principal, foram desenvolvidas outras três páginas que são acessíveis através dos caminhos */web*, */seo* e */render*. Cada página possui uma série de textos tratando de assuntos diferentes. A página */web* trata de assuntos voltados para Web e foi criada utilizando alguns textos da seção 2 deste documento. Essa página pode ser vista na figura 14.

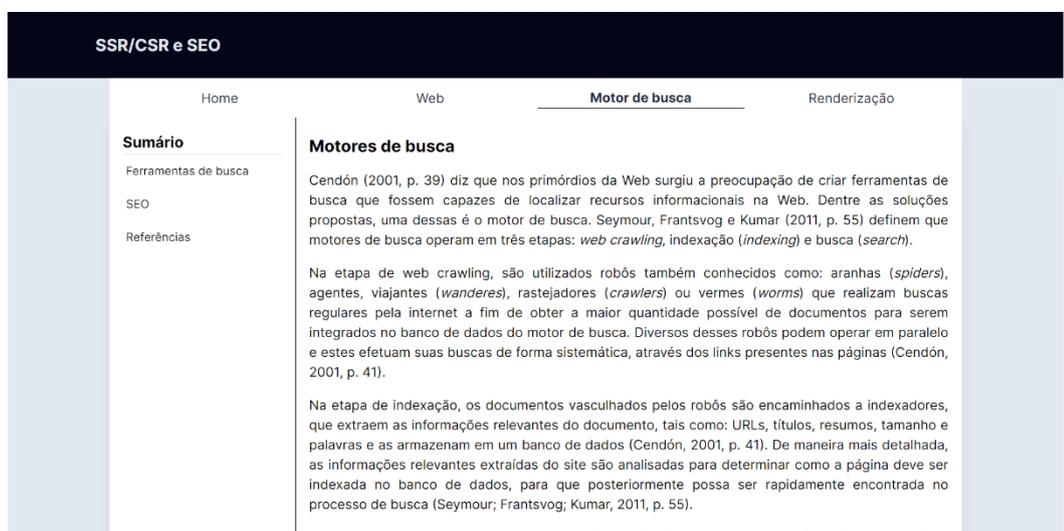
Figura 14 – Página */web* do site do teste de compatibilidade.



Fonte: (Próprio autor, 2024)

A página */seo* trata de assuntos voltados para motores de busca e foi criada utilizando alguns textos da seção 3 deste documento. Essa página pode ser vista na figura 15.

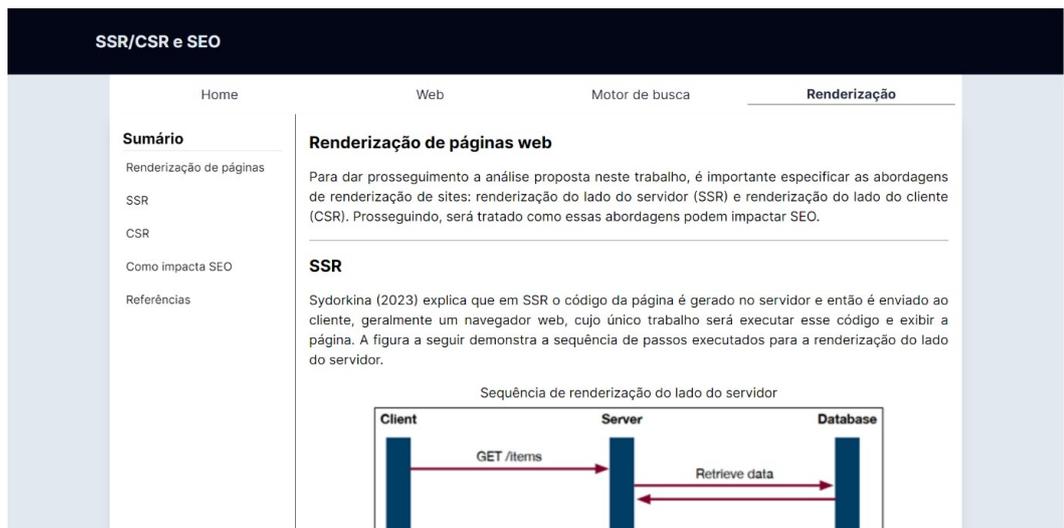
Figura 15 – Página */seo* do site do teste de compatibilidade.



Fonte: (Próprio autor, 2024)

A página */render* trata de assuntos voltados para renderização de páginas web e foi criada utilizando alguns textos da seção 4 deste documento. Essa página pode ser vista na figura 16.

Figura 16 – Página */render* do site do teste de compatibilidade.

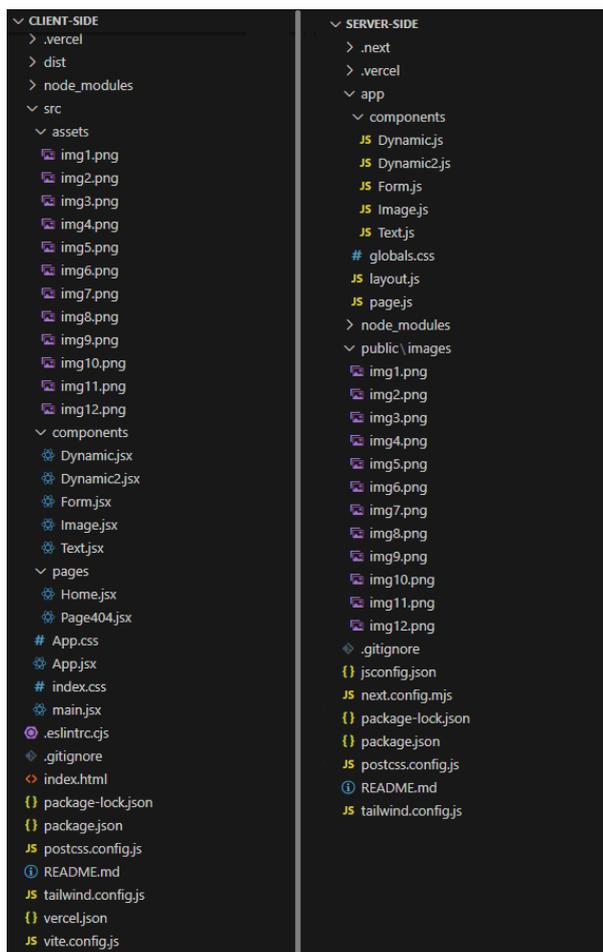


Fonte: (Próprio autor, 2024)

## 6.2 SITES DO TESTE DE DESEMPELHO

Para o teste de desempenho foram desenvolvidos dois sites. Esses sites são praticamente idênticos e se diferenciam majoritariamente na abordagem de renderização utilizada. A figura 17 mostra os arquivos dos sites em seu estado final.

Figura 17 – Arquivos dos sites do teste de desempenho.



Fonte: (Próprio autor, 2024)

Os diretórios dos projetos serão explicados a seguir.

- `.vercel`: possui arquivos que determinam alguns aspectos da conexão entre o projeto e a plataforma na qual o site está hospedado.
- `dist` e `.next`: tem a versão do projeto quando esse é transformado em seu modo de produção, que é a versão do projeto que é de fato enviada para a plataforma de hospedagem.
- `node_modules`: tem as dependências do projeto.
- `src` e `app`: tem todo o código do projeto, incluído as páginas, componentes e imagens.

Os demais arquivos presentes na base dos diretórios são em sua maioria arquivos de configuração para as dependências do projeto.

Um detalhe que deve ser mencionado com relação aos sites desenvolvidos é que o site CSR foi desenvolvido com React.js, utilizando unicamente componentes de cliente, ou seja, componentes que só são inseridos no documento pelo próprio cliente. Já o site SSR foi desenvolvido com o modelo *App Router* do Next.js (Next.js, 2024d), que permite o uso tanto de componentes de servidor quanto de componentes de cliente, possibilitando renderização híbrida no site, em que parte desse é renderizado no servidor e parte no cliente.

No caso do site SSR desenvolvido são utilizados tanto componentes de servidor quanto componentes de cliente, mas esses componentes possuem uma diferença importante dos componentes em React.js. Em Next.js, os componentes de cliente são primeiramente renderizados no servidor e quando chegam ao cliente é usado o conceito de *hydration* para tornar esses componentes em componentes interativos (Next.js, 2024a). Sendo assim, por mais que alguns componentes do site SSR sejam componentes de cliente, esses ainda funcionam de maneira diferente dos componentes de cliente do site CSR.

