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

Uma **API RESTful** é uma interface que segue os princípios do REST, permitindo comunicação entre sistemas usando protocolos HTTP e recursos identificados por URLs.

Ela é amplamente usada porque é:

- Simples e padronizada.
- Independente de linguagem ou plataforma.
- Escalável e fácil de integrar com diferentes clientes (web, mobile, etc.).
- Facilita a construção de sistemas modulares e 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

Axios é uma biblioteca JavaScript para fazer requisições HTTP de forma simples e eficiente.

Vantagens sobre Fetch API:

- Suporte nativo a requisições **JSON** (envio e resposta).
- Melhor tratamento de erros, capturando falhas HTTP facilmente.
- Permite configurar interceptadores para modificar requisições e respostas.
- Suporta cancelamento de requisições e timeout.
- Compatível com navegadores antigos e Node.js.

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

- **GET:** recuperar dados do servidor.
- **POST:** criar novos recursos.
- **PUT:** atualizar recursos existentes completamente.
- **PATCH:** atualizar parcialmente um recurso.

- **DELETE:** remover recursos do servidor.

Esses métodos definem a ação que o cliente deseja realizar sobre os dados na API.

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

O **timeout** é o tempo máximo que o Axios espera por uma resposta da API antes de cancelar a requisição.

Ele é importante para evitar que a aplicação fique travada aguardando respostas demoradas ou que nunca chegam, melhorando a experiência do usuário e controlando recursos do sistema.

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

CORS (Cross-Origin Resource Sharing) é uma política de segurança dos navegadores que controla o acesso de recursos entre diferentes origens (domínios).

Se o front-end Vue.js estiver em um domínio e tentar acessar uma API em outro, o servidor precisa permitir esse acesso configurando cabeçalhos CORS. Caso contrário, a requisição será bloqueada pelo navegador, impedindo a comunicação entre front-end e back-end.

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

- **API pública:** pode ser acessada por qualquer usuário sem necessidade de login ou credenciais.
  Exemplo: API de dados meteorológicos públicos.
- **API autenticada:** requer credenciais (token, login) para garantir segurança e controle de acesso.
  Exemplo: API de redes sociais para postar ou ler dados privados do usuário.

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

Tratar erros é essencial para garantir que a aplicação lide corretamente com falhas, evitando travamentos e melhorando a experiência do usuário.

Principais tipos de erros:

- **Erros de rede:** falta de conexão ou tempo esgotado.
- **Erros HTTP:** respostas com status 4xx (cliente) ou 5xx (servidor).
- **Erros de dados:** respostas malformadas ou inesperadas.
- **Erros de autenticação/autorização:** acesso negado por falta de credenciais.

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 com a requisição ou resposta HTTP.

Ao fazer requisições a uma API, cabeçalhos podem ser usados para:

- Autenticação (ex: tokens).
- Definir o tipo de conteúdo (Content-Type).
- Controlar cache.
- Informar sobre idioma ou formato aceito.

Eles ajudam a API a entender e processar corretamente a requisição.

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

O **baseURL** define a URL base para todas as requisições feitas pela instância do Axios.

Isso facilita o desenvolvimento porque evita repetir a parte comum da URL em todas as chamadas, tornando o código mais limpo e fácil de manter, especialmente quando a API tem um endereço fixo.

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

- **.then/.catch** usa Promises para tratar resultados e erros de forma encadeada, com callbacks.
- **async/await** permite escrever código assíncrono de forma mais **síncrona e legível**, usando palavras-chave para esperar resultados.

async/await facilita o entendimento do fluxo e o tratamento de erros com try/catch, deixando o código mais limpo e próximo do estilo imperativo.