RELATÓRIO DO DESAFIO PROPOSTO PELA EMPRESA AIKO DIGITAL

Stanley Silva Sampaio

Novembro de 2021

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Este é um relatório referente ao desafio proposto pela empresa Aiko Digital.

A implementação da geolocalização em softwares cresce cada vez mais no cenário atual, pois, com essa tecnologia, é possível acompanhar, em tempo real, a localização geográfica de equipamentos, equipes, dispositivos e até mesmo aplicações.

2 OBJETIVOS

O Objetivo geral deste desafio é utilizar de conhecimentos em ReactJS, JavaScript, Typescript, HTML e CSS para implementar uma aplicação web que trata e exibe informações dos equipamentos de uma operação florestal.

Os objetivos específicos são:

- Posições dos equipamentos: Exibir no mapa os equipamentos nas suas posições mais recentes.
- Estado atual do equipamento: Visualizar o estado mais recente dos equipamentos. Exemplo: mostrando no mapa, como um pop-up, mouse hover sobre o equipamento, etc.
- Histórico de estados do equipamento: Permitir a visualização do histórico de estados de um equipamento específico ao clicar sobre o equipamento

3 METODOLOGIA

O IDE utilizado para implementação da aplicação foi o Visual Studio Code (https://code.visualstudio.com/)

Recursos ultilizados:

- ReactJS
- TypeScript
- JavaScript
- HTML

- CSS

A biblioteca escolhida para construir o mapeamento da aplicação foi a Leaflet. O motivo da escolha é o fato dessa biblioteca ser de código aberto.

A empresa Aiko forneceu acesso à um repositório no GitHub com os dados referentes ao projeto. Foi realizado um Fork, e criada uma branch, que serviu para desenvolvimento e versionamento da aplicação em questão.

Após a conclusão da aplicação, foi realizado um pull request da branch.

Para termos de desenvolvimento, foi adotado um nome fictício para a aplicação: AEMS - aiko equipment monitoring system.

4 DESENVOLVIMENTO:

4.1 Posição dos equipamentos:

Inicialmente, o repositório fornecido foi clonado, e então, iniciou-se o projeto ReactJS, o comando utilizado foi:

npx create-react-app teste-frontend-estagio-v2 –template typescript.

Após a criação do projeto, foram instaladas as bibliotecas e dependências necessárias para funcionamento da aplicação, entre elas: Leaflet, react-leaflet, react-dom-leaflet. Eslint.

Então, a aplicação foi reorganizada de acordo com os costumes do desenvolvedor, seguindo a idéia de clean code.

Foi definido que o mapa da aplicação seria um componente, de modo que todo o consumo e tratamento de informações fossem realizados dentro do próprio componente. O componente em questão é o O arquivo “MapComponent.tsx”.

Criou-se uma constante (LeafletMarkers) de array de Markers (componente responsável por plotar a localização do equipamento no mapa). O arquivo “equipmentPositionHistoy.json” possui as informações de geolocalização de todos os equipamentos (um array de arrays com o histórico de posições, e o ID

referente ao equipamento posicionado no local em questão). Desse modo, na criação de cada Marker, foi utilizado o método **map** para percorrer todo arquivo, em conjunto com o método **sort**, para ordenar as localizações do equipamento, de modo que a mais atual se alocasse na posição [0] do vetor relacionado à determinado ID. O marker também possui uma Popup (caixa flutuante), uma ul(lista) e um botão, tais elementos-filhos foram criados para auxiliar na manipulação e exibição das informações.

OBS: No momento da criação do Marker, todas as informações solicitadas no desafio são coletadas.

```
function MapComponent() {

  const LeafletMarkers = locData.map(marker => {

    marker.positions = marker.positions.sort((a, b) => (new Date(b.date).getTime() - new Date(a.date).getTime()))
```

Figura 1: Trecho do código onde os métodos map e sort são ultilizados.