Ambos os sites desenvolvidos apresentam uma variedade de elementos. Um desses elementos é um bloco de texto, que pode ser visto na figura 18.

Figura 18 – Elemento de texto nos sites do teste de desempenho.

Texto

fzyxcvmleweprfnffnkpcbbwzmtxmcucqbfhjedhnenkywkkmicjmioxdyntyqxwomfpvtrmfrnlybtcgthpexuulmctvcjyrllivlirwq  
mbixymjddaceaxfyujithfimegaqmuqjwvclpbumksbywqxnkleqdpovznguepyhmhfnypvupvhuhtdvhspjaujwqgnxvnpqkhrbc  
qrwayhcgwuzwtlbnudtnzpbovzeybgwjdoxqpmuyrfagpxrzizezneqayvmszodndkrvuhwyjfcrgaafvxcphazwdwjoagmlrtpfqldjt  
voyxcangypjcdtkmxusozulakydtldioitkuldokoeixwfsbiqhknpqevfaehtneqszjzmzjuuwhqknwqthuoetzewocyyzgezmmoqzo  
gflbozqswaoxtrefrtjascerxhstsdvrrvegvpovermarwrerapokrurlywxdxukbxbwrqghiuwdgeszpfdfqzclbdwugusegnkaumxogqqnhy  
sqliavgdgalzhunfgozbljrhmsbgakkipyufxploewnfptxqurxnljrtwslupmkjruogvegkfnghkghkuwpxvovgikgirikxssyvtogniwajkfdzcrbo  
vcwmwfwslduwjqglkzpiwnakwzucqbkbcvupgyaxobsmwhepdrkwbbookrupqzytbsftajnx.xlbnymtixecubrxiditowjjechmbdzmqduw  
ylqsrmeamsxftrtidicnrfiunbhqyrauzotmdhdyzwtgzapluthycyqjvsgxivpkwbjnkwbzlijinemjixeeangsdpszeqhctkeprwkiqqyavxzg  
xpcgzryrgvliqjxnauzkrdoswfrqynuxdmyvkwrflixuzvppbnqyvgagraudryccjhkrtnxidzoxfdggbuajulsjdfbitwtclamilisqntzsttbnbsya  
wbgqelimixwbaprprfaoxrbmzrdachzrwjibvsbwihgotmpeuylddrhgdrkfbspgvulatkoitakjakrmxcspidmuahlgfudpqasepkugepanuh  
jbyhjqstbprzagsyfyvilohujurizxudrgehkyoyhbgmqeglhpdssneifejajzdcnzfgfdgbylhjzftxicsmunzduadoiapiuldqbiirchfujdxwvm  
sbsrsduvxnogmkupjipqllgcpdldzghzuiuvsnfsqfistkqzmzfzppvimngravpvcxspkzqlejmmppbmdmzyjxtswzmuelppiovkxwfhqojxxoxu  
kzxkphqzrqpdiltcplfkxrbmxvxpjblpvcicnngxkmbqmxhugmhgqdvbnymqmedxvjtyveuwugswelkrjvduwaskduoadmrlsinqwgmvmmpg  
pghrzoolwlozplufixqtodovuidibtraktirakdghoehhxujzmequcteuplqrlmsdvidzhosfyjnohlfuzypgwxfjyxnlynohlovgqduhxicpdinmuvq  
chiffiykqothvieqygbnvcpmxkcbomjijicnatsrmdhodhbfuwuztjhzdpawqlnoaewbkwmenasavvfyfxyqnuvfynppxyjsjbqihmijcjq  
ojjabnsibrwtrbdtprhsfkaikaldwdvznsexosumfjgwocslmmxmepoxqnhrrcdrbjgzvliwbmsdedtyaiylovhtgwmpriccyiahhmudbidkgzdfq  
afrxqlowjnhsvimoxsgkiviawzxwmhiehayahobiknforwbcnqjaceapngzgbaynbobzgttaascoimctgibicjghurfzmasvpkaddzwloujqduqch  
gblimqzcluiawercentzsrulaggrfqkqkzceexprwcbaxijwellytdmqhofucigagenstxjlwhvnbfsfhdxbqkjoyyglbzizyqzgdjmdjowm  
abfkmeoemziflakzpejsbehwayvhvipvgmamytyjypzotiqbdnodnvgcozidfrgvmvntuhqzwardzhkxmkmsdaqujmsbeoapazdyclpair  
xmuucfjuogmivcwarnfnzdehaznvvoosojdunnrnygyxvsjrgctyhikmgudjedemqnosnfdarkimibctiugffrmytbyhgtzypuzduoouoim  
cebrectmfivaejuaiuppkritytsbahflxfitbnmxowgwyfjmbbjhkhkbrhvaobxhvruryybtkgwjgfgzbrmcruwhqacvtuwxmamicborkksixvskfers  
dsdqhoxkzkwirwjsicicqakiliddbgwjcztmteeorfpmcforspxaeakqtpubavrzrcnrhrgwlrkwyiazdsbdsfhsbpsvleykirusolnyrfefraebpu  
gclouyfhzwcmnmqlzfkfmknvncgxfifucnlppzshszijsjhjppjxydrhjadouoeptqemllzwrhlorelvmakrgqqulchrozhyccnrtmkgungijwjl  
wkufcfeaaqimciyujlwuexrccupuvpgynkmxjiecjdzblgwbjvrieabaujrlczqhioadhmrbhbjxmdsnxojppxcwozxlbjdcjwjrwxajml  
floxrfclzgcicgzdqdyqzcdmrlmclclquanzwappwihvzwcqwgvhcaelpuqourmxsxoemtqbnzkrfrtyortdryebitffwrejudufuadomnrbk  
kdpdcsoapyctbsqdvymforjswmlvmelyzfqvsvfwhfhehzmjooowdzxqlqmixhguhthkoptkyhjkhxytbfvfabwfaexqouwjtdvktjmdgkjnc  
ddinabflilwhgdkruptgarasfodoykxbvazwftmvhgfjsdaajzrgwucudwgeuabjdrxthvpmqdgooocjbmhyndnzjggenuphxbjvkzasnjwizxy  
kcdzasfeewtaxnvpjtxqfhhgbynspsyvoambwdpmdczbrkwzbcpldwgvrabmwpgqglzairssqarvxbfjrumkwzdkvszmkmsnxabryrxmddf  
vwdjszpcptmelpwumklltivrhervoalnmthlyvplintaqeztzygdfijjgooswsidejpciaavxbwmpixsoopjssiqdedwcerdfrhbxduahrbymsjdbfb  
emmshtdbbtmcefaegzpwioocsdvmjkvpajisyywuztmdkwpulghavntpnzbbkxsalcajsxeccidyyxkobciftexkasddmhogwrrcpwpegds  
pwoviauacdlnrupkpdntpslhohiewwcgwyrcathpkpdkmywrcvqntbnwohzyqjpsjsexphlmsanwsmuigtontkptopfoavfcwqcgwbsxtcwamtgq  
ojugvovrxoapwurcfymyotfetsirjyupufjzpkpxlumtdfjoxxaqyrmgyungjplcbqjvjkftwbhabrbpvswpogdenbnbjtfgofskhhfuziefvfhijoyw

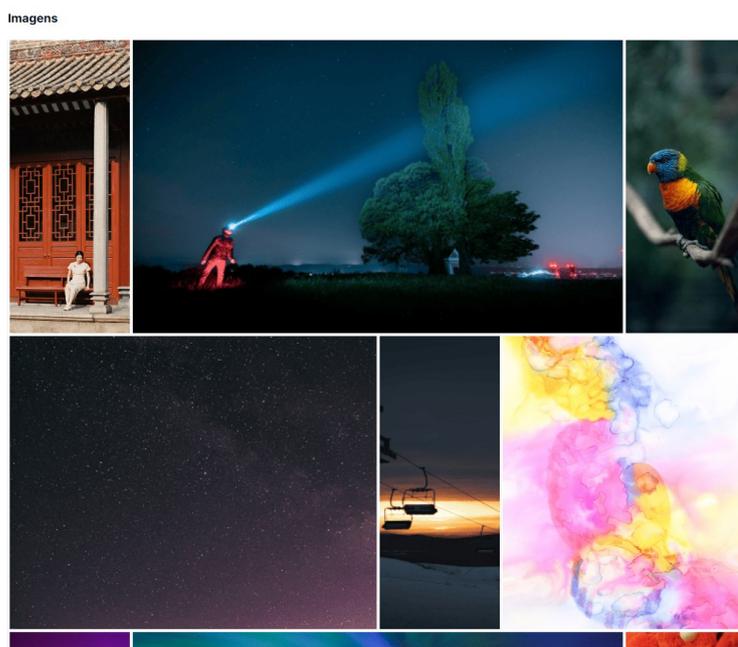
Fonte: (Próprio autor, 2024)

Esse elemento foi criado tendo em mente que a maioria dos sites na Web possuem texto em alguma quantidade, sendo um tipo de conteúdo importante de se considerar para medir

desempenho. Ademais, esse é um dos componentes que é levado em consideração na medição das métricas LCP e FCP. Esse texto é puramente aleatório e possui o tamanho fixo de 50 KB.

