SSR X CSR – Duas formas de renderização de aplicativos.

Resumo

O objetivo desse paper é determinar qual a melhor forma de renderizar o aplicativo utilizando React e TypeScript, para isso, no primeiro cenário, criei uma aplicação sem a dependência do Express (que é o Framework que irei utilizar para a aplicação SSR) e utilizei o comando npm run build para criar um projeto que roda diretamente em produção, mas que não utiliza renderização ao lado do servidor. Já no outro projeto, incluirei o arquivo server.ts e irei utilizar as ferramentas de renderização ao lado do servidor do Express.js.

O que é CSR?

CSR é uma técnica de desenvolvimento de páginas Web que significa Client-Side Rendering, ou renderização ao lado do cliente. Diferentemente de muitas outras aplicações, no CSR ao invés de você gerar .dlls diretamente no seu servidor e fornecer acesso a elas, você fornece um arquivo .html e .js sendo assim compatível com praticamente todos os servidores.

Vantagens:

1. Implementações menores: Ao rodar um comando como npm run build a tendência é que a implementação da sua aplicação contenha menos memória do que a implementação gerada por um dotnet publish, que gera uma aplicação para ser renderizada ao lado do servidor.
2. O conteúdo do seu site é baixado apenas uma única vez: O conteúdo do seu site não é baixado diversas vezes como quando o seu site é desenvolvido utilizando SSR. A partir disso, você pode baixar apenas o que for necessário, todas as vezes em que estiver entrando em seu site. Num app de redes sociais ou de videoconferências, é inviável que funções inteiras de JavaScript sejam baixadas toda a hora. Por isso, utilizamos uma abordagem que baixa TODO o conteúdo do site, principalmente o conteúdo que ainda não havia sido roteado. Isso torna a navegação muito mais rápida e simples e a interação com o site muito mais amigável.
3. Amigável ao servidor: Ao rodar um comando como npm run build, você gera um arquivo .html, um arquivo .css e um arquivo .js com todo o conteúdo do seu site, sites .html são mais amigáveis para servidores do que arquivos .dll, que tendem a necessitar de dependências externas como o Microsoft Web Deploy para rodarem em produção. Para um servidor pequeno que quiser rodar um site estático, um arquivo .html é extremamente amigável.
4. Menos processamento: Com uma abordagem totalmente SSR, o processamento é totalmente feito ao lado do servidor e com CSR o processamento passa a ser ao lado do cliente, o que permite um melhor desempenho para a aplicação de forma geral e menos sobrecarga no servidor.
5. Maior tempo de Resposta: A partir da abordagem CSR, ao invés da página inteira ter que ser carregada diretamente, pequenos componentes individuais podem ser carregados sob demanda, o que afeta diretamente a experiência de usuário.

Desvantagens:

1. Baixa pontuação de SEO: Seu site provém apenas conteúdo .js e certos módulos de SEO não são amigáveis para arquivos .js, portanto, para muitos desses módulos, você enviará para eles ao invés de um arquivo .html com o conteúdo da página, um arquivo .html totalmente vazio. O que tende a prejudicar o SEO.
2. Latência: Utilizando SSR, você terá que transferir todo o código HTML e JavaScript para o navegador do usuário, além disso, em muitos casos os usuários passam pouquíssimo tempo dentro da nossa aplicação, por nesses casos, utilizar SSR é uma abordagem EXTREMAMENTE conveniente. Uma técnica que tende a melhorar o desempenho é a técnica de code-splitting, que é o envio para o cliente de pequenos pedaços de JavaScript.
3. Muita dependência do cliente: Isso significa que tudo dentro da sua aplicação é feito ao lado do cliente, principalmente o processamento e roteamento da aplicação, isso pode impactar na experiência de usuário a depender da qualidade do equipamento dele. Certamente, uma máquina melhor irá proporcionar melhores experiências de usuário para os clientes.

SSR – Renderização ao Lado do Servidor

Nesse modelo, ao invés do navegador do usuário baixar o código HTML e JavaScript o servidor por meio de um programa gera código HTML estático e manda para o usuário. Isso é feito especialmente quando o usuário acessa cada uma das rotas do sistema.

