UniSENAI

Trilha 07

Integração com APIs no Vue.js

7 - Integração com APIs no Vue.js

A integração com APIs é uma das tarefas mais comuns no desenvolvimento web moderno. No Vue.js, essa funcionalidade permite que a aplicação interaja com **APIs RESTful** para buscar, enviar e atualizar dados dinamicamente. Nesta unidade, abordaremos a **configuração do Axios**, a **integração com APIs públicas** e a **manipulação de respostas e tratamento de erros**.

7.1 Consumindo APIs no Vue.js

O que é uma API?

Uma **API (Application Programming Interface)** é um conjunto de regras e definições que permite que diferentes sistemas se comuniquem. No contexto do desenvolvimento web, APIs são frequentemente utilizadas para permitir que um cliente (front-end) acesse um banco de dados ou serviço hospedado em um servidor (back-end).

Por que consumir APIs no Vue.js?

✔ Permite **atualizar dados dinamicamente** sem recarregar a página.
✔ Facilita a **integração de serviços externos**, como APIs de clima, CEP, conversão de moedas etc.
✔ Separa a lógica do **front-end** e do **back-end**, tornando o desenvolvimento modular e escalável.

7.2 Configuração de Axios

O que é Axios?

O **Axios** é uma biblioteca JavaScript baseada em Promises que facilita o consumo de APIs no Vue.js. Ele é amplamente utilizado por sua **simplicidade**, **suporte a requisições assíncronas** e **facilidade no tratamento de erros**.

Instalação do Axios

Para utilizar o Axios em um projeto Vue.js, primeiro instalamos a biblioteca:

```
npm install axios
```

Após a instalação, podemos importar e configurar o Axios no projeto.

Configuração Global do Axios

Podemos configurar o Axios globalmente para definir uma **baseURL** padrão e incluir cabeçalhos personalizados. Isso facilita a organização das chamadas à API.

Crie um arquivo de configuração `src/services/api.js`:

```
import axios from 'axios';

const api = axios.create({
  baseURL: 'https://jsonplaceholder.typicode.com', // Exemplo de API pública
  timeout: 5000, // Tempo máximo de resposta (5 segundos)
  headers: {
    'Content-Type': 'application/json'
  }
});

export default api;
```

Agora, podemos importar essa configuração onde for necessário:

```
import api from '@/services/api';
```

7.3 Integração com APIs Públicas

Podemos utilizar o Axios para **buscar dados de APIs públicas**, como o **ViaCEP** para consulta de CEPs ou o **OpenWeatherMap** para informações meteorológicas.

Exemplo 1: Consulta de CEP com ViaCEP

A **API ViaCEP** retorna informações sobre endereços a partir de um CEP informado.

📌 **URL da API**: `https://viacep.com.br/ws/{CEP}/json/`

Implementação no Vue.js

Consulta De Endereço

```vue
<template>
  <div>
    <h2>Consulta de Endereço</h2>
    <input v-model="cep" placeholder="Digite o CEP" />
    <button @click="buscarEndereco">Buscar</button>

    <div v-if="endereco">
      <p><strong>Rua:</strong> {{ endereco.logradouro }}</p>
      <p><strong>Bairro:</strong> {{ endereco.bairro }}</p>
      <p><strong>Cidade:</strong> {{ endereco.localidade }} - {{ endereco.uf }}</p>
    </div>
  </div>
</template>

<script>
import axios from 'axios';

export default {
  data() {
    return {
      cep: '',
      endereco: null
    };
  },
  methods: {
    async buscarEndereco() {
      if (this.cep.length === 8) {
        try {
          const response = await axios.get(`https://viacep.com.br/ws/${this.cep}/json/`);
          this.endereco = response.data;
        } catch (error) {
          console.error("Erro ao buscar CEP:", error);
        }
      } else {
        alert("CEP inválido! Digite 8 números.");
      }
    }
  }
};
</script>
```

Explicação:

- O usuário digita o CEP no input.
- Ao clicar no botão **"Buscar"**, é feita uma **requisição GET** à API do ViaCEP.
- A resposta da API retorna o endereço correspondente e exibe os dados na tela.

Exemplo 2: Consulta de Clima com OpenWeatherMap

A API do **OpenWeatherMap** permite obter dados meteorológicos de qualquer cidade.

📌 **URL da API**:
`https://api.openweathermap.org/data/2.5/weather?q={cidade}&appid={API_KEY}&units=metric`

Para utilizar essa API, é necessário obter uma chave gratuita no site:
🔗 https://openweathermap.org/api

Implementação no Vue.js

Previsão Do Tempo

```vue
<template>
  <div>
    <h2>Previsão do Tempo</h2>
    <input v-model="cidade" placeholder="Digite a cidade" />
    <button @click="buscarClima">Buscar</button>

    <div v-if="clima">
      <p><strong>Temperatura:</strong> {{ clima.temp }}°C</p>
      <p><strong>Clima:</strong> {{ clima.descricao }}</p>
    </div>
  </div>
</template>

<script>
import axios from 'axios';

export default {
  data() {
    return {
      cidade: '',
      clima: null
    };
  },
  methods: {
    async buscarClima() {
      const API_KEY = "SUA_CHAVE_AQUI"; // Substitua pela sua chave de API
      try {
        const response = await axios.get(
          `https://api.openweathermap.org/data/2.5/weather?q=${this.cidade}
          &appid=${API_KEY}&units=metric&lang=pt`);
        this.clima = {
          temp: response.data.main.temp,
          descricao: response.data.weather[0].description
        };
      } catch (error) {
        console.error("Erro ao buscar clima:", error);
      }
    }
  }
};
</script>
```

Explicação:

- O usuário insere o nome da cidade e clica em **"Buscar"**.
- O Vue.js usa o **Axios** para chamar a API do OpenWeatherMap.
- A temperatura e a descrição do clima são exibidas na interface.

