Desafio Front-end

O desafio consiste em implementar uma aplicação 100% front-end, que consome um endpoint JSON Rest com capacidades de leitura e escrita, utilizando tecnologias avançadas de desenvolvimento para web.

A Aplicação

Você vai criar um aplicativo de blog em uma SPA, com design responsivo pronto para ser exibido em qualquer tamanho de tela, com as seguintes funcionalidades:

- Lista de posts, ordenados pelas postagens mais recentes. Com paginação. Exibindo os seguintes campos:
    - Título
    - Data de criação (formatada em DD/MM/YYYY HH:mm)
    - Autor
    - Imagem (resolução thumbnail (ex: 300x300))
    - Texto introdutório
    - Tags (com link para busca de posts pela tag clicada)
- Cada post deve ter um link, no titulo e na imagem, que leva o usuário para a página de detalhe do post.
- Possibilidade de alterar a ordenação de posts, a partir de um determinado campo (título, data de criação, autor) em ordem crescente ou decrescente.
- Incluir campo para busca de posts, que deve ser acessível em qualquer tela da aplicação (lista de posts, detalhe de post, etc.).

Pode ficar em uma barra lateral ao conteúdo principal ou no header da aplicação.

- Incluir lista de todas as tags, que deve ser acessível em qualquer tela da aplicação (lista de posts, detalhe de post, etc.). Deve ficar em uma barra lateral.
- Detalhe de post, com lista de comentários relacionados ao post, exibindo os seguintes campos:
    - Título
    - Data de criação (formatada em DD/MM/YYYY HH:mm)
    - Autor
    - Imagem (resolução que preencha o corpo do detalhe do post)
    - Conteúdo
    - Tags (com link para busca de posts pela tag clicada, levando para a lista de posts)
- Listagem de comentários em um detalhe de post, ordenados por data de criação decrescente, com um botão para exibir o formulário para adição de novos comentários relacionados ao post, com campos para Autor e Texto do comentário. OBS: deve ser enviado também para a API a data de criação do formulário, sendo esta a data atual.

Requisitos de tecnologia

Estes são os frameworks e bibliotecas que devem ser utilizados para a construção da aplicação. Vale lembrar que você pode utilizar outras bibliotecas para implementar funcionalidades que estas não atendem, como por exemplo, manipulação de datas e de estruturas de dados em

javascript.

- Bootstrap ou Materialize: Frameworks CSS utilizados para a construção de aplicações responsivas ou mobile first.
- SASS: Linguagem que extende as funcionalidades da linguagem CSS.
- React: Biblioteca javascript baseada em componentes para a construção de interfaces interativas com o usuário.
- React Router: Conjunto de componentes React utilizados para realizar a navegação e o roteamento entre as telas de uma SPA (Single Page Application).
- Babel: Transpilador javascript que serve para transformar código javascript com as features mais modernas (ES6) em um código que consegue ser interpretador por qualquer navegador.
- Webpack: para o build do javascript da aplicação, utilizando o Babel para transpilar o código desenvolvido.
- ES6: Atualização da linguagem javascript, com novas features que servem para facilitar e otimizar o desenvolvimento de aplicações para Web.

Requisitos da aplicação

Estes são os requisitos de UI que a aplicação deve possuir para ser aceita neste desafio.

- Responsiva: A aplicação deve adaptar o seu layout e as suas interfaces de acordo com o tamanho da tela do dispositivo do

usuário (smartphone, tablet ou desktop).

- Single Page Application: Não pode haver refresh no browser para acessar outra área da aplicação.
- Tratamento de erros: Se houver algum erro não previsto no uso da aplicação, por conta de uma falha de requisição por exemplo, deve ser implementada uma estratégia de fallback (exibir dados em cache), ou deve ser exibido para o usuário o motivo da falha de maneira clara e concisa.

API JSON Rest

Será utilizada para este desafio uma API JSON de Posts gerada pelo json-server, que é um módulo npm que produz uma API com funcionalidades de CRUD a partir de um arquivo .json. O arquivo para a API de Posts está incluso no desafio (db.json), e para subir o json-server deve ser executado os seguintes passos:

- Instalar o npm do json-server:

```
npm install -g json-server
```

- Na pasta onde está o arquivo db.json, executar o comando para iniciar o json-server:

```
json-server --watch db.json
```

Este comando irá iniciar um servidor para a API Rest na porta 3000, acessível pela url http://localhost:3000

Referência da API JSON Rest

- Listagem de Posts: [GET] /posts
    - Paginação: parâmetro **_page** para indicar o número da página. (A primeira renderização deve exibir a primeira página).
    - Ordenação: parâmetro **_sort** para indicar o nome do campo a ser ordenado, e **_order** para ordenação crescente (ASC) ou decrescente (DESC).
    - Busca: parâmetro **q** para indicar o termo de busca.
    - Busca por nome de tag: parâmetro **tags_like** para indicar o nome da tag a ser buscada.
- Detalhe de Post: [GET] /posts/{id_do_post}
- Lista de Comentários: [GET] /comments?post_id={id_do_post}
- Lista de Tags: [GET] /tags
- Criar novo Comentário: [POST] /comments
    - Corpo JSON para criação de um novo comentário:

```
{
  "post_id": "number",
  "parent_comment_id": null,
  "author": "string",
  "comment": "string",
  "creation_date": "date(ISO_8601)"
}
```

Dicas

- Utilizar o Webpack para realizar o build do SASS, construir um sourceMap para facilitar o debugging e subir um devServer para atualizar a aplicação com novas alterações no código em real time.
- Utilizar o React Router para realizar a navegação entre listagem e detalhe de posts.
- Utilizar a biblioteca momentjs para manipulação de datas.

Bônus

- OBS: Pesquisar a documentação do json-server para as chamadas REST que não estão listadas neste documento, necessárias para a implementação dos desafios bônus.
- Implementar as seguintes funcionalidades para a área de comentários:
    - Responder a um comentário (utilizando o campo parent_comment_id).
    - Editar um comentário.
- Implementar a lista de tags com efeito de *tagcloud*.
- Criar uma nova área para a administração de Posts. (Criação, Edição e Remoção de Posts)
- Criar uma nova área para a administração de Tags (Criação, Edição e Remoção de Tags), junto com um campo de busca por autocomplete para inclusão de tags na administração de Posts.
- Implementar o controle do estado da sua aplicação React

utilizando alguma biblioteca **Flux**, como por exemplo, o Mobx ou o Redux.

- Implementação de testes unitários javascript.