1. O que é o Node.js e por que ele é uma escolha popular para o desenvolvimento de back-end?

Node.js é uma plataforma que permite executar código JavaScript no servidor.

É popular no back-end por ser rápido, leve e assíncrono, ideal para aplicações em tempo real, APIs e sistemas com muitas conexões simultâneas. Além disso, permite usar a mesma linguagem (JavaScript) no front-end e back-end.

2. Explique o papel do Express.js no desenvolvimento de APIs em Node.js. Quais são suas principais vantagens?

O **Express.js** é um framework minimalista para Node.js usado na criação de **APIs e aplicações web**.
Ele simplifica o roteamento, o tratamento de requisições e respostas, e a integração com middlewares.

Principais vantagens:

- Sintaxe simples e rápida.
- Suporte a middlewares e rotas.
- Ampla comunidade e documentação.
- Flexível para projetos pequenos ou grandes.

3. O que é um ORM e qual a vantagem de utilizar o Sequelize para manipular bancos de dados no Node.js?

Um **ORM (Object-Relational Mapping)** é uma ferramenta que permite interagir com bancos de dados relacionais usando **objetos JavaScript**, sem precisar escrever SQL diretamente.

Vantagens do Sequelize no Node.js:

- Mapeia tabelas e colunas em modelos JS.
- Facilita **CRUD**, relacionamentos e validações.
- Suporta múltiplos bancos (MySQL, PostgreSQL, etc.).
- Melhora a legibilidade e a manutenção do código.

4. Como funciona a comunicação entre Vue.js e Node.js? Qual a importância do uso de Axios nesse processo?

A comunicação entre **Vue.js (front-end)** e **Node.js (back-end)** acontece por meio de **requisições HTTP** a uma API.

O **Axios** é uma biblioteca usada no Vue.js para fazer essas requisições de forma simples e eficiente.

Importância do Axios:

- Envia e recebe dados da API (GET, POST, PUT, DELETE).
- Trabalha com Promises.
- Permite interceptar requisições/respostas e tratar erros.
  Com ele, o Vue pode consumir dados do Node e atualizar a interface dinamicamente.

5. Explique o conceito de APIs RESTful e a importância dos métodos HTTP (GET, POST, PUT, DELETE) na construção dessas APIs.

APIs **RESTful** seguem princípios de arquitetura para comunicação entre cliente e servidor usando o protocolo HTTP.

Os métodos HTTP são fundamentais para indicar a ação a ser realizada:

- **GET:** buscar dados.
- **POST:** criar novos recursos.
- **PUT:** atualizar recursos existentes.
- **DELETE:** remover recursos.

6. O que é um middleware no Express.js e como ele pode ser utilizado para manipular requisições e respostas em uma API?

Middleware é uma função que processa as requisições e respostas antes de chegar às rotas finais.

Ele pode:

- Modificar dados da requisição ou resposta.
- Fazer autenticação, validação e logging.
- Controlar erros e permissões.

No Express, middlewares permitem adicionar funcionalidades e controlar o fluxo da aplicação de forma modular e reutilizável.

7. Quais são os benefícios de utilizar JWT (JSON Web Token) na autenticação de usuários em uma aplicação Node.js?

Os benefícios do JWT incluem:

- **Autenticação sem estado:** não precisa armazenar sessão no servidor.
- **Transporte seguro de informações:** dados codificados e assinados.
- **Escalabilidade:** fácil uso em arquiteturas distribuídas.
- **Flexibilidade:** pode conter permissões e dados do usuário.
- **Compatibilidade:** funciona bem com APIs RESTful e front-ends modernos.

8. Explique a diferença entre uma requisição síncrona e uma requisição assíncrona no Node.js. Como o uso de async/await facilita a programação assíncrona?

- **Requisição síncrona:** o código espera a operação terminar antes de continuar, bloqueando o fluxo.
- **Requisição assíncrona:** o código dispara a operação e continua executando, sem esperar a resposta imediata.

O async/await facilita a programação assíncrona tornando o código mais **limpo e parecido com código síncrono**, evitando o uso excessivo de callbacks e .then(), melhorando a legibilidade e tratamento de erros.

9. Quais são os principais desafios ao integrar um banco de dados MySQL a uma aplicação Node.js e como o Sequelize ajuda a resolver esses desafios?

Desafios:

- Escrever consultas SQL complexas manualmente.
- Gerenciar conexões e transações.
- Mapear dados relacionais para objetos JS.
- Manter consistência e evitar erros.

Como o Sequelize ajuda:

- Fornece uma camada ORM que abstrai o SQL.
- Facilita definição de modelos e relacionamentos.
- Gerencia conexões e transações automaticamente.
- Oferece métodos simples para CRUD e validações, reduzindo erros e aumentando produtividade.

10. Quais são as boas práticas para estruturar um projeto de back-end em Node.js com Express e Sequelize?

- **Separar pastas** por responsabilidade: routes/, controllers/, models/, middlewares/.
- **Usar controllers** para lógica de negócio, mantendo rotas limpas.
- **Definir modelos Sequelize** organizados e relacionamentos claros.
- **Configurar variáveis de ambiente** para conexões e segredos.
- **Tratar erros e validar dados** com middlewares.
- **Usar migrações e seeders** para versionar banco de dados.
- **Manter código modular e reutilizável** para facilitar manutenção e testes.