Atividades Propostas

Teóricas:

1. Explique as diferenças entre Server-Side Rendering (SSR) e Client-Side Rendering (CSR).

Server-Side Rendering (SSR) e Client-Side Rendering (CSR) são duas abordagens diferentes para renderizar páginas web. No SSR, o HTML completo da página é gerado no servidor a cada requisição e enviado ao navegador do usuário. Assim, o conteúdo é exibido quase imediatamente após o carregamento inicial, mesmo antes do JavaScript ser executado. Já no CSR, a maior parte do trabalho é feita no navegador: o servidor entrega uma estrutura HTML básica, e então o JavaScript baixa e monta dinamicamente o conteúdo da página no cliente. Isso resulta em uma carga inicial mais rápida no nível do código, mas o conteúdo visível pode demorar mais a aparecer, especialmente em conexões lentas.

2. Liste três vantagens e três desvantagens do SSR.

O SSR traz diversas vantagens importantes. Primeiro, ele melhora o desempenho percebido, pois o conteúdo aparece mais rapidamente para o usuário, especialmente em conexões lentas ou dispositivos menos potentes. Segundo, ele favorece o SEO, já que os bots de mecanismos de busca recebem uma página já renderizada e podem indexá-la corretamente. Terceiro, ele oferece uma melhor experiência para usuários que desativaram o JavaScript ou têm restrições técnicas. Por outro lado, o SSR tem desvantagens. Ele pode aumentar a carga do servidor, pois cada requisição exige renderização dinâmica. Também pode introduzir tempos de resposta mais longos se o servidor estiver sobrecarregado. E por fim, ele limita certas interações em tempo real, como animações e navegação fluida, exigindo mais esforço para integrar essas funcionalidades após o carregamento inicial.

3. Qual é o impacto do SSR no SEO? Justifique com exemplos.

O SSR tem um impacto altamente positivo no SEO. Isso acontece porque os motores de busca como o Google ou o Bing conseguem indexar com mais facilidade páginas que já vêm renderizadas do servidor, contendo todo o conteúdo visível logo na resposta HTTP inicial. Por exemplo, em um site de e-commerce que utiliza SSR, a página de um produto já chega ao navegador (e ao crawler de busca) com o nome, a descrição e o preço do item em HTML. Isso garante que esses dados sejam corretamente indexados. Já em CSR, se o conteúdo depende da

execução de JavaScript para aparecer, há o risco de o bot não executá-lo corretamente, o que pode afetar a visibilidade da página nos resultados de busca.

4. Explique a diferença entre hidratação e renderização inicial no SSR.

No contexto de SSR, a renderização inicial acontece no servidor, que monta o HTML completo da página e o envia para o navegador. Isso permite que o usuário veja o conteúdo assim que ele chega, mesmo antes do JavaScript ser executado. Já a hidratação é o processo que ocorre no lado do cliente logo após o HTML ser carregado: o JavaScript entra em ação para "ligar" os componentes interativos, como botões, formulários e animações, transformando o HTML estático em uma aplicação funcional e dinâmica. Ou seja, a renderização inicial mostra o conteúdo, e a hidratação dá vida a ele.

5. Pesquise e explique como o Next.js utiliza SSR para melhorar a experiência do desenvolvedor.

O Next.js é um framework React que oferece suporte nativo ao Server-Side Rendering, entre outras técnicas modernas de renderização. Ele facilita muito a implementação de SSR ao permitir que o desenvolvedor apenas exporte uma função chamada getServerSideProps em suas páginas. Essa função é executada no servidor a cada requisição, gerando conteúdo dinâmico com facilidade e sem a necessidade de configurar um servidor do zero. Além disso, o Next.js oferece uma abordagem híbrida, permitindo que cada página use SSR, Static Site Generation (SSG) ou Client-Side Rendering (CSR), conforme a necessidade. Isso melhora a experiência do desenvolvedor ao dar flexibilidade, desempenho e boas práticas de SEO sem exigir configuração manual complexa. A integração com APIs, a otimização automática de imagens e a divisão de código também são recursos que tornam o SSR mais simples e eficiente dentro do Next.js.

Obs.: As atividades teóricas se encontram realizadas nos arquivos do projeto na pasta Módulo 5 - Templating e Interface com Servidor