Weather App

Documento de Arquitetura de Software

Versão <1.0>6

Documento de Arquitetura de Software

1. Introdução

Está sendo desenvolvido no curso de Javascript da EBAC, uma aplicação utilizando HTML, CSS, Javascript, React e Vite para uma tela de meteorologia.A primeira premissa do software é trazer um card indicando a cidade do usuário, seguido pela temperatura atual.

No segundo container, ou divisão do site, deve conter as informações sobre a temperatura dos próximos cinco dias, indicando a data, temperatura mínima e máxima.

Inicialmente os dados serão fictícios, mas posteriormente, deve estar aberto a receber uma API externa para se conectar à aplicação.

1.1. Finalidade

Esse software será desenvolvido como projeto final do curso, para que o aluno possa aplicar seus conhecimentos de HTML, CSS, JavaScript e bibliotecas no desenvolvimento de uma aplicação final para o usuário.

1.2. Escopo

A aplicação tem como foco a exibição de previsões meteorológicas para uma localização específica, com possibilidade de expansão para consultas em múltiplas cidades em uma versão 2.

1.3. Definições, Acrônimos e Abreviações

- **API**: Interface de Programação de Aplicativos
- **React**: Biblioteca JavaScript para criação de interfaces de usuário
- **Vite**: Ferramenta de build e desenvolvimento rápido para projetos front-end
- **JSON**: Formato de intercâmbio de dados utilizado na comunicação com APIs

1.4. Visão Geral

Este documento está organizado nas seguintes seções:

- **Representação Arquitetural**: Estrutura do software e suas camadas.
- **Metas e Restrições**: Requisitos que influenciam a arquitetura.
- **Visão de Casos de Uso**: Como a aplicação será utilizada.
- **Visão Lógica**: Estrutura do código-fonte.
- **Visão de Processos**: Fluxo de execução do software.
- **Visão de Implantação**: Onde e como o sistema será executado.

2. Representação Arquitetural

A arquitetura da aplicação é baseada em componentes do React e segue um modelo modular e escalável. As principais visões da arquitetura incluem:

Visão de Casos de Uso

- Define como os usuários interagem com o sistema.
- Inclui fluxos como obtenção da previsão atual e visualização da previsão futura.

Visão Lógica

- Organiza o software em camadas e componentes reutilizáveis.
- Inclui a interface do usuário, a camada de serviços e o gerenciamento de estado.

Visão de Processos

- Mostra como os dados fluem dentro da aplicação e como as requisições são tratadas.
- Destaca a obtenção de dados da API e a renderização dinâmica dos componentes.

Visão de Implantação

- Define como a aplicação será hospedada e acessada.
- Considera a implementação utilizando Vercel ou Netlify.

Visão de Implementação

- Especifica a estrutura dos arquivos e diretórios.
- Inclui diretórios para componentes, serviços, estilos e hooks personalizados.

3. Metas e Restrições da Arquitetura

- **Responsividade**: O design deve se adaptar a diferentes tamanhos de tela.
- **Eficiência**: Utilização otimizada de requisições HTTP para reduzir o consumo de dados.
- **Extensibilidade**: Estrutura flexível para futura integração com APIs meteorológicas reais.

4. Visão de Casos de Uso

Caso de Uso 1: Obter Previsão Atual

- O usuário acessa a aplicação e visualiza a temperatura atual de sua cidade.
- O sistema obtém a localização através do navegador e retorna os dados meteorológicos.

Caso de Uso 2: Visualizar Previsão para os Próximos Dias

- O sistema exibe a previsão do tempo para os próximos cinco dias.

5. Visão Lógica

A aplicação é estruturada nos seguintes componentes principais:

- **Header**: Exibe o nome da aplicação.
- **WeatherCard**: Apresenta a previsão do dia atual.
- **ForecastList**: Lista a previsão dos próximos dias.
- **API Service**: Responsável por buscar dados da API.

5.1. Visão Geral

A aplicação segue a arquitetura de componentes do React

6. Visão de Processos

1. O usuário acessa a página.
2. O sistema busca os dados meteorológicos na API.
3. Os componentes React são atualizados com as informações.
4. O usuário pode buscar outra cidade manualmente através da barra de pesquisa. (Versão 2)
5. Os dados da nova cidade são carregados e exibidos na interface. (Versão 2)

7. Visão de Implantação

A aplicação será implantada utilizando Vercel ou Netlify, permitindo acesso via web sem necessidade de servidor backend.

8. Visão da Implementação

A implementação segue uma estrutura modular:

- **/src/components**: Contém os componentes React.
- **/src/services**: Implementação das requisições HTTP. (Versão 2)
- **/src/styles**: Contém arquivos de estilo CSS.

9. Tamanho e Desempenho

A aplicação foi projetada para ser leve e eficiente, garantindo um tempo de carregamento rápido mesmo em conexões mais lentas. As principais considerações incluem:

- **Otimização do Bundle**: Utilização do Vite para compilar e otimizar os arquivos JavaScript e CSS.

- **Limitação de Requisições**: Para evitar sobrecarga, as requisições à API são armazenadas em cache durante a sessão do usuário.
- **Responsividade**: Adaptação fluida da interface para diferentes tamanhos de tela, minimizando processamento desnecessário em dispositivos móveis.
- **Escalabilidade**: A arquitetura permite fácil expansão para novas funcionalidades sem comprometer o desempenho atual.

10. Qualidade

- **Extensibilidade**: Possibilidade de integração com APIs externas.
- **Confiabilidade**: Exibição precisa dos dados.
- **Portabilidade**: Pode ser executada em diferentes dispositivos e navegadores.