**Questões Teóricas**

**1. O que é uma API RESTful e por que ela é amplamente utilizada no desenvolvimento web?**

Uma API RESTful é uma interface de comunicação entre sistemas que segue os princípios da arquitetura REST, utilizando o protocolo HTTP para troca de dados de forma simples e padronizada. Ela é amplamente utilizada no desenvolvimento web porque permite escalabilidade, reutilização e integração entre diferentes aplicações, independentemente da linguagem ou plataforma utilizadas. Além disso, sua estrutura baseada em recursos e métodos HTTP facilita a implementação e a manutenção de sistemas distribuídos.

**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.**

O Axios é uma biblioteca JavaScript voltada para realizar requisições HTTP, bastante utilizada em frameworks como Vue.js. Sua principal vantagem em relação à Fetch API é oferecer uma interface mais simples e recursos adicionais, como interceptores de requisições e respostas, tratamento automático de dados JSON e configuração global de parâmetros. Esses recursos tornam o desenvolvimento mais ágil e reduzem a repetição de código em projetos maiores.

**3. Quais são os principais métodos HTTP utilizados em requisições a APIs e para que cada um deles é utilizado?**

Os principais métodos HTTP são GET, POST, PUT, PATCH e DELETE. O método GET é usado para recuperar informações de um servidor, enquanto o POST envia dados para criar novos recursos. O PUT substitui um recurso existente por outro, e o PATCH realiza atualizações parciais em um recurso já existente. Já o DELETE é utilizado para remover recursos do servidor. Cada método segue convenções que tornam a comunicação mais clara e previsível.

**4. O que é um timeout no Axios e por que ele é importante ao realizar requisições para APIs?**

Um timeout no Axios é o tempo máximo de espera configurado para uma requisição antes que ela seja automaticamente cancelada. Esse recurso é importante porque evita que o sistema fique indefinidamente aguardando respostas de um servidor lento ou indisponível. Dessa forma, melhora-se a experiência do usuário, garantindo feedback mais rápido e permitindo que o aplicativo adote estratégias alternativas em caso de falhas.

**5. Explique o conceito de CORS e como ele afeta a comunicação entre um front-end Vue.js e um back-end remoto.**

CORS, ou Cross-Origin Resource Sharing, é um mecanismo de segurança dos navegadores que controla quais domínios podem acessar recursos de um servidor. No contexto de um front-end em Vue.js, caso ele tente consumir dados de um back-end remoto em outro domínio, a requisição pode ser bloqueada se o servidor não permitir explicitamente esse acesso. Assim, a configuração adequada de CORS no servidor é fundamental para viabilizar a comunicação entre aplicações hospedadas em diferentes origens.

**6. Qual é a diferença entre uma API pública e uma API que exige autenticação? Dê exemplos de cada uma.**

Uma API pública é aquela que pode ser acessada livremente, sem necessidade de credenciais, como algumas APIs de informações climáticas ou de moedas. Já uma API que exige autenticação requer um processo de identificação do usuário, geralmente por meio de tokens ou chaves de acesso, garantindo maior segurança e controle de uso. Exemplos incluem a API do Twitter ou do Google Maps, que limitam o acesso apenas a clientes devidamente autenticados.

**7. Por que é essencial tratar erros ao consumir APIs? Quais são os principais tipos de erros que podem ocorrer?**

Tratar erros ao consumir APIs é essencial para garantir a resiliência do sistema e a boa experiência do usuário, prevenindo falhas inesperadas. Os principais tipos de erros incluem erros de rede, como indisponibilidade do servidor; erros de autenticação e autorização, quando as credenciais não são aceitas; e erros de validação, quando os dados enviados não seguem o formato esperado. O tratamento adequado permite informar o usuário e aplicar soluções alternativas.

**8. O que é um cabeçalho HTTP e como ele pode ser utilizado ao realizar requisições a uma API?**

Um cabeçalho HTTP é um conjunto de informações adicionais enviadas junto a uma requisição ou resposta, definindo parâmetros de comunicação entre cliente e servidor. Ao consumir uma API, cabeçalhos podem ser usados para especificar o formato esperado de dados, incluir tokens de autenticação ou controlar o cache. Eles são fundamentais para a correta interpretação e segurança das interações entre sistemas.