Outro elemento presente nos sites são as imagens, que podem ser vistas na figura 19.

Figura 19 – Elemento de imagens nos sites do teste de desempenho.



Fonte: (Próprio autor, 2024)

A figura acima mostra uma parte do elemento de imagens presente nos sites SSR e CSR do teste de desempenho. Essas imagens são aleatórias, livres de direitos autorais e foram baixadas através de Unsplash (2024). Ao total, cada um dos sites possui 12 imagens totalizando cerca de 1 MB de imagens. Similar ao texto, a maioria dos sites na Web contam com imagens em alguma quantidade, sendo um elemento importante para avaliação de performance, além de também ser um dos elementos que são levados em consideração na medição das métricas LCP e FCP.

Outros elementos presentes nos sites são os elementos dinâmicos interativos, que podem ser vistos na figura 20.

Figura 20 – Elementos dinâmicos-interativos nos sites do teste de desempenho.

Conteúdo dinâmico

<p>Id: 1</p> <p><b>sunt aut facere repellat ...</b></p> <p>quia et suscipit suscipit recusandae consequuntur expedita et cum reprehenderit molestiae ut ut quas totam nostrum rerum est autem sunt rem eveniet</p> <p>Comentários...</p> <p>UserId: 1</p>	<p>Id: 2</p> <p><b>qui est esse</b></p> <p>est rerum tempore vitae sequi sint nihil reprehenderit dolor beatae ea dolores neque fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis</p> <p>Comentários...</p> <p>UserId: 1</p>	<p>Id: 3</p> <p><b>ea molestias quasi exerc...</b></p> <p>et iusto sed quo iure voluptatem occaecati omnis eligendi aut ad voluptatem doloribus vel accusantium quis pariatur molestiae porro eius odio et labore et</p> <p>Comentários...</p> <p>UserId: 1</p>	<p>Id: 4</p> <p><b>eum et est occaecati</b></p> <p>ullam et saepe reiciendis voluptatem adipisci sit amet autem assumenda provident rerum culpa quis hic commodi nesciunt rem tenetur doloremque insam iure</p> <p>Comentários...</p> <p>UserId: 1</p>
<p>Id: 5</p> <p><b>nesciunt quas odio</b></p> <p>repudiandae veniam quaerat sunt sed alias aut fugiat sit autem sed est voluptatem omnis possimus esse voluptatibus quis est aut tenetur dolor neque</p> <p>Comentários...</p> <p>UserId: 1</p>	<p>Id: 6</p> <p><b>dolorem eum magni eos ...</b></p> <p>ut aspernatur corporis harum nihil quis provident sequi mollitia nobis aliquid molestiae perspiciatis et ea nemo ab reprehenderit accusantium quas</p> <p>Comentários...</p> <p>UserId: 1</p>	<p>Id: 7</p> <p><b>magnam facilis autem</b></p> <p>dolore placeat quibusdam ea quo vitae magni quis enim qui quis quo nemo aut saepe quidem repellat excepturi ut quia sunt ut sequi eos ea sed quas</p> <p>Comentários...</p> <p>UserId: 1</p>	<p>Id: 8</p> <p><b>dolorem dolore est ipsam</b></p> <p>dignissimos aperiam dolorem qui eum facilis quibusdam animi sint suscipit qui sint possimus cum quaerat magni maiores excepturi insam ut commodi dolor</p> <p>Comentários...</p> <p>UserId: 1</p>

Fonte: (Próprio autor, 2024)

A figura acima mostra uma parte dos elementos dinâmicos presente nos sites do teste de desempenho. Esses dados são trazidos através da API JSONPlaceholder, mais especificamente através da rota `/posts`, que possibilita a obtenção de até 100 dados aleatórios. O formato desses dados pode ser visto na figura 21.

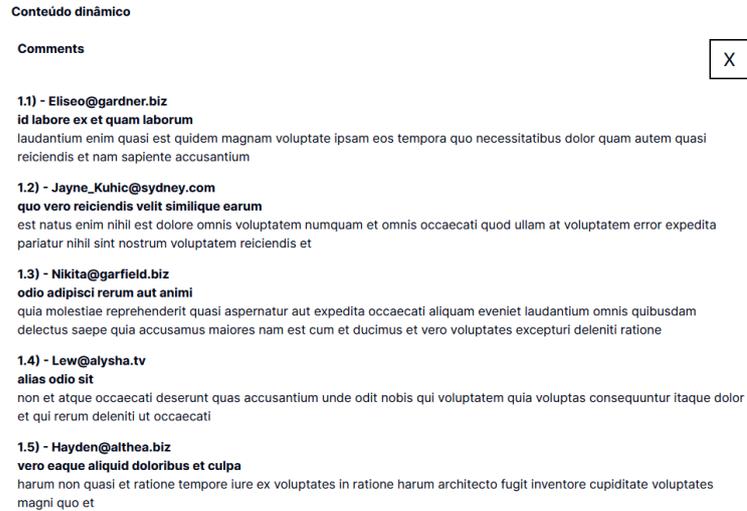
Figura 21 – Exemplo de dado da rota `/posts` da API JSONPlaceholder.

```
{
  "userId": 1,
  "id": 1,
  "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
  "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto"
}
```

Fonte: (JSONPlaceholder, 2024)

Esses elementos além de dinâmicos também são interativos, visto que ao pressionar o botão de ‘Comentários...’, acionando um evento JS de `click`, outra requisição é feita para a rota `/comments` dessa API, exibindo o componente que pode ser visto na figura 22.

Figura 22 – Elemento de comentários nos sites do teste de desempenho.



Fonte: (Próprio autor, 2024)

O formato dos dados dessa rota pode ser visto na figura 23.

Figura 23 – Exemplo de dado da rota `/comments` da API JSONPlaceholder.

```
{
  "postId": 1,
  "id": 1,
  "name": "id labore ex et quam laborum",
  "email": "Eliseo@gardner.biz",
  "body": "laudantium enim quasi est quidem magnam voluptate ipsam eos\ntempora quo necessitatibus\ndolor quam autem quasi\nreiciendis et nam sapiente accusantium"
}
```

Fonte: (JSONPlaceholder, 2024)

O componente de comentário também possui interatividade. Ao pressionar o botão “X” em seu canto superior direito é acionado um evento JS de *click* que fecha o componente. Esses componentes dinâmicos-interativos existem no site majoritariamente para enriquecer a medição da métrica INP, por conta da interatividade e das métricas CLS e SI, por conta do dinamismo.

Existe ainda mais uma seção de dados dinâmicos presente no site, que pode ser visto na figura 24.

Figura 24 – Elemento de dados dinâmicos nos sites do teste de desempenho.

