Teste

Desenvolvimento de aplicações Front-end

hypesoft.com.br

AV. MARQUES DE SÃO VICENTE, Nº 2219 | 1-139
BARRA FUNDA | SÃO PAULO - SP | CEP 05036-040

hypesoft

Objetivo

Desenvolver uma aplicação web utilizando Next.js que consiste em uma página inicial e um formulário de contato. O teste deve demonstrar o conhecimento em boas práticas de desenvolvimento front-end, manipulação de formulários, integração com APIs, e validação de dados.

Requisitos Funcionais

Seção de Apresentação:

- Exibir uma mensagem de boas-vindas que inclui o nome do desenvolvedor e uma breve descrição sobre sua especialidade (ex.: Software Engineer, UI/UX Designer, etc.).
- Incluir uma imagem ou avatar representativo, que pode ser obtido dinamicamente de uma API ou carregado localmente.

Seção de Serviços:

- Apresentar os principais serviços oferecidos, como "UI/UX Design", "Product Design", "Branding Design", e "Front-End Development", com ícones e breves descrições.

Seção de Trabalhos Selecionados:

- Exibir três ou mais exemplos de projetos realizados, com imagens ou screenshots dos projetos.
- Cada projeto deve ter um link para uma descrição detalhada ou uma página dedicada, embora essas páginas não precisem ser implementadas no teste.

Rodapé:

- Incluir links para redes sociais e informações de contato básico.

Formulário de Contato:

- **Campos do Formulário:**

- Nome (campo obrigatório, texto simples)
- E-mail (campo obrigatório, com validação de formato de e-mail)
- Mensagem (campo obrigatório, texto de até 500 caracteres)

Validações:

- Implementar validação de dados utilizando uma biblioteca como Yup em conjunto com Formik ou React Hook Form.
- Todos os campos devem ser validados no front-end antes do envio. O formulário não deve permitir o envio se houver erros de validação.

Envio

- O formulário deve ser enviado para uma API (um endpoint fictício pode ser usado para simulação), e o usuário deve receber feedback visual indicando sucesso ou falha no envio.

Acessibilidade

- Certificar-se de que todos os elementos do formulário são acessíveis via teclado e que possuem descrições adequadas para tecnologias assistivas.

Tratamento de Erros

- Implementar mensagens de erro claras para cada campo, exibindo o erro abaixo do respectivo campo.

Requisitos Técnicos

Boas Práticas de Desenvolvimento:

- Utilizar TypeScript em toda a aplicação.
- Estruturar o código seguindo princípios de SOLID e Clean Architecture.
- Componentização e reutilização de código quando aplicável.
- Usar CSS Modules, styled-components, ou Tailwind CSS para estilos, garantindo encapsulamento e evitando conflitos de escopo.
- Implementar Lazy Loading para componentes não essenciais e imagens, otimizando o desempenho da página.
- Utilizar Next.js Image Optimization para carregar imagens de forma otimizada.

- Configurar a aplicação para ser Server-Side Rendered (SSR) e Static Site Generated (SSG) onde apropriado.
- Incluir metadados adequados e tags de Open Graph para melhorar SEO e compartilhamento em redes sociais.
- Implementar responsividade para garantir que a aplicação seja perfeitamente utilizável em dispositivos móveis, tablets e desktops.

Testes:

- Escrever testes unitários para os componentes principais utilizando Jest e React Testing Library.
- Incluir testes de integração para o formulário, garantindo que as validações funcionam como esperado e que o feedback ao usuário está correto.
- Testar a aplicação em múltiplos navegadores para garantir compatibilidade cross-browser.

Containerização com Docker:

- Criar um Dockerfile para a aplicação, permitindo que ela seja executada em um contêiner Docker.
- Incluir um arquivo docker-compose.yml se necessário, para orquestrar a aplicação e seus serviços associados (ex.: API fictícia).
- Documentar o processo de build e execução da aplicação utilizando Docker no README.md.

Controle de Versão e Documentação:

- Utilizar Github para controle de versão e criar commits significativos e descritivos.
- Commits Convencionais: Seguir o padrão de Conventional Commits, onde cada commit deve indicar o tipo de mudança (feat, fix, chore, etc.), um escopo opcional, e uma descrição clara. Exemplo: feat(home): add responsive layout for mobile devices.
- Criar um README.md detalhado explicando como configurar o ambiente, rodar os testes, utilizar Docker, e implementar a aplicação localmente.
- Disponibilizar o repositório do Github para avaliação.

Critérios de Avaliação:

- Qualidade do código, incluindo clareza, organização, e uso de boas práticas.
- Implementação de funcionalidades conforme especificado.
- Qualidade da interface de usuário e experiência do usuário.

- Implementação e cobertura de testes automatizados.
- Desempenho e otimização da aplicação.
- Conformidade com os padrões de acessibilidade.
- Implementação de responsividade para diversos dispositivos.
- Uso correto e eficiente do Docker para containerização da aplicação.
- Aderência ao padrão de Conventional Commits no controle de versão.
- Essa é uma oportunidade para demonstrar suas habilidades técnicas, capacidade de seguir boas práticas de desenvolvimento e sua competência em criar uma aplicação web moderna e bem estruturada.

Protótipo

https://www.figma.com/design/wk1JMamLja1wvxKuc32thb/Hypesoft---Desafio?node-id=0-1&t=mg0sWGphnLDCyf7Z-0