**9. Qual é o papel do baseURL ao configurar uma instância do Axios? Como isso facilita o desenvolvimento?**

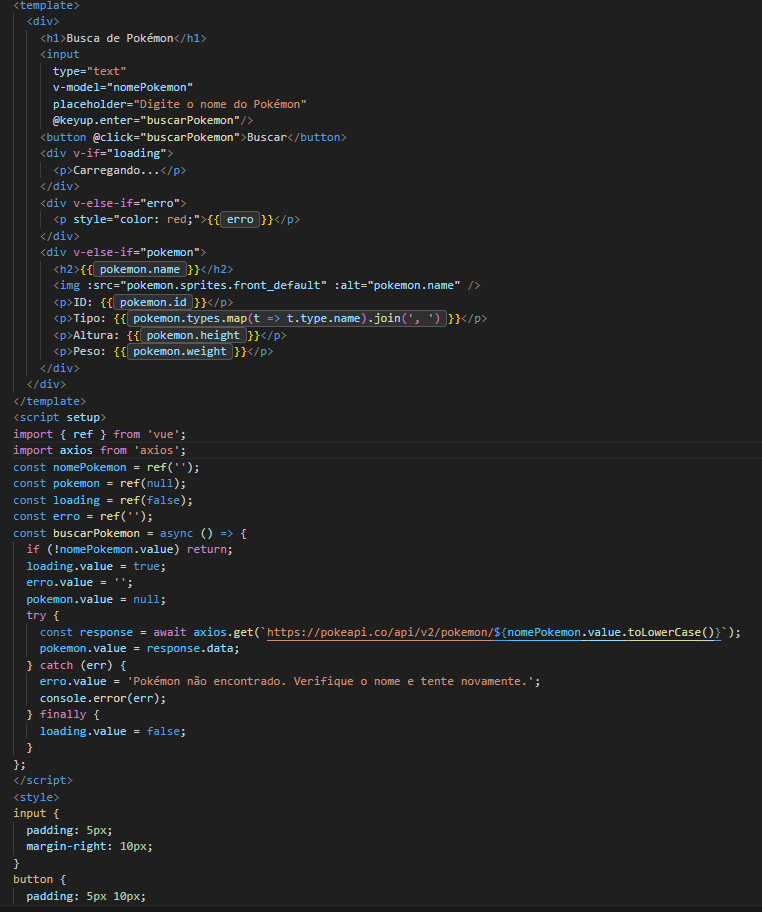
O baseURL define o endereço base que será automaticamente utilizado em todas as requisições feitas por uma instância do Axios. Isso facilita o desenvolvimento porque elimina a necessidade de repetir manualmente o mesmo domínio ou rota inicial em cada requisição. Além disso, centraliza a configuração, tornando mais fácil ajustar o endpoint da API em ambientes diferentes, como desenvolvimento, testes e produção.

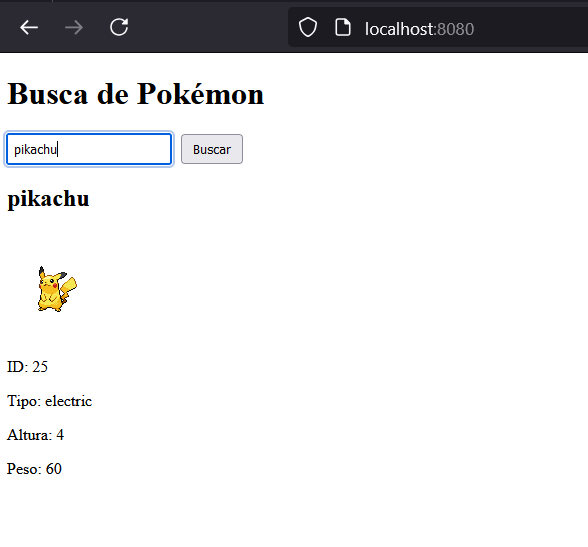
**10. Explique a diferença entre async/await e .then/.catch ao trabalhar com requisições assíncronas em Vue.js.**

O uso de async/await permite escrever código assíncrono de forma mais próxima ao síncrono, facilitando a leitura e o tratamento sequencial de requisições. Já o uso de .then/.catch utiliza promessas diretamente, criando uma cadeia de callbacks que pode se tornar menos legível em operações complexas. Embora ambas as abordagens sejam válidas, async/await costuma ser preferida em projetos Vue.js modernos pela clareza e simplicidade que oferece ao fluxo do código.

