1. Explique a diferença entre callbacks, promises e async/await.

- **Callback**: função passada como argumento que é chamada após a conclusão de uma tarefa.
- **Promise**: objeto que representa um valor futuro, com .then() e .catch().
- **Async/await**: sintaxe moderna que usa Promises de forma mais legível.

2. O que é o Worker Threads no Node.js e qual sua utilidade?
Worker Threads permitem executar código JavaScript em **threads separadas**, útil para tarefas pesadas que não devem travar o event loop principal.

3. Como o Socket.io facilita a comunicação em tempo real?
Socket.io cria uma conexão WebSocket entre cliente e servidor, permitindo **envio e recebimento de dados em tempo real**, como em chats, jogos ou notificações.

4. Qual é o papel dos clusters em aplicações Node.js?
Clusters permitem rodar múltiplas instâncias da aplicação em **diferentes núcleos da CPU**, melhorando desempenho e aproveitamento do hardware.

5. Liste três vantagens do PM2 para o gerenciamento de processos.

1. Mantém a aplicação sempre ativa (auto-restart).
2. Permite monitoramento em tempo real.
3. Suporta logs unificados e balanceamento de carga com clusters.

6. O que é um child process e como ele é utilizado?
Child process é um processo filho criado a partir do Node.js para executar comandos do sistema ou scripts externos, útil para tarefas paralelas.

7. Explique o conceito de logging estruturado e sua importância.
Logging estruturado registra logs em formato padrão (como JSON), facilitando **análise, filtragem e integração com sistemas de monitoramento**.

8. Qual é a vantagem de usar o Winston para logs?
Winston é um gerenciador de logs flexível e poderoso, que permite **salvar logs em arquivos, console ou serviços externos**, com níveis e formatos personalizados.

9. Por que é importante escalar aplicações Node.js em ambientes de produção?
Para lidar com mais usuários, evitar travamentos e **garantir alta disponibilidade e desempenho**, principalmente sob alta carga.

10. Como a programação assíncrona melhora a performance de aplicações?
Ela permite que o Node.js **continue executando outras tarefas enquanto aguarda operações demoradas**, como leitura de arquivos ou acesso ao banco de dados.