**KZAS – Challenge**

1. **Desafio:**

Criar um app simples que deve conter duas páginas, uma que exibe um formulário e outra que liste os dados cadastrados, permitindo, assim, criar, listar, editar e excluir os dados cadastrados.

Dados:

* + Nome completo
  + CPF
  + Telefone
  + E-mail

1. **Solução:**

Para a solução do desafio utilizei o framework JavaScript VUE.JS em conjunto com algumas bibliotecas e frameworks:

1. **BootstrapVue:**

Um framework que permite a utilização de componentes do Bootstrap dentro do Vue, garantindo uma responsabilidade à aplicação.

1. **Axios:**

Cliente HTTP baseado em Promises que permite consumir uma API retornando as respostas em JSON.

1. **Vue-router:**

Ferramenta para controle de rotas dentro da aplicação oficial para o Vue e que permite a utilização do modo History, que remove o Hash da URL e permite uma navegação sem que a página seja carregada.

1. **Vuelidate:**

Uma ferramenta de validação para o Vue que permite a declaração de regras diretamente para os dados do modelo.

1. **Vue Toasted:**

Um plugin para Vue que permite o lançamento de alertas na página, o que ajuda bastante na hora de mostrar alguns erros ou mensagens de sucesso.

* 1. **Organização das pastas**

A organização de pastas dentro da aplicação foi feita da seguinte forma:

A pasta views, será responsável pelo armazenamento dos arquivos de Páginas, no caso as páginas Home e Admin.

A pasta components, onde estarão os componentes que são ou poderiam ser usados mais de uma vez na aplicação, no caso o Form, que contém o formulário, o List, que é a tabela de listagem, e o Navbar, que é a barra de navegação.

A pasta config, que contém alguns arquivos de configuração de ferramentas ou frameworks, no caso o axios, bootstrap e Vue Toasted.

A pasta router que contém as configurações de rotas.

E a pasta Assets, que contém o arquivo de CSS.

* 1. **Estilos**

Para a estilização foi levado em consideração o guia de estilo disponibilizado.

1. **Criação do projeto**

Para dar início ao projeto, utilizei o vue cli, que criou um template de início onde pude dar início ao desenvolvimento. A partir disso, fiz uma limpeza no template, removendo alguns arquivos que não iria utilizar.

Após fiz a configuração do eslint e prettier, ferramentas que ajudam na formatação e organização do código e configurei o vue-router para duas rotas, que seria a Home e a Admin.

* 1. **Páginas**
     1. **Home**

Na página Home contém a importação e utilização do componente List.

* + 1. **Admin**

Na página Admin contém a importação e utilização do componente Form.

* 1. **Componentes**
     1. **List**

Na componente List comecei criando uma tabela, onde iria listar os dados cadastrados e ter os botões de editar e excluir. Com a tabela pronto, pude começar a povoar ela, usando o recurso disponibilizado <https://private-21e8de-rafaellucio.apiary-mock.com/users>

Assim, fiz uma requisição via Axios, e após a conclusão da Promise, armazenei esses dados no localStorage, utilizando o objeto localStorage.data, dando assim um estado inicial para a lista de dados. Um ponto importante é que a aplicação só fará essa requisição se o localStorage.data não existir, portanto coloquei essa condição. Caso já exista, a aplicação utilizará os dados já armazenados em localStorage.data.

Os campos da tabela foram definidos no objeto fields, e permite que seja feito a ordenação do maior para o menos e vice versa.

Foram definidos os botões para excluir o dado, onde de acordo com o index do array, ele encontra a posição dele dentro do localStorage.data e faz a exclusão apenas do dado escolhido, e o botão para editar, que ao ser clicado passa dados via props para o componente de Form, e mostra o formulário, permitindo assim a edição do dado escolhido.

* + 1. **Form**

Componente que contém o formulário para cadastro e edição de dados. Aqui tem a definição do atributo validations, disponibilizado pelo Vuelidate, que permite fazer a validação dos campos do formulário. Ao clicar no botão, o submit é feito é verificado se todos os campos estão preenchidos corretamente, caso não esteja ele mostra qual campo está incorreto.

De acordo com as props passadas pelo componente pai, o componente Form verifica se será feito um cadastro ou uma edição dos dados.

1. **Testes**

Essa parte foi um grande desafio pois não tinha tantos conhecimentos em testes, mas foi a parte que achei mais legal do desafio, e ver todos os testes dando certo é muito bom.

Para fazer os testes e2e, usei o Cypress e os teste foram os seguintes:

* Testar a listagem de dados;
* Testar o cadastro de um novo dado;
* Verificar se esse novo cadastro apareceu na listagem;
* Carregar os dados no form para edita-los;
* Editar o valor e mostrar na lista;
* Deletar um dado e remover da lista;