**Questões Práticas**

**1. Instale e configure o Axios em um projeto Vue.js. Teste uma requisição a uma API pública e exiba a resposta no console.**





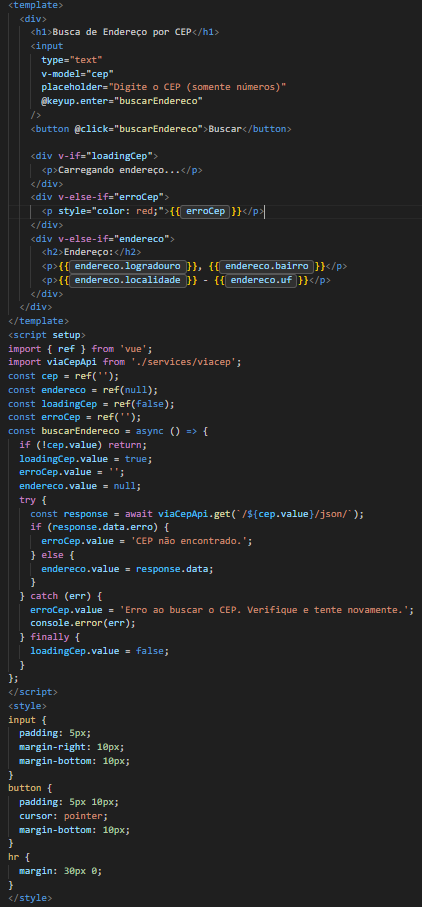
**2. Crie um serviço (api.js) para centralizar as requisições HTTP do seu projeto Vue.js, definindo uma baseURL padrão.**

****

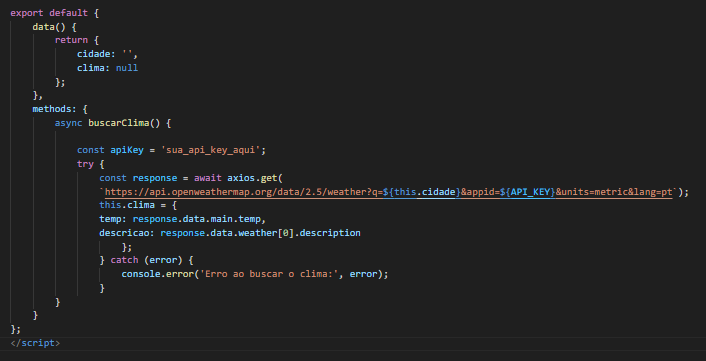
3. **Implemente uma funcionalidade que permita buscar um endereço a partir de um CEP utilizando a API do ViaCEP.**



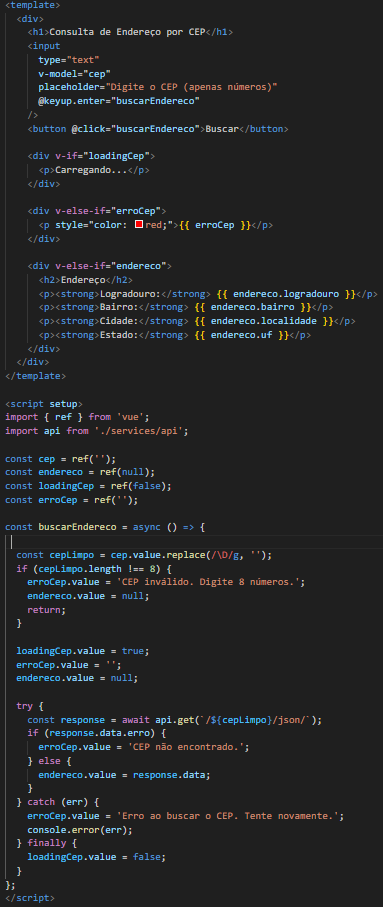
**3. Implemente uma funcionalidade que permita buscar um endereço a partir de um CEP utilizando a API do ViaCEP.**