Vantagens

1. Melhorias de SEO: O conteúdo da página é provido por meio do HTML, o que melhora o SEO. Visto que em uma abordagem CSR o conteúdo é disponibilizado por meio de funções de JavaScript.
2. Menor tempo de carregamento: Diferentemente do CSR, que baixa arquivos .js e .html que por vezes podem ser pesados e que o navegador ainda tem que processar esses arquivos. Ao utilizar SSR, o servidor já retorna o HTML estático necessário. Isso melhora o tempo de carregamento e é voltado para páginas onde o usuário irá interagir pouco e de maneira rápida. Além disso, quando uma página feita com SSR é carregada, é carregada a página inteira e não pequenos pedaços individuais.
3. Criação de um sistema mais democrático: Seu sistema tende a ser muito mais democrático, pois o processamento que antes ficaria ao lado do cliente passa a ser 100% coberto ao lado do servidor, o que torna tudo mais democrático, visto que o trabalho que o cliente terá é de receber uma página HTML já pronta e disponibilizar diretamente no navegador.
4. Segurança: Utilizando SSR, você pode manter dados sensíveis como dados de cliente, API Keys e outros dados relevantes sem se preocupar com a utilização indevida dos seus dados pelo cliente. Dados como chaves de API por exemplo, ficam diretamente no seu servidor e não são repassados ao cliente.

Desvantagens

1. Maior processamento: Utilizando SSR, o processamento é inteiramente feito ao lado do servidor, a depender do seu servidor, pode não valer a pena a utilização de SSR justamente porque ele terá que enviar as páginas HTML e fazer o processamento completo antes de enviar para o cliente.

2. Maior tempo até a primeira resposta: Para que o usuário tenha acesso ao primeiro componente, é necessário que ele espere a página inteira ser carregada, o que tende a prejudicar um pouco a experiência de usuário.

Meu estudo e metodologia

Para esse estudo, criarei uma aplicação dentro da Academy SM e colocarei ela diretamente ao servidor na seguinte URL: https://academysm.com/server, a fim de verificar os resultados dessa implementação.

Para o estudo, utilizarei as seguintes métricas de desempenho:

1. Tempo de Carregamento das páginas;
2. Armazenamento no Servidor da aplicação
3. Interatividade da Página: Interatividade é um termo importante e relativo, a metodologia que utilizaremos para calcular a interatividade na página se baseia em diferentes níveis de interatividade e em diferentes quantidades de tempo para essa interatividade.
    a. Nível 1 – Páginas Estáticas sem Formulários como Portfólios;
    b. Nível 2 – Páginas Estáticas com Baixa Interatividade como formulários de contrato, cadastro ou outro tipo de páginas desse tipo.
    c. Nível 3 – Páginas com Conteúdos atualizados dinamicamente após a Renderização Inicial, tais como uma página de produtos ou carrinho;
    d. Nível 4 – Páginas que Exigem APIs em Tempo Real, tais como Sistemas de Chat em tempo real;
    e. Nível 5 – Aplicações Extremamente Interativas, tais como Videoconferências.
4. SEO: Verificar a pontuação da minha página em mecanismos de SEO;
5. Latência;

A partir daqui nós iremos começar de fato o nosso estudo.

Aplicação SSR acessável em https://academysm.com/ssr conseguiu as seguintes pontuações em computadores:

Primeira exibição de conteúdo: 0,9 s – Bom.

Tempo total de Bloqueio – 220ms – Moderado

Índice de Velocidade – 1,3s – Bom

Maior Exibição de Conteúdo – 1,1s – Bom

Mudanças cumulativas no Layout – 0,005s – Bom

Desempenho Geral - 89

SEO – 75

Aplicação SSR em dispositivos móveis:

Primeira exibição de conteúdo: 2,1 s – Moderado.

Tempo total de Bloqueio – 2420ms – Ruim

Índice de Velocidade – 4,3s – Bom

Maior Exibição de Conteúdo – 3,7s – Bom

Mudanças cumulativas no Layout – 0,001s – Bom

Desempenho Geral - 56

SEO – 82

Aplicação CSR acessível pelo link: https://academysm.com se comporta da seguinte forma em computadores:

Primeira exibição de conteúdo: 0,5 s – Bom.