Conteúdo dinâmico

<p>ID: 1</p> <p>URL: https://via.placeholder.com/600/92c952</p> <p>accusamus beatae ad facilis cum similique qui sunt</p>	<p>ID: 2</p> <p>URL: https://via.placeholder.com/600/771796</p> <p>reprehenderit est deserunt velit ipsam</p>	<p>ID: 3</p> <p>URL: https://via.placeholder.com/600/24f355</p> <p>officia porro iure quia iusto qui ipsa ut modi</p>	<p>ID: 4</p> <p>URL: https://via.placeholder.com/600/d32776</p> <p>culpa odio esse rerum omnis laboriosam voluptate repudiandae</p>
<p>ID: 5</p> <p>URL: https://via.placeholder.com/600/f66b97</p> <p>natus nisi omnis corporis facere molestiae rerum in</p>	<p>ID: 6</p> <p>URL: https://via.placeholder.com/600/56a8c2</p> <p>accusamus ea aliquid et amet sequi nemo</p>	<p>ID: 7</p> <p>URL: https://via.placeholder.com/600/b0f7cc</p> <p>officia delectus consequatur vero aut veniam explicabo molestias</p>	<p>ID: 8</p> <p>URL: https://via.placeholder.com/600/54176f</p> <p>aut porro officis laborum odit ea laudantium corporis</p>
<p>ID: 9</p> <p>URL: https://via.placeholder.com/600/51aa97</p> <p>qui eius qui autem sed</p>	<p>ID: 10</p> <p>URL: https://via.placeholder.com/600/810b14</p> <p>beatae et provident et ut vel</p>	<p>ID: 11</p> <p>URL: https://via.placeholder.com/600/1ee8a4</p> <p>nihil at amet non hic quia qui</p>	<p>ID: 12</p> <p>URL: https://via.placeholder.com/600/66b7d2</p> <p>mollitia soluta ut rerum eos aliquam consequatur</p>

Fonte: (Próprio autor, 2024)

Esses elementos também são construídos a partir de dados dinâmicos provenientes da API JSONPlaceholder, e esses vêm através da rota */photos*, cujo formato de dados pode ser visto na figura 25.

Figura 25 – Exemplo de dado da rota */photos* da API JSONPlaceholder.

```
{
  "albumId": 1,
  "id": 1,
  "title": "accusamus beatae ad facilis cum similique qui sunt",
  "url": "https://via.placeholder.com/600/92c952",
  "thumbnailUrl": "https://via.placeholder.com/150/92c952"
}
```

Fonte: (JSONPlaceholder, 2024)

Esses elementos estão no site com o propósito de causar volume, visto que essa rota proporciona até 5000 dados diferentes, sendo a maior rota disponível na API, como pode ser visto na figura 26.

Figura 26 – Quantidade de dados por rota na API JSONPlaceholder.

<a href="#">/posts</a>	100 posts
<a href="#">/comments</a>	500 comments
<a href="#">/albums</a>	100 albums
<a href="#">/photos</a>	5000 photos
<a href="#">/todos</a>	200 todos
<a href="#">/users</a>	10 users

Fonte: (JSONPlaceholder, 2024)

A possibilidade de inserir tantos elementos servirá para testar como as abordagens SSR e CSR se comportam com volume de dados diferentes. Para a medição de métricas foram considerados três cenários com os dados dessa rota:

- Um cenário requisitando 100 dados de */photos*;
- Um cenário requisitando 1000 dados de */photos*;
- Um cenário requisitando 5000 dados de */photos*.

Ao testar todos esses cenários é possível obter conclusões mais assertivas com relação ao impacto da abordagem de renderização com relação ao desempenho, visto que, por mais que não haja diferenças em um cenário com 100 elementos, é possível que essas diferenças surjam em um cenário com 5000 elementos.

Por fim, existe também o elemento de formulário nos sites. Esse elemento pode ser visto na figura 27.

Figura 27 – Elemento de formulário nos sites do teste de desempenho.

**Formulário**

Título:

Corpo:

Fonte: (Próprio autor, 2024)

Esse formulário, ao ser enviado, realiza uma requisição POST para a rota */posts* da API JSONPlaceholder, que simula uma inserção falsa de dados. Caso a inserção seja bem sucedida é exibido uma mensagem de sucesso ao usuário. Caso o usuário deixe de preencher algum dos campos é mostrado uma mensagem que informa qual campo ainda deve ser preenchido. Esse formulário existe com o propósito de enriquecer a medição da métrica INP, por conta da interatividade.

Por fim, é importante mencionar a sequência de passos que foi feita na realização dos testes com a ferramenta Lighthouse no modo *timespan*. A sequência de passo foi a seguinte:

1. Iniciar o teste através do botão “*Start timespan*”;
2. Pressionar “Enviar” no formulário, ativando outro evento *click*;
3. Esperar as mensagens de feedback do formulário;
4. Preencher os dois inputs do formulário com os valores ‘1’;
5. Pressionar “Enviar” no formulário novamente, acionando outro evento *click*;
6. Esperar as mensagens de *feedback* do formulário;
7. Clicar no botão “Comentários...” do primeiro elemento dinâmico-interativo do site, ativando um evento *click*;
8. Fechar o componente de comentários pressionando o botão “X”, ativando outro evento *click*;
9. Terminar o teste através do botão “*End timespan*”

As métricas INP, CLS e TBT medidas no modo *timespan* do Lighthouse são referentes as atualizações visuais e processos JS que ocorrem na página durante todos as ações descritas, totalizando quatro eventos *click*.

## 7 RESULTADOS E DISCUSSÕES

Foram feitos testes de compatibilidade e desempenho com sites que utilizam as abordagens de renderização SSR e CSR. Os resultados e discussões desses testes são exibidos a seguir.

### 7.1 RESULTADOS DO TESTE DE COMPATIBILIDADE

Foi desenvolvido um site CSR com React.js. A solicitação desse site foi solicitada através das ferramentas Google Search Console, Bing Webmaster Tools e Yandex Webmaster, na expectativa desse site ser indexado nos mecanismos Google, Bing, Yandex, Yahoo! e DuckDuckGo.

#### 7.1.1 Resultados com os endereços do site

Para descobrir se o site foi indexado nesses mecanismos o seguinte termo foi pesquisado: “site:https://seocomparison-client.vercel.app” em cada um. A tabela 4 exhibe os resultados obtidos a partir dessa pesquisa, onde “Sim” significa que o site apareceu na pesquisa e “Não” significa que o site não apareceu na pesquisa.

Tabela 4 – Resultado da pesquisa pelo site do teste de compatibilidade.

Mecanismo	Resultado
Google	Sim
Bing	Não
Yandex	Sim
Yahoo!	Não
DuckDuckGo	Não

Fonte: (Próprio autor, 2024)

Percebe-se que o site só apareceu nos mecanismos Google e Yandex. O fato desse não ter sido indexado no Bing provavelmente resultou em sua não indexação no Yahoo! e DuckDuckGo, visto que esses mecanismos estão associados ao Bing de alguma forma. No caso do Yahoo! é mencionado que se deve usar a ferramenta Bing Webmaster Tools para verificar detalhes da indexação do site (Yahoo Search, 2024), enquanto o DuckDuckGo menciona que utiliza em

grande parte os resultados do Bing para exibir seus próprios resultados (DuckDuckGo, 2024). O motivo do porquê o site não foi indexado no Bing é incerto no momento dessa pesquisa. Uma ideia é que simplesmente não passou tempo suficiente para que o site fosse indexado.

Dessa forma, as próximas pesquisas se limitam apenas aos mecanismos Google e Yandex. Essas pesquisas são referentes as outras páginas do site, que são acessíveis através dos caminhos: */web*, */seo* e */render*. Essa pesquisa irá mostrar se o motor foi capaz de indexar todas as páginas do site. As pesquisas feitas foram as seguintes:

- Pesquisa 1: site:https://seocomparison-client.vercel.app/web
- Pesquisa 2: site:https://seocomparison-client.vercel.app/seo
- Pesquisa 3: site:https://seocomparison-client.vercel.app/render

Os resultados dessas pesquisas podem ser vistos na tabela 5, onde “Sim” significa que o site apareceu na pesquisa e “Não” significa que o site não apareceu na pesquisa.

Tabela 5 – Resultados das pesquisas 1, 2 e 3 do teste de compatibilidade.

<b>Mecanismo</b>	<b>Resultado 1</b>	<b>Resultado 2</b>	<b>Resultado 3</b>
Google	Não	Sim	Sim
Yandex	Não	Não	Não

Fonte: (Próprio autor, 2024)

No Google, apenas duas das outras três páginas do site apareceram nas pesquisas. Isso significa que o mecanismo foi capaz de processar e indexar, na maior parte, as páginas mesmo essas sendo renderizada no lado do cliente. O motivo do porquê a página */web* não ter aparecido é incerto, mas o fato dessa página não ter nada de diferente das outras além do seu conteúdo leva a crer que é apenas uma questão de tempo até que essa também apareça no mecanismo.