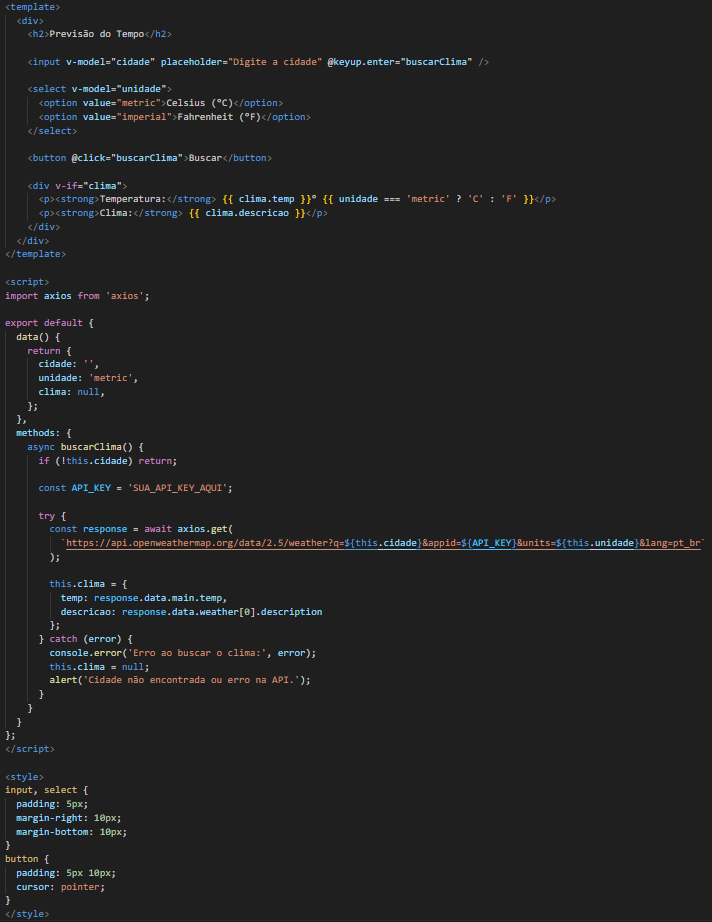
**4. Crie um componente Vue.js que exiba informações climáticas de uma cidade utilizando a API do OpenWeatherMap.**

****

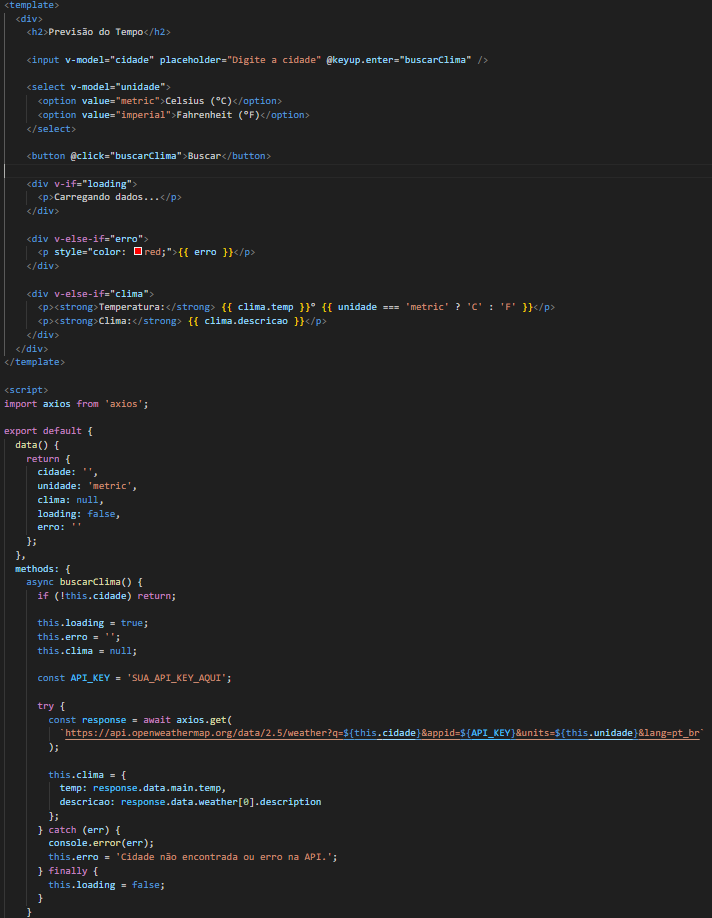
**5. Modifique o exemplo de consulta de CEP para exibir um erro caso o usuário insira um CEP inválido.**



