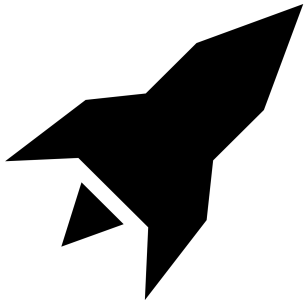

Review desafio frontend

Candidato: Florencia Sol
Github: https://github.com/florenciafer/Nave.rs
Revisores: Vítor Tavares

Repositório e estrutura do projeto

Prós

- Boa organização e separação de pastas.
- Boa separação de lógica e estilos.

Contras

- Muitos console.logs e comentários deixados no código. Na grande maioria dos casos, comentários não devem ser usados como documentação, é melhor documentar pontos necessários ou interessantes no README.md do projeto, por exemplo.
- Alguns arquivos padrão do Create-react-app deixados no projeto sem necessidade.

Layout e UX

Prós

- Mantém login ao recarregar a página.
- Design próximo ao do Figma provido.
- Controlou a responsividade básica do projeto.
- Features CRUD totalmente funcionais.

Contras

- Permite realizar request de login para a API mesmo com os campos do form vazios. Ambos os inputs do form da página de login deveriam ter o atributo “required” para torná-los obrigatórios e assim evitar chamadas desnecessárias para a API.

- É boa prática mudar o cursor para “pointer” quando ocorre o hover de elementos clicáveis (como por exemplo botões, ícones clicáveis e links).
- No modal de detalhes alguns dos campos estão com uma fonte não estilizada (Times New Roman padrão do navegador). Por o projeto utilizar apenas uma fonte, ao invés de repetir a propriedade font- family dentro de vários elementos poderia ter aplicado “font-family: Montserrat;” dentro do seletor universal (*) dentro dos estilos globais.
- O “X” para fechar o modal de detalhes está mudando de lugar dependendo do tamanho da imagem. A melhor solução seria fixar o tamanho do modal e fazer a imagem se adaptar ao espaço pré-determinado utilizando a propriedade object-fit do CSS, assim o botão ficaria consistentemente no mesmo lugar e seria só posicioná-lo no canto superior direito.
- Poderia ter adicionado um loader para dar feedback ao usuário de que uma ação está ocorrendo.
- Na página de Adição ou Edição de Navers o layout está bastante diferente do Figma, além disso no caso de Edição os campos do formulário deveriam vir preenchidos com as informações atuais do Naver que está sendo editado.
- Após editar ou adicionar um Naver deveria ser feito um redirecionamento de volta para a página inicial.
- Alguns elementos da interface estão em espanhol, não seria um problema fazer tudo em espanhol, português ou inglês, só é interessante manter uma consistência para a comunicação com o usuário final ser sempre a mais clara e homogênea possível.
- Na listagem de Navers imagens retangulares “quebram” o layout de grade. A solução é similar à já citada do modal, fixar o tamanho da imagem e utilizar object-fit para evitar distorções da imagem.

Código

Prós

- Bom uso de features do ES6+ (template literals, async await, arrow functions).
- Bom uso de React Hooks básicos (useState, useEffect).
- Boa abstração para uso de localStorage através de custom hook.
- Bom uso de React router.

Contras

- Cuidar formatação (principalmente indentação) do código para facilitar a leitura e comprensão. Recomendo utilizar ESLint e Prettier para fazer automaticamente essa checagem e adequação às boas práticas e formatação de código.
- Procurar estabecer um padrão entre async await e then e utilizar apenas um deles, na maior parte dos casos prezando pelos padrões mais novos do ES6+ (async await). O mesmo vale para function e arrow function (=>), dando preferência para arrow functions.
- Ao invés de construir uma estrutura de Custom Hooks para os diferentes endpoints, seria mais interessante criar um arquivo para criar uma instância do axios e utilizá-la através de import em outro arquivo que apenas exporta todas as funções dos diferentes endpoints sendo usados, assim não seria necessário criar uma instância do axios nem fazer o import dessa biblioteca para cada endpoint diferente sendo utilizado (o que não é muito escalável) e pouparia bastante código duplicado.
- Os endpoints de adição e edição (post e put) em geral recebem apenas um parâmetro (ex.: data) que é um objeto com todas as informações necessárias, não é necessário passar cada uma das informações individualmente. O mais comum é “montar” este objeto no local onde a request está sendo chamada.
- Quando está utilizando Sass é interessante usar pelo menos as features básicas como nesting e variáveis, caso contrário, faz mais sentido manter a extensão como .css pois não há ganho algum por utilizar Sass nos casos onde não se utiliza nenhuma funcionalidade dele.