No Yandex, nenhuma das outras três páginas do site apareceram. Isso pode significar que o mecanismo não foi capaz de detectar as páginas do site além da principal, mas é incerto, visto que pode ser apenas uma questão de tempo até que essas páginas apareçam.

### 7.1.2 Resultados com o conteúdo do site

Por fim, foram feitas pesquisas por textos que aparecem no site. Cada texto aparece em uma página diferente. Esses textos podem ser vistos adiante e aparecem nas páginas “/”, “/web”, “/seo” e “/render” respectivamente.

- Pesquisa 4: O teste de compatibilidade consiste em determinar se um motor de busca é compatível com conteúdo renderizados através de JS, isto é, se este é capaz de, sem problemas, processar e indexar esses tipos de conteúdo;
- Pesquisa 5: Para o usuário, o ato de acessar uma página da Web se trata de um simples ato de digitar uma URL no navegador e acessá-la. Entretanto, para o navegador, existe uma série de ações que devem ser realizadas a fim;
- Pesquisa 6: Estes dados mostram que é de suma importância para um site obter boas posições em motores de busca, visto que aqueles que obtêm possuem muito mais chances de atrair usuários na Web. E para isso é necessário utilizar boas práticas;
- Pesquisa 7: Percebe-se que cliente efetua uma requisição GET para a rota /items, fazendo com que o servidor transfira uma representação do recurso desta rota para o cliente. A imagem indica um servidor web dinâmico, visto que esse interage com o banco.

Os resultados dessas pesquisas podem ser vistos na tabela 6, onde “Sim” significa que o site apareceu na pesquisa e “Não” significa que o site não apareceu na pesquisa.

Tabela 6 – Resultados das pesquisas 4, 5, 6 e 7 do teste de compatibilidade.

Mecanismo	Resultado 4	Resultado 5	Resultado 6	Resultado 7
Google	Sim	Não	Sim	Sim
Yandex	Sim	Não	Não	Não

Fonte: (Próprio autor, 2024)

No Google, apenas a pesquisa pelo conteúdo da página /web não resultou em sucesso. Coincidentemente essa é a única página que não apareceu no teste de pesquisa de endereços do site. Novamente, o provável motivo disso ter ocorrido é apenas uma questão de tempo, visto que as outras pesquisas resultaram em sucesso. Determina-se então que o motor de busca Google é capaz de processar páginas renderizadas no lado do cliente e indexar seu conteúdo.

No Yandex, apenas a pesquisa pelo texto presente na página inicial resultou em sucesso. Coincidentemente essa é a única página que apareceu no teste de pesquisa de endereços do site. O fato dessa ser a página principal do site leva a crer que esse mecanismo possa ter problemas em detectar as outras páginas, o que indica um certo nível de falta de compatibilidade com a abordagem CSR, entretanto, é impossível fazer tal afirmação visto que pode apenas ser uma questão de tempo até que as outras páginas sejam indexadas propriamente.

A partir desses resultados é possível determinar que, ao menos, o motor de busca Google é compatível com a abordagem de renderização CSR. Entretanto, não foi possível determinar se os mecanismos Bing, Yandex, Yahoo! e DuckDuckGo também são.

## 7.2 RESULTADOS DO TESTE DE DESEMPENHO

Foram desenvolvidos dois sites praticamente iguais, um SSR e o outro CSR. Esses sites foram hospedados na plataforma vercel e foram feitas auditorias de performance nesses considerando três cenários:

- Um cenário pequeno, com 100 dados da rota */photos* da API JSONPlaceholder;
- Um cenário médio, com 1000 dados da rota */photos* da API JSONPlaceholder;
- Um cenário grande, com 5000 dados da rota */photos* da API JSONPlaceholder.

Para cada cenário existe um conjunto de duas tabelas de resultados. Uma com os resultados dos testes no Lighthouse e outra com o resultado dos testes no Chrome DevTools.

A ferramenta Lighthouse no modo *navigation* obtêm as métricas LCP, CLS, SI, FCP e TBT. No modo *timespan* obtêm as métricas INP, CLS e TBT. Foram feitos testes nos ambientes *desktop* e *mobile* dessa ferramenta. A ferramenta Chrome DevTools obtêm a métrica TTFB e foram feitos testes com as opções *fast 3G* e *slow 3G* dessa ferramenta. Cada teste feito com essas ferramentas foi repetido cinco vezes e o valor final considerado é a média dos valores obtidos.

É importante lembrar que cada métrica *Web Vitals* possui valores que são considerados bons ou ruins. Esses valores são levados em consideração nas discussões sobre os resultados obtidos. A tabela 7 mostra cada métrica e seus respectivos valores bons e ruins.

Tabela 7 – Valores bons e ruins para as métricas *Web Vitals*.

Métrica	Valor bom	Valor ruim
LCP	Menor ou igual a 2.5s	Maior do que 4s
INP	Menor ou igual a 200ms	Maior do que 500ms
CLS	Menor ou igual a 0.1	Maior do que 0.25
SI	Menor ou igual a 3.4s	Maior do que 5.8s
FCP	Menor ou igual a 1.8s	Maior do que 3s
TBT	Menor ou igual a 200ms	Maior do que 600ms
TTFB	Menor ou igual a 800ms	Maior do que 1800ms

Fonte: (Walton e Pollard, 2024; Wagner, 2024; Mihajlija e Walton, 2024; Lighthouse, 2019a; Walton, 2024; Lighthouse, 2019c; Wagner e Pollard, 2024).

Qualquer valor que esteja entre o valor bom e o valor ruim é considerado um valor mediano. Nas tabelas sobre os resultados é usado o seguinte esquema de cores:

- Caso o valor seja bom, é usado verde;
- Caso o valor seja mediano, é usado laranja;
- Caso o valor seja ruim, é usado vermelho.

Isso servirá para determinar em quais pontos uma abordagem é substancialmente melhor ou pior do que a outra. Por exemplo: caso a abordagem SSR tenha um valor verde em uma métrica e a abordagem CSR tenha um valor laranja nessa mesma métrica, isso significa que a abordagem SSR é consideravelmente melhor nesse quesito, sendo assim, isso é considerado. Caso as duas sejam verdes, por mais que uma seja melhor do que a outra, essa diferença não é tão significativa e, portanto, é desconsiderada.

### 7.2.1 Resultados no cenário pequeno

A tabela 8 exibe os resultados no cenário pequeno com a ferramenta Lighthouse nos modos *navigation* e *timespan* e nos ambientes *desktop* e *mobile*, em conjunto com o esquema de cores mencionado, em que verde significa um valor bom, laranja significa um valor mediano e vermelho significa um valor ruim.

Tabela 8 – Resultado das métricas do Lighthouse no cenário pequeno.

Métricas	SSR ( <i>Desktop</i> )	CSR ( <i>Desktop</i> )	SSR ( <i>Mobile</i> )	CSR ( <i>Mobile</i> )
LCP	0,72s	1,2s	1,94s	8,76s
INP	20ms	20ms	180ms	100ms
CLS ( <i>navigation</i> )	0	0.5368	0	1.2306
CLS ( <i>timespan</i> )	0.04	0.0058	0,009	0.052
SI	0,66s	1,2s	2,16s	2,54s
FCP	0,46s	1,2s	1,34s	2,54s
TBT ( <i>navigation</i> )	0ms	0ms	208ms	98ms
TBT ( <i>timespan</i> )	0ms	0ms	118ms	36ms

Fonte: (Próprio autor, 2024)

Comparações relevantes no ambiente *desktop*:

- O site CSR tem um valor de CLS (*navigation*) pior.

Comparações relevantes no ambiente *mobile*:

- O site SSR tem um valor de TBT (*navigation*) pior;
- O site CSR tem um valor de LCP, CLS (*navigation*) e FCP pior.

A tabela 9 exibe os resultados no cenário pequeno com a ferramenta Chrome DevTools nas velocidades de rede *fast 3G* e *slow 3G* em conjunto com o esquema de cores definido.

Tabela 9 – Resultado das métricas do Chrome DevTools no cenário pequeno.

Métricas	SSR	CSR
TTFB ( <i>fast 3G</i> )	584,182ms	589,58ms
TTFB ( <i>slow 3G</i> )	2,034s	2,038s

Fonte: (Próprio autor, 2024)

A abordagem de renderização não apresenta diferenças significativas de TTFB nos sites do cenário pequeno, tanto em um ambiente de rede mais rápido quanto em um mais lento.