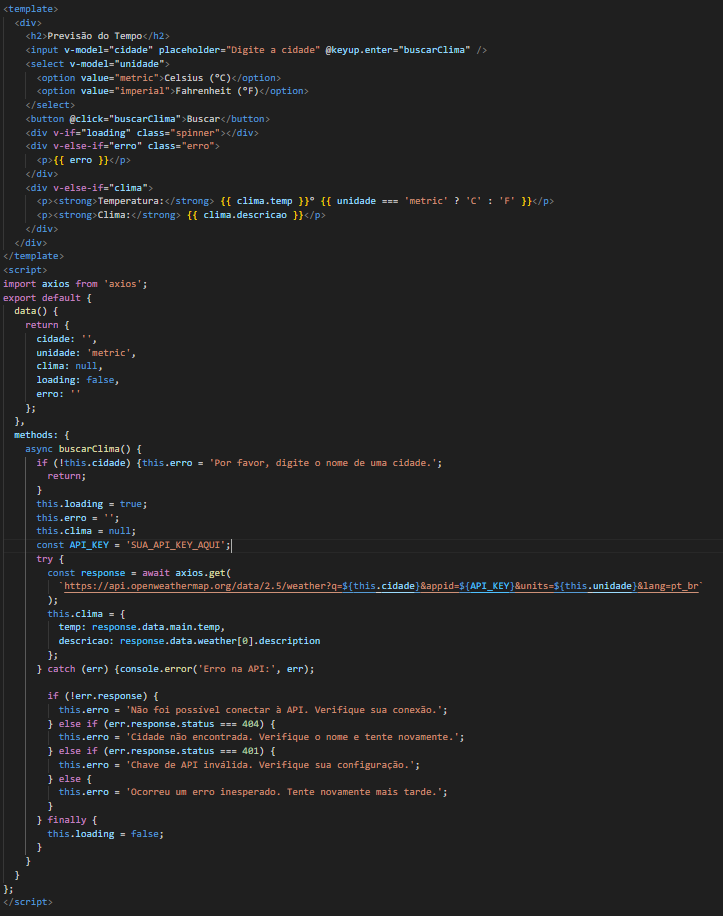
**6. Adapte a integração com a API do OpenWeatherMap para permitir que o usuário selecione a unidade de temperatura (Celsius ou Fahrenheit).**



**7. Implemente um sistema de carregamento (loading) que exiba um indicador enquanto a requisição à API está sendo processada.**

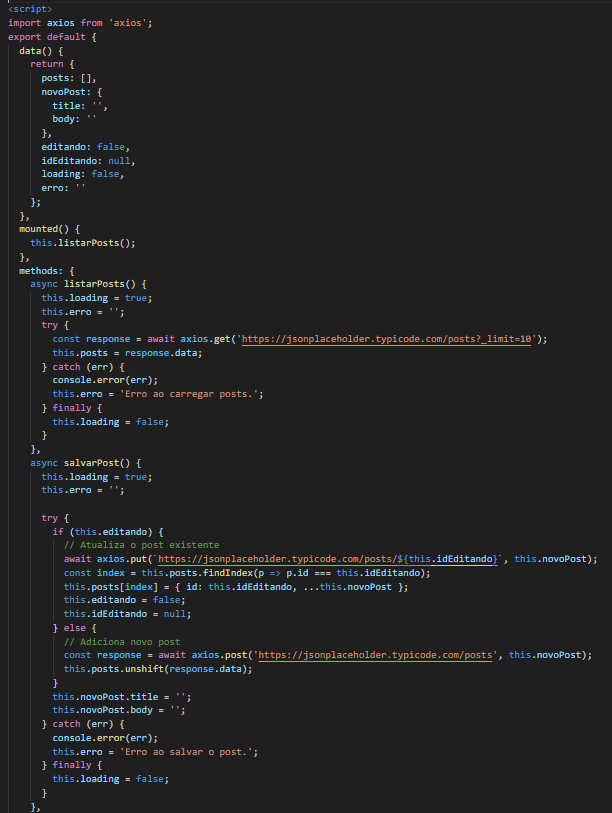


**8. Crie um mecanismo de tratamento de erros para exibir mensagens amigáveis ao usuário caso ocorra um erro na requisição à API.**



**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.**

