Documentação - Teste frontend Aiko

1. Introdução

1.1 Descrição:

O projeto consiste em uma aplicação frontend que consome dados de equipamentos florestais e exibe essas informações em uma interface web. A aplicação inclui um mapa que mostra a localização de cada equipamento, além de seu histórico de posições. Também há uma lista detalhada dos equipamentos, contendo informações específicas como o histórico de estados (Operando, Parado, Manutenção) e a produtividade de cada um.

1.2 Pré-requisitos:

Para rodar o projeto em ambiente de desenvolvimento, é necessário ter os seguintes softwares instalados:

Node.js: Para executar o ambiente JavaScript no servidor.
Next.js: Para gerenciamento das rotas, renderização do lado do servidor e construção da interface.

2. Estrutura do projeto:

- */public*: Contém arquivos públicos do projeto, como dados estáticos e imagens.
- */src*: Diretório principal com todo o código-fonte da aplicação.
- */app*: Armazena os arquivos que serão usados como rotas no projeto, além de arquivos de configuração do Next.js.
- */src/components*: Contém componentes reutilizáveis, como a barra de navegação e os elementos das telas.
- */src/styles*: Armazena os arquivos de estilo, organizando o design visual do projeto.

3. Instalação e execução:

Após instalar os softwares descritos na introdução, e clonar o repositório, basta executar no terminal os seguintes comandos

yarn //para instalar as dependências do projeto

yarn run dev //para rodar em servidor local

Abra o navegador e acesse o endereço http://localhost:3000/ para visualizar a aplicação em execução

4. Tecnologias utilizadas:

- Next.js: Framework React utilizado para renderização do lado do servidor e geração de páginas estáticas.
- Styled-components: Biblioteca para gerenciamento e escopo de estilos CSS dentro dos componentes React.
- OpenLayers: Biblioteca para renderização do mapa com dados de geolocalização dos equipamentos.
- Prettier: Formatador de código.
- Material UI (MUI): Biblioteca de componentes React para criar a barra de navegação e outros elementos da interface de maneira responsiva e acessível.

5. Guia de Componentes:

5.1 Componente: HomeMap

Função:
O componente exibe um mapa interativo que mostra a posição atual e o histórico de posições dos equipamentos. Os usuários podem clicar nos ícones dos equipamentos para visualizar seu histórico, e o mapa fornece um "tooltip" com informações detalhadas sobre cada equipamento quando o mouse passa por cima deles.

Props e Tipos:
Este componente não recebe props diretamente, mas utiliza os dados importados dos arquivos JSON (como equipment.json, equipmentPositionHistory.json, etc.) para renderizar as informações dos equipamentos no mapa.

Principais Funções:
getLastPosition(positions: Position[])
Retorna a última posição registrada de um equipamento com base em um array de posições.
Retorno: Última posição (Position).

getLastState(states: StateHistory[])
Retorna o último estado registrado de um equipamento a partir de um array de históricos de estados.
Retorno: Último estado (StateHistory).

renderInitialPins()
Renderiza os ícones dos equipamentos no mapa usando as últimas posições conhecidas de cada um.
Função: Cria os ícones SVG e estiliza os pinos de cada equipamento com cores baseadas no modelo do equipamento.

renderHistoryPins(positions: Position[])

Renderiza os pinos do histórico de posições de um equipamento selecionado, desenhando uma linha ligando as diferentes posições e aplicando um estilo diferente para cada pino.
Função: Renderiza o caminho completo percorrido por um equipamento com base em seu histórico de posições.

handleReset()
Limpa a seleção atual de equipamentos e restaura a visualização inicial dos pinos.
Função: Reinicia a visualização para o estado inicial, removendo o histórico exibido.

Exemplo de Uso:
Este componente é usado diretamente na tela de mapas da aplicação. Para usá-lo, basta importá-lo e renderizá-lo sem props

5.2 Componente: listEquipamentos

Função:
O componente listEquipamentps gera uma listagem dos equipamentos disponíveis, exibindo informações detalhadas como nome, modelo, estado atual e localização. Ele se comunica com o mapa para sincronizar a exibição das informações.

Props e Tipos:
Assim como HomeMap, este componente não recebe props diretamente. Ele utiliza dados JSON (como equipment.json, equipmentState.json, etc.) para listar os equipamentos.

Principais Funções:
getLastPosition(positions: Position[])
Retorna a última posição conhecida do equipamento, da mesma forma que no componente HomeMap.

getLastState(states: StateHistory[])
Retorna o último estado registrado do equipamento, similar ao uso no componente HomeMap.

renderEquipmentList()
Cria uma lista dos equipamentos com base nas informações atuais, exibindo nome, estado e modelo. Permite interação para visualizar mais detalhes no mapa.
Função: Mostra a lista de equipamentos na interface com seus dados principais, e dispara ações ao clicar nos itens.

handleEquipmentClick(equipmentId: string)
Executa ações ao clicar em um equipamento da lista, sincronizando a seleção no mapa.
Função: Altera o equipamento selecionado e exibe seu histórico no mapa.

Exemplo de Uso:
Esse componente também pode ser renderizado diretamente em uma tela de listagem de equipamentos

6. Rotas:

/(public): Diretório que contém as rotas de acesso público, sem necessidade de autenticação.
/(public)/page.tsx: Componente que representa a página inicial da aplicação, onde são exibidas informações gerais do sistema.
/(public)/equipamento: Rota responsável por renderizar a tela de listagem de equipamentos, exibindo dados detalhados como estados, produtividade e localização.

7. Demonstração do sistema:

Link do vídeo: https://drive.google.com/file/d/1oy-HKiGH8ReXLKy2lW4OFLln1Ov331LD/view?usp=sharing

8. Boas práticas:

- Modularização: As telas e seus componentes são organizados de forma modular. Cada tela é dividida em componentes menores e reutilizáveis, facilitando a manutenção e expansão do código.
- Padronização de Nomes: Segue-se a convenção camelCase para a nomeação de variáveis e funções, garantindo consistência no código e alinhamento com as melhores práticas do desenvolvimento JavaScript/TypeScript.

9. Documentações externas úteis:

1. Next: https://nextjs.org/docs
2. OpenLayers: https://openlayers.org/doc/
3. Mui: https://mui.com/material-ui/getting-started/