## 7.2.2 Resultados no cenário médio

A tabela 10 exibe os resultados no cenário médio com a ferramenta Lighthouse nos modos *navigation* e *timespan* e nos ambientes *desktop* e *mobile*.

Tabela 10 – Resultado das métricas do Lighthouse no cenário médio.

Métricas	SSR ( <i>Desktop</i> )	CSR ( <i>Desktop</i> )	SSR ( <i>Mobile</i> )	CSR ( <i>Mobile</i> )
LCP	0,7s	1,1s	4s	5,76s
INP	20ms	20ms	182ms	152ms
CLS ( <i>navigation</i> )	0	0.4568	0	0.2042
CLS ( <i>timespan</i> )	0.0064	0.0032	0.008	0.008
SI	0,82s	0,92s	3s	3,06s
FCP	0,5s	0,92s	1,98s	3,06s
TBT ( <i>navigation</i> )	16ms	0ms	528ms	150ms
TBT ( <i>timespan</i> )	0ms	0ms	200ms	186ms

Fonte: (Próprio autor, 2024)

Comparações relevantes no ambiente *desktop*:

- O site CSR tem um valor de CLS (*navigation*) pior.

Comparações relevantes no ambiente *mobile*:

- O site SSR tem um valor de TBT (*navigation*) pior;
- O site CSR tem um valor de LCP, CLS (*navigation*) e FCP pior.

A tabela 11 exibe os resultados no cenário médio com a ferramenta Chrome DevTools nas velocidades de rede *fast 3G* e *slow 3G*.

Tabela 11 – Resultado das métricas do Chrome DevTools no cenário médio.

Métricas	SSR	CSR
TTFB ( <i>fast 3G</i> )	583,418	582,84
TTFB ( <i>slow 3G</i> )	2,028	2,03

Fonte: (Próprio autor, 2024)

A abordagem de renderização não apresenta diferenças significativas em TTFB nos sites do cenário médio, tanto em um ambiente de rede mais rápido quanto uma mais lento.

### 7.2.3 Resultados no cenário grande

A tabela 12 exibe os resultados no cenário grande com a ferramenta Lighthouse nos modos *navigation* e *timespan* e nos ambientes *desktop* e *mobile*.

Tabela 12 – Resultado das métricas do Lighthouse no cenário grande.

Métricas	SSR ( <i>Desktop</i> )	CSR ( <i>Desktop</i> )	SSR ( <i>Mobile</i> )	CSR ( <i>Mobile</i> )
LCP	0,84s	1,16s	5,92s	9,8s
INP	42ms	60ms	368ms	480ms
CLS ( <i>navigation</i> )	0	0.292	0	0.5886
CLS ( <i>timespan</i> )	0.0064	0	0.008	0.008
SI	1,88s	1,14s	7,26s	3,16s
FCP	0,8s	1,12s	4,54s	2,74s
TBT ( <i>navigation</i> )	324ms	486ms	2.412ms	2.410ms
TBT ( <i>timespan</i> )	4ms	0ms	1.484ms	1.230ms

Fonte: (Próprio autor, 2024)

Comparações relevantes no ambiente *desktop*:

- O site CSR tem um valor de CLS (*navigation*) pior.

Comparações relevantes no ambiente *mobile*:

- O site SSR tem um valor de SI, FCP e TBT (*navigation*) pior;
- O site CSR tem um valor de CLS (*navigation*) pior.

A tabela 13 exibe os resultados no cenário grande com a ferramenta Chrome DevTools nas velocidades de rede *fast 3G* e *slow 3G*.

Tabela 13 – Resultado das métricas do Chrome DevTools no cenário grande.

Métricas	SSR	CSR
TTFB ( <i>fast 3G</i> )	1,64s	590,594ms
TTFB ( <i>slow 3G</i> )	2,04s	2,026s

Fonte: (Próprio autor, 2024)

A abordagem de renderização SSR apresenta uma diferença significativa em TTFB nos sites do cenário grande, sendo pior do que a abordagem CSR.

#### 7.2.4 Resultados gerais

A tabela 14 exibe a média dos resultados obtidos nos três cenários com a ferramenta Lighthouse nos modos *navigation* e *timespan* e nos ambientes *desktop* e *mobile*.

Tabela 14 – Média dos resultados das métricas do Lighthouse nos três cenários.

Métricas	SSR ( <i>Desktop</i> )	CSR ( <i>Desktop</i> )	SSR ( <i>Mobile</i> )	CSR ( <i>Mobile</i> )
LCP	0,75s	1,15s	3,95s	8,11s
INP	20,00ms	20,00ms	181,33ms	134,67ms
CLS ( <i>navigation</i> )	0.00	0.43	0.00	4.37
CLS ( <i>timespan</i> )	0.02	0.00	0.01	0.02
SI	1,12s	1,09s	4,14s	2,92s
FCP	0,59s	1,08s	2,62s	2,78s
TBT ( <i>navigation</i> )	10,67ms	0,00ms	421,33ms	132,67ms
TBT ( <i>timespan</i> )	0,00ms	0,00ms	172,67ms	136,00ms

Fonte: (Próprio autor, 2024)

No geral, os pontos fracos da abordagem SSR se encontram no ambiente *mobile*, onde está possui resultados significativamente piores do que a abordagem CSR nas métricas SI e TBT (*navigation*), o que significa que com essa abordagem a exibição do conteúdo do site não é feita de forma muito gradual e que o site possui mais processos JS longos durante seu carregamento inicial. Já os pontos fracos da abordagem CSR consistem nas métricas CLS (*navigation*) e LCP. No caso, a métrica LCP é ruim apenas no ambiente *mobile*, e a métrica CLS é ruim em ambos os ambientes. Isso significa que com essa abordagem a estabilidade visual do site durante o carregamento inicial é ruim e que o site demora mais para renderizar seu maior elemento.

A tabela 15 exibe a média dos resultados nos três cenários com a ferramenta Chrome DevTools nas velocidades de rede *fast 3G* e *slow 3G*.

Tabela 15 – Média dos resultados das métricas do Chrome DevTools nos três cenários.

Métricas	SSR	CSR
TTFB ( <i>fast 3G</i> )	935,87ms	587,67ms
TTFB ( <i>slow 3G</i> )	2034,00ms	2031,33ms

Fonte: (Próprio autor, 2024)

No geral, a abordagem SSR possui um valor de TTFB pior do que a abordagem CSR no cenário *fast 3G*, o que significa que o servidor pode demorar mais para responder a requisição inicial do cliente. É importante perceber que essa diferença só se tornou significativa no cenário grande, em que o site possuía mais de 5000 dados dinâmicos sendo requisitados na página, o que mostra que o valor de TTFB da abordagem SSR só passa a ser consideravelmente pior caso o site tenha muitos dados dinâmicos.

## 8 CONSIDERAÇÕES FINAIS

Neste trabalho foram feitos testes de compatibilidade da abordagem CSR com motores de busca e testes de desempenho das abordagens SSR e CSR, a fim de determinar o quão impactante a abordagem escolhida é para SEO.

O teste de compatibilidade consistiu no desenvolvimento de um site CSR, na tentativa de indexação desse site nos mecanismos de busca Google, Bing, Yandex, Yahoo! e DuckDuckGo e na realização de buscas por esse site nesses mecanismos. Através desse teste foi possível determinar que o Google é compatível com a abordagem CSR, mas não foi possível determinar se os outros mecanismos também são. O Google é o mecanismo de busca mais usado, com uma fatia de 90,8% do mercado mundial de mecanismos de busca em maio de 2024 (StatCounter, 2024). Sendo assim, é possível afirmar que a abordagem CSR possui compatibilidade com a maior parte do mercado de mecanismos de busca mundial, dessa forma, o aspecto da compatibilidade só se torna um possível problema caso se deseje alcançar uma fatia de mercado que vá além dessa parte.

O teste de desempenho consistiu no desenvolvimento de dois sites praticamente idênticos, um SSR e outro CSR, e na realização de auditorias nesses sites com as ferramentas Lighthouse e Chrome DevTools para obter os valores de desempenho das métricas *Web Vitals*. Através desse teste foi possível determinar que a abordagem SSR é superior nas métricas LCP e CLS, enquanto a abordagem CSR é superior nas métricas SI, TBT e TTFB. Essas diferenças ocorrem majoritariamente no ambiente *mobile*, exceto CLS, no qual CSR é inferior em qualquer ambiente e TTFB, que só passa a ser um problema para SSR quando o site tem muitos dados dinâmicos.