7.4 Manipulação de Respostas e Tratamento de Erros

Nem todas as requisições retornam **respostas bem-sucedidas**. É essencial tratar **erros de rede**, **tempo limite** e **respostas inesperadas** para melhorar a experiência do usuário.

Tipos de Erros Comuns

1. **Erro de rede:** A API pode estar fora do ar ou o usuário pode estar sem internet.
2. **Requisição inválida:** O servidor pode retornar erro `400 (Bad Request)` devido a um dado incorreto.
3. **Erro de autenticação:** A API pode exigir um token de autenticação (`401 Unauthorized`).
4. **Erro interno do servidor:** Pode ocorrer um `500 (Internal Server Error)` se houver falha no back-end.

Exemplo: Tratamento de Erros no Axios

Buscar Dados

```
async function buscarDados() {
  try {
    const response = await axios.get('https://api.exemplo.com/dados');
    console.log("Dados recebidos:", response.data);
  } catch (error) {
    if (error.response) {
      console.error("Erro na resposta da API:", error.response.status, error.response.data);
    } else if (error.request) {
      console.error("Erro de conexão: Nenhuma resposta recebida.");
    } else {
      console.error("Erro desconhecido:", error.message);
    }
  }
}
```

Conclusão

✔ O **Axios** facilita o consumo de **APIs RESTful** no Vue.js.
✔ **APIs públicas**, como **ViaCEP e OpenWeatherMap**, permitem obter dados externos de forma dinâmica.
✔ O **tratamento de erros** melhora a **experiência do usuário** e evita falhas inesperadas.

Com esse conhecimento, você pode **integrar qualquer API ao Vue.js** e criar aplicações interativas e funcionais! 🚀

Lista de Atividades - Integração com APIs no Vue.js

Questões Teóricas (10 questões)

1. **O que é uma API RESTful e por que ela é amplamente utilizada no desenvolvimento web?**
2. **Explique o que é o Axios e quais são suas vantagens em relação ao uso da Fetch API para consumir APIs no Vue.js.**
3. **Quais são os principais métodos HTTP utilizados em requisições a APIs e para que cada um deles é utilizado?**
4. **O que é um `timeout` no Axios e por que ele é importante ao realizar requisições para APIs?**
5. **Explique o conceito de CORS e como ele afeta a comunicação entre um front-end Vue.js e um back-end remoto.**
6. **Qual é a diferença entre uma API pública e uma API que exige autenticação? Dê exemplos de cada uma.**
7. **Por que é essencial tratar erros ao consumir APIs? Quais são os principais tipos de erros que podem ocorrer?**
8. **O que é um cabeçalho HTTP e como ele pode ser utilizado ao realizar requisições a uma API?**
9. **Qual é o papel do `baseURL` ao configurar uma instância do Axios? Como isso facilita o desenvolvimento?**
10. **Explique a diferença entre `async/await` e `.then/.catch` ao trabalhar com requisições assíncronas em Vue.js.**

Questões Práticas (10 questões)

1. **Instale e configure o Axios em um projeto Vue.js. Teste uma requisição a uma API pública e exiba a resposta no console.**
2. **Crie um serviço (`api.js`) para centralizar as requisições HTTP do seu projeto Vue.js, definindo uma `baseURL` padrão.**
3. **Implemente uma funcionalidade que permita buscar um endereço a partir de um CEP utilizando a API do ViaCEP.**
4. **Crie um componente Vue.js que exiba informações climáticas de uma cidade utilizando a API do OpenWeatherMap.**
5. **Modifique o exemplo de consulta de CEP para exibir um erro caso o usuário insira um CEP inválido.**
6. **Adapte a integração com a API do OpenWeatherMap para permitir que o usuário selecione a unidade de temperatura (`Celsius` ou `Fahrenheit`).**
7. **Implemente um sistema de carregamento (`loading`) que exiba um indicador enquanto a requisição à API está sendo processada.**
8. **Crie um mecanismo de tratamento de erros para exibir mensagens amigáveis ao usuário caso ocorra um erro na requisição à API.**

9. **Utilizando Vue.js e Axios, desenvolva um CRUD simples consumindo uma API externa que permita adicionar, listar, editar e excluir itens.**
10. **Crie um sistema de autenticação que realize login em uma API protegida e armazene o token JWT para futuras requisições.**

- *Responda todas as questões teóricas e práticas.*
- *Organize as respostas em um arquivo PDF, contendo nome e data.*
- *Envie o PDF aos professores responsáveis, seguindo o padrão de nomenclatura.*
- *Valide o material com um professor antes de prosseguir.*
- *Certifique-se de cumprir o prazo de entrega.*

Padrão de nomeclatura
NomeCompleto_TrilhaX_DataDeEntrega.pdf
Exemplo: JoãoSilva_Trilha1_2025-01-30.pdf