Após criado o vetor de objetos do tipo Marker, o mesmo é chamado dentro do elemento <MapContainer>, renderizando todos Markers criados anteriormente.

4.2 Estado atual e Histórico de estados do equipamento:

Ao obter a posição, foi criada uma key contendo o ID do equipamento, dessa forma, é possivel obter todas as informações do equipamento relacionado àquela ID, usando o ID como key para varrer os outros arquivos.

Para isso, foram criadas as seguintes funções:

- **equipmentName**: Função que recebe um ID, filtra o arquivo “equipment.json”, e retorna o vetor que contem o ID recebido, o nome do equipamento e o ID do modelo do equipamento. O retorno é o nome do equipamento.
- **equipmentModelId**: Similar à “equipmentName”, porém, pegamos a informação “equipmentModelId”, e usamos esse dado para filtrar outro arquivo, o “equipmentModel.json”, pois, dessa forma, conseguimos associar o ID ao nome do modelo, que é o retorno dessa função.

- **equipmentState**: Com o ID do equipamento, Buscamos no arquivo “equipmentStateHistory.json” o ID do estado mais atual do equipamento. Após obter a informação, filtramos o arquivo “equipmentState.json”, que associa o ID do estado ao nome. O retorno da função é o nome do estado mais recente.
- **getStateHistory**: O primeiro passo é semelhante à função anterior, porém, buscamos o array que contém todos os estados de determinado equipamento (ID), e aplicamos o método sort para ordenar esse vetor e ter o retorno da função. (histórico de estados do equipamento).
- **getStateName**: Essa função auxilia na organização do histórico de estados do equipamento, retornando o nome de todos os estados do vetor contendo o Histórico de estados do equipamento.
- **showHistory**: Essa função pertence ao botão inserido em cada marker, e tem como finalidade alterar a propriedade CSS do elemento <ul> (lista), pois, como essas informações foram adquiridas anteriormente, a lista começa com a propriedade **display:none**, dessa forma, ao clicar no botão (“ver mais”), a lista fica visível.

```
function equipmentName(id: string) {
  var index = eqData.filter(x => x.id === id);
  return index[0].name;
}

function equipmentModelId(id: string) {
  var index = eqData.filter(x => x.id === id)[0].equipmentModelId;
  var model = modelData.filter(x => x.id === index)[0].name;
  return model;
}

function equipmentState(id: string) {
  var equipmentStates = stateHistoryData.filter((x) => x.equipmentId === id)[0];
  equipmentStates.states = equipmentStates.states.sort((a, b) => (new Date(b.date).getTime() - new Date(a.date).getTime())
  var stateId = equipmentStates.states[0].equipmentStateId;

  return stateData.filter((state) => state.id === stateId)[0].name;
}

function getStateHistory(id: string) {
  var equipmentStates = stateHistoryData.filter((x) => x.equipmentId === id)[0];
  return equipmentStates.states.sort((a, b) => (new Date(b.date).getTime() - new Date(a.date).getTime()))
}

function getStateName(id: string) {
  return stateData.filter((state) => state.id === id)[0].name;
}

function showHistory(id: string) {
  let el = document.getElementById(id);
  if (el?.classList.contains("hide")) {
    el.classList.remove('hide');
    el.classList.add('show');
    return undefined;
```

Figura 2: Trecho do código com a implementação das funções descritas.

Com as funções descritas acima, foi possível coletar e exibir todas as informações solicitadas, o resultado final foi o seguinte (figura 3):

Figura 3: Aplicação web AEMS – versão entregue.

5 CONCLUSÃO

Os conceitos utilizados nessa aplicação são fundamentais para o desenvolvimento front-end de softwares destinados à geolocalização, e contribuem, cada vez mais, para o crescimento e evolução da computação, além da melhoria de gestão de várias empresas.

O desafio foi gratificante e de enorme aprendizado, agradeço à empresa Aiko pela experiência proporcionada.