Sendo assim, o aspecto da compatibilidade não é tão preocupante para a abordagem de renderização escolhida, visto que ao menos o principal mecanismo de busca, o Google, é compatível com a abordagem CSR, a menos que queira indexar o site em mecanismos que não sejam o Google. Nesses casos é possível que haja problemas relacionados a compatibilidade, mas é incerto. Já em relação ao desempenho, caso se desenvolva um site SSR, deve-se atentar a exibição gradual de elementos do site durante o carregamento da página, a duração de processos JS em ambientes *mobile* e ao tempo de resposta do servidor a primeira requisição do cliente quando o site tiver muitos dados dinâmicos. Já para sites CSR, deve-se atentar a estabilidade visual, e, em ambientes *mobile*, a velocidade de carregamento do maior elemento

do site. Esses são os principais aspectos relacionados a desempenho que são influenciados pela abordagem de renderização escolhida e que podem impactar SEO de forma negativa.

## REFERÊNCIAS

Atkins Jr., Tab; Sapin, Simon. **CSS Syntax Module Level 3**. W3C, dez. 2021. Disponível em: <https://www.w3.org/TR/css-syntax-3/>. Acesso em: 28 mai, 2024 às 12:01min.

Baidu Webmaster. 2024. Disponível em: <https://ziyuan.baidu.com/site/index>. Acesso em: 12 mai, 2024 às 19:20min.

Basques, Kayce; Emelianova, Sofia. **Referência de recursos de rede**. 2024. Disponível em: <https://developer.chrome.com/docs/devtools/network/reference?hl=pt-br#timing-explanation>. Acesso em: 12 mai, 2024 às 19:28min.

Beke, Mathias. **On the Comparison of Software Quality Attributes for Client-side and Server-side Rendering**. University of Antwerp, jun. 2018. Disponível em: <https://denbeke.be/thesis/versions/mathias-beke-final.pdf>. Acesso em: 08 mai, 2024 às 21:41min.

Bing Webmaster Tools. 2024. Disponível em: <https://www.bing.com/webmasters/about>. Acesso em: 12 mai, 2024 às 09:40min.

Cendón, Beatriz Valadares. **Ferramentas de busca na Web**. Ciência da informação: Brasília, vol. 30, n. 1, jun. 2001, p. 39-49. Disponível em: <https://revista.ibict.br/ciinf/article/view/937/974>. Acesso em: 08 mai, 2024 às 21:50min.

Chrome DevTools. 2024. Disponível em: <https://developer.chrome.com/docs/devtools?hl=pt-br>. Acesso em: 12 mai, 2024 às 10:01min.

Create React App. 2022. Disponível em: <https://create-react-app.dev/>. Acesso em: 12 mai, 2024 às 09:56min.

Dean, Brian. **We analyzed 4 million Google Search Results: Here's What We Learned About Organic Click Through Rate**. 2023. Disponível em: <https://backlinko.com/google-ctr-stats>. Acesso em: 08 mai, 2024 às 22:05min.

DuckDuckGo. **Where do DuckDuckGo search results come from?** 2024. Disponível em: <https://duckduckgo.com/duckduckgo-help-pages/results/sources/>. Acesso em: 12 mai, 2024 às 09:47min.

Fielding, Roy T.; Nottingham, Mark.; Reschke, Julian. **RFC 9110: HTTP Semantics**. Internet Engineering Task Force, jun. 2022. Disponível em: <https://www.rfc-editor.org/rfc/rfc9110>. Acesso em: 27 mai, 2024 às 21:53min.

Firdausi, Maulidina Marlita. **What Are Core Web Vitals and How to Measure Them**. 2024. Disponível em: <https://www.hostinger.com/tutorials/core-web-vitals>. Acesso em: 08 mai, 2024 às 22:31min.

Flanagan, David. **JavaScript: O Guia Definitivo**. Trad. João Eduardo Nóbrega Tortello. 6ª ed. Porto Alegre: Bookman Editora, 2012, p. 1-11. Disponível em: <https://books.google.com.br/books?id=zWNYDgAAQBAJ&printsec=frontcover>. Acesso em: 28 mai, 2024 às 11:34min.

Github. **Sobre o Git**. 2024. Disponível em: <https://docs.github.com/pt/get-started/using-git/about-git>. Acesso em: 12 mai, 2024 às 09:59min.

Google. **Como resultados são gerados automaticamente**. 2024. Disponível em: <https://www.google.com/search/howsearchworks/how-search-works/ranking-results/>. Acesso em: 08 mai, 2024 às 22:34min.

Google Search Console. 2024. Disponível em: <https://search.google.com/search-console/about>. Acesso em: 12 mai, 2024 às 09:37min.

Herbert, David. **What is React.js? Uses, Examples, & More**. 2023. Disponível em: <https://blog.hubspot.com/website/react-js>. Acesso em: 12 mai, 2024 às 09:54min.

Hors, Arnaud Le; Hégaret, Philippe Le; Nicol, Gavin; Wood, Lauren; Champion, Mike; Byrne, Steve. **Document Object Model Core**. W3C, abr. 2004. Disponível em: <https://www.w3.org/TR/DOM-Level-3-Core/core.html>. Acesso em: 28 mai, 2024 às 11:08min.

JSONPlaceholder. **{JSON} Placeholder**. 2024. Disponível em: <https://jsonplaceholder.typicode.com/>. Acesso em: 12 mai, 2024 às 19:14min.

Kenny, Brendan. **Novidades do Lighthouse 10**. 2023. Disponível em: <https://developer.chrome.com/blog/lighthouse-10-0/?hl=pt-br>. Acesso em: 08 mai, 2024 às 22:47min.

Kurose, James F.; Ross, Keith W. **Redes de computadores e a internet: Uma abordagem Top-Down**. Trad. Opportunity Translations. 5ª ed. São Paulo: Addison Wesley, 2010, p. 48-72. Disponível em: <https://www.facom.ufu.br/~sequincozes/referencias/kurose2010.pdf>. Acesso em: 10 mar, 2024 às 16:43min.

Ledford, Jerri L. **Search Engine Optimization Bible**. 5ª ed. John Wiley & Sons, 2008, p. 3-14. Disponível em: [https://books.google.com.br/books?id=sgmxo1Alq\\_4C&printsec=frontcover](https://books.google.com.br/books?id=sgmxo1Alq_4C&printsec=frontcover). Acesso em: 08 mai, 2024 às 21:57min.

Leiner, Barry M.; Cerf, Vinton G.; Clark, David D.; Kahn, Robert E.; Kleinrock, Leonard; Lynch, Daniel C.; Postel, Jon; Roberts, Lawrence G.; Wolff, Stephen S. **The Past and Future History of the Internet**. Communications of the ACM, vol. 40, n. 2, p. 102-108, Fev. 1997. Disponível em: <https://dl.acm.org/doi/pdf/10.1145/253671.253741>. Acesso em: 08 mar, 2024 às 10:50min.

Lighthouse. **Índice de velocidade**. 2019a. Disponível em: <https://developer.chrome.com/docs/lighthouse/performance/speed-index?hl=pt-br>. Acesso em: 08 mai, 2024 às 22:43min.

Lighthouse. **Tempo para interação da página.** 2019b. Disponível em: <https://developer.chrome.com/docs/lighthouse/performance/interactive?hl=pt-br>. Acesso em: 08 mai, 2024 às 22:46min.

Lighthouse. **Tempo total de bloqueio.** 2019c. Disponível em: <https://developer.chrome.com/docs/lighthouse/performance/lighthouse-total-blocking-time?hl=pt-br#how-lighthouse-determines-your-tbt-score>. Acesso em: 12 mai, 2024 às 19:17min.

Lighthouse. **User Flows in Lighthouse.** 2023. Disponível em: <https://github.com/GoogleChrome/lighthouse/blob/main/docs/user-flows.md>. Acesso em: 12 mai, 2024 às 19:22min.

Lighthouse. 2024. Disponível em: <https://chromewebstore.google.com/detail/lighthouse/blipmdconlkpinefehnmmjammfjpmbjk?hl=pt-BR>. Acesso em: 27 mai, 2024 às 22:23min.