Tempo total de Bloqueio – 440ms – Ruim

Índice de Velocidade – 1,1s – Bom

Maior Exibição de Conteúdo – 1,2s – Bom

Mudanças cumulativas no Layout – 0.003 – Bom

Desempenho Geral - 78

SEO – 75

Aplicação CSR acessível pelo seguinte link: https://academysm.com/ssr se comporta da seguinte forma em dispositivos móveis:

Primeira exibição de conteúdo: 2,1 s – Bom.

Tempo total de Bloqueio – 2580ms – Ruim

Índice de Velocidade – 2,9s – Bom

Maior Exibição de Conteúdo – 3s – Bom

Mudanças cumulativas no Layout – 0.017 – Bom

Desempenho Geral - 63

SEO – 82

Comparativo geral em computadores:

Quesito	**Favorece**	**SSR**	**CSR**	**Vencedor**
Primeira exibição de conteúdo	CSR	0,9s	0,5s	CSR
Tempo Total de Bloqueio	SSR	220ms	440ms	SSR
Índice de Velocidade	CSR	1,3s	1,1s	CSR
Maior Exibição de Conteúdo	SSR	1,1s	1,2s	SSR
Mudanças Cumulativas no Layout	SSR	0.005	0.003	CSR
Desempenho Geral	N/A	89	78	SSR
SEO	SSR	75	75	Empate

As tecnologias CSR conseguiram um significativo empate em um comparativo favorável ao lado do servidor e por ser uma página pequena e sem layout, o CSR conseguiu uma vitória num quesito extremamente favorável ao SSR.

O comparativo para os dispositivos móveis ficou da seguinte forma:

Quesito	Favorece	SSR	CSR	Vencedor
Primeira exibição de conteúdo	CSR	2,1s	2,1s	Empate
Tempo Total de Bloqueio	SSR	2420ms	2580ms	SSR
Índice de Velocidade	CSR	4,3s	2,9s	CSR
Maior Exibição de Conteúdo	SSR	3,7s	3s	CSR
Mudanças Cumulativas no Layout	SSR	0.001	0.017	SSR
Desempenho Geral	N/A	56	63	CSR
SEO	SSR	82	82	Empate

Temos diferenças significativas em métricas como o índice de velocidade, a maior exibição de conteúdo e as mudanças cumulativas no layout entre as duas tecnologias, mas nos dispositivos móveis, também podemos decretar o CSR como vencedor, especialmente por vencer numa métrica extremamente favorável às tecnologias CSR.

Resultado do Estudo

Antes de tudo, é importante citar que nenhuma das tecnologias foi utilizada ao máximo, apesar do uso de tecnologias inovadoras, tais como o Vite para o CSR e o Express para o SSR. Mas inicialmente, as otimizações devem ser feitas a nível de código e devem ser feitas focando em extrair o máximo que o CSR é capaz de extrair. Além disso, o indicador que me levou a realizar o estudo que foi justamente o SEO registrou um empate nos dois tipos de dispositivos. Portanto, decretamos que por ora, a utilização do Vite sozinho é o ideal para o desenvolvimento da nossa aplicação. No caso de diferentes atualizações desse paper, nós iremos salvá-lo diretamente neste repositório.

Bibliografia:

Insights – Academy SM SSR - PageSpeed Insights

Insights – Academy SM CSR - PageSpeed Insights

Índice de Velocidade – Explicação - Índice de velocidade | Lighthouse | Chrome for Developers

Maior Exibição de Conteúdo – Explicação - Maior exibição de conteúdo | Lighthouse | Chrome for Developers

Mudanças Significativas no Layout - Cumulative Layout Shift (CLS) | Articles | web.dev

Primeira Exibição de Conteúdo - Primeira exibição de conteúdo | Lighthouse | Chrome for Developers

Tempo Total de Bloqueio - Tempo total de bloqueio | Lighthouse | Chrome for Developers

Fundamentos da Pesquisa Google - Fundamentos da Pesquisa Google (antigas Diretrizes para webmasters) | Central da Pesquisa Google | Documentação | Google for Developers