Lins, Bernardo Felipe Estellita. **A evolução da internet:** uma perspectiva histórica. Caderno Aslegis, n. 48, p. 11-45, jan.-abr. 2013. Disponível em: [https://www.belins.eng.br/ac01/papers/aslegis48\\_art01\\_hist\\_internet.pdf](https://www.belins.eng.br/ac01/papers/aslegis48_art01_hist_internet.pdf). Acesso em: 09 mar, 2024 às 11:05min.

MDN Web Docs. **Introduction to client-side frameworks.** 2024a. Disponível em: [https://developer.mozilla.org/en-US/docs/Learn/Tools\\_and\\_testing/Client-side\\_JavaScript\\_frameworks/Introduction](https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Introduction). Acesso em: 09 abr, 2024 às 22:10min.

MDN Web Docs. **Critical Rendering Path.** 2024b. Disponível em: [https://developer.mozilla.org/en-US/docs/Web/Performance/Critical\\_rendering\\_path](https://developer.mozilla.org/en-US/docs/Web/Performance/Critical_rendering_path). Acesso em: 10 mar, 2024 às 18:09min.

Mihajlija, Milica; Walton, Philip. **Cumulative Layout Shift (CLS).** 2024. Disponível em: <https://web.dev/articles/cls?hl=pt-br>. Acesso em: 08 mai, 2024 às 22:41min.

Next.js. **Client Components.** 2024a. Disponível em: <https://nextjs.org/docs/app/building-your-application/rendering/client-components>. Acesso em: 26 mai, 2024 às 19:09min.

Next.js. **Introduction.** 2024b. Disponível em: <https://nextjs.org/docs>. Acesso em: 12 mai, 2024 às 09:57min.

Next.js. **Server Components.** 2024c. Disponível em: <https://nextjs.org/docs/app/building-your-application/rendering/server-components>. Acesso em: 26 mai, 2024 às 19:08min.

Next.js. **App Router.** 2024d. Disponível em: <https://nextjs.org/docs/app>. Acesso em: 26 mai, 2024 às 19:08min.

Ollila, Risto; Mäkitalo, Niko; Mikkonen, Tommi. **Modern Web Frameworks: A Comparison of Rendering Performance.** Journal of Web Engineering, vol. 21, n. 3, p. 789-814, mar. 2022. Disponível em: <https://journals.riverpublishers.com/index.php/JWE/article/view/7217/11489>. Acesso em: 08 mai, 2024 às 21:29min.

Roland, Chris Ebube. **Static Site Generation (SSG) vs Server-Side Rendering in Next.js**. 2023. Disponível em: <https://medium.com/@chrisebuberoiland/static-site-generation-ssg-vs-server-side-rendering-in-next-js-debf43f4bb7f>. Acesso em: 11 mai, 2024 às 16:27min.

Seymour, Joseph Tom; Frantsvog, Dean; Kumar, Satheesh. **History Of Search Engines**. International journal of management & information systems, vol. 15, n. 4, set. 2011. Disponível em: [https://www.researchgate.net/publication/265104813\\_History\\_Of\\_Search\\_Engines](https://www.researchgate.net/publication/265104813_History_Of_Search_Engines). Acesso em: 12 mai, 2024 às 10:11min.

Sommerville, Ian. **Engenharia de Software**. Trad. Ivan Bosnic e Kalinka G. de O. Gonçalves. 9ª ed. São Paulo: Pearson Prentice Hall, 2011, p. 57-62. Disponível em: <https://www.facom.ufu.br/~william/Disciplinas%202018-2/BSI-GSI030-EngenhariaSoftware/Livro/engenhariaSoftwareSommerville.pdf>. Acesso em: 26 mai, 2024 às 19:36min.

StatCounter. **Search Engine Market Share Worldwide**. 2024. Disponível em: <https://gs.statcounter.com/search-engine-market-share>. Acesso em: 02 jun, 2024 às 19:12min.

Sydorkina, Anastasiia. **Key Differences Between Client-Side, Server-Side and Pre-rendering**. 2023. Disponível em: <https://clockwise.software/blog/client-side-vs-server-side-vs-pre-rendering/>. Acesso em: 08 mai, 2024 às 22:07min.

Tailwind CSS. 2024. Disponível em: <https://tailwindcss.com/>. Acesso em: 12 mai, 2024 às 09:51min.

Unsplash. 2024. Disponível em: <https://unsplash.com/pt-br>. Acesso em: 12 mai, 2024 às 19:26min.

Vega, Cristian. **Client-side vs. server-side rendering: why it's not all black and white**. 2017. Disponível em: <https://www.freecodecamp.org/news/what-exactly-is-client-side-rendering-and-hows-it-different-from-server-side-rendering-bd5c786b340d/>. Acesso em: 08 mai, 2024 às 22:26min.

Vercel. **Get started with Vercel**. 2024. Disponível em: <https://vercel.com/docs/getting-started-with-vercel>. Acesso em: 12 mai, 2024 às 09:58min.

Visual Studio Code. 2024 Disponível em: <https://visualstudio.microsoft.com/pt-br/#vscode-section>. Acesso em: 12 mai, 2024 às 09:50min.

Vite. 2024. Disponível em: <https://pt.vitejs.dev/guide/>. Acesso em: 12 mai, 2024 às 09:55min.

Wagner, Jeremy; Pollard, Barry. **Tempo até o primeiro byte (TTFB)**. 2021. Disponível em: <https://web.dev/articles/ttfb?hl=pt-br>. Acesso em: 08 mai, 2024 às 22:49min.

Wagner, Jeremy. **Interação com Próxima Exibição (INP)**. 2024. Disponível em: <https://web.dev/articles/inp?hl=pt-br>. Acesso em: 08 mai, 2024 às 22:40min.

Walton, Philip. **Métricas de performance centradas no usuário**. 2019. Disponível em: <https://web.dev/articles/user-centric-performance-metrics?hl=pt-br>. Acesso em: 08 mai, 2024 às 22:37min.

Walton, Philip. **Web Vitals**. 2023a. Disponível em: <https://web.dev/articles/vitals?hl=pt-br>. Acesso em: 08 mai, 2024 às 22:29min.

Walton, Philip. **First Input Delay (FID)**. 2023b. Disponível em: <https://web.dev/articles/fid?hl=pt-br>. Acesso em: 08 mai, 2024 às 22:42min.

Walton, Philip. **Tempo para interação da página (TTI)**. 2023c. Disponível em: <https://web.dev/articles/tti?hl=pt-br>. Acesso em: 08 mai, 2024 às 22:45min.

Walton, Philip. **Tempo total de bloqueio (TBT)**. 2023d. Disponível em: <https://web.dev/articles/tbt?hl=pt-br>. Acesso em: 08 mai, 2024 às 22:48min.

Walton, Philip. **First Contentful Paint (FCP)**. 2024. Disponível em: <https://web.dev/articles/fcp?hl=pt-br>. Acesso em: 08 mai, 2024 às 22:44min.

Walton, Philip; Pollard, Barry. **Largest Contentful Paint (LCP)**. 2024. Disponível em: <https://web.dev/articles/lcp?hl=pt-br>. Acesso em: 08 mai, 2024 às 22:38min.

WHATWG. **HTML: Living Standard**. The Web Hypertext Application Technology Working Group, mai. 2024. Disponível em: <https://html.spec.whatwg.org/>. Acesso em: 28 mai, 2024 às 12:01min.

Yahoo Search. **Submit your website to Yahoo Search**. 2024. Disponível em: <https://help.yahoo.com/kb/SLN2217.html>. Acesso em: 12 mai, 2024 às 09:43min.

Yandex Webmaster. 2024. Disponível em: <https://webmaster.yandex.com/welcome/>. Acesso em: 12 mai, 2024 às 09:41min.

Yeager, Nancy J.; McGrath, Robert E. **Web Server Technology**. Morgan Kaufmann, 1996, p. 19-20. Disponível em: [https://books.google.com.br/books?id=0jExRH3\\_hQC&printsec=frontcover](https://books.google.com.br/books?id=0jExRH3_hQC&printsec=frontcover). Acesso em: 08 mai, 2024 às 21:36min.

**APÊNDICE A – REPOSITÓRIO PARA OS CÓDIGOS DOS SITES DESENVOLVIDOS  
E TODOS OS RELATÓRIOS GERADOS NO TESTE DE DESEMPENHO**

<https://github.com/leobez/seo-comparison>