4) Dado uma aplicação que possui apenas um CRUD para uma tabela de Produtos, responda:

4.1) Qual sua sugestão de modelagem de dados para a necessidade de associar Fornecedores aos Produtos cadastrados, bem como categorizar esses produtos.

Construir 03 tabelas: fornecedor, categoria, produto, e criar o relacionamento entre elas.

4.2) Quanto tempo você acha que você levaria para desenvolver essa funcionalidade acima?

Depende de quantos atributos cada tabela teria. Se for uma média de 06 atributos básicos para cada tabela, acredito que em 03 horas poderiam ser construídas estas tabelas, e testadas algumas relações entre elas.

4.3) Agora que os Produtos já possuem Fornecedores. Surgiu a necessidade de liberar acesso no sistema para que cada Fornecedor faça a gestão somente de seus Produtos. Quais as principais tarefas que você enxerga para desenvolver essa nova necessidade?

A criação de mais uma tabela (usuário), com no mínimo os seguintes atributos: ID, nome, senha, perfil, fornecedor.

5) Veja o código a seguir e responda as perguntas a seguir.

```
for(var i = 0; i <= 3; i++) {
  setTimeout(function() { console.log(i); }, 100);
}
```

A) O que será apresentado no console? Por quê?

O número 4, quatro vezes. Isto porque neste código está programado uma função de iteração, com 4 posições (índices) e é mostrado em tela a quantidade de índices (de 0 a 3) a ser percorridos, a cada laço de iteração, que neste código são 4 laços.

B) Altere o código de forma que os números 0, 1, 2 e 3 sejam apresentados no console. Justifique a alteração.

Para mostrar esses números, basta colocar o laço para mostrar o número do índice, que neste caso está representado pela variável i no código abaixo:

```
for (let i = 0; i <= 3; i++) {
  console.log(i);
}
```

6) Mostre na tela números de 1 a 100, mas para os números múltiplos de 3 mostre a palavra “Fizz”, para os números múltiplos de 5 mostre a palavra “Buzz” e para os números múltiplos de 3 e 5 a palavra “FizzBuzz”.

```
for (i = 1; i <= 100; i++) {
  if (i % 3 === 0 && i % 5 === 0) {
    console.log("fizzbuzz");
```

```
  } else if (i % 3 === 0) {
    console.log("fizz");
  } else if (i % 5 === 0) {
    console.log("buzz");
  } else if (i != (i % 3 === 0) && i != (i % 5 === 0)) {
    console.log(i);
  }
}
```

7) Troque os valores das variáveis x e y sem o auxílio de outra terceira variável.

x = 24

y = 99

```
let x = 24
let y = 99

x = 99
y = 24

console.log(x, y);
```

devops

---

8) Descreva a sequência de comandos para "matar" um processo zumbi no servidor.

Nem sempre será possível encerrar alguma atividade que não esteja respondendo, por meio do Gerenciador de Tarefas. Mas isso pode ser feito pelo prompt de comando (CMD), através dos comandos a seguir:

1. Abrir o CMD e digitar o comando tasklist:
   C:\tasklist.exe
2. Digitar enter. Vai aparecer uma lista de atividades em execução. Pela coluna ‘uso da memória’ poderá ser identificada qual atividade está consumindo mais memória e comprometendo o desempenho da máquina.
3. Após identificar qual atividade dever ser encerrada, verificar qual o número de identificação na coluna ‘identifi’ e anotar.
4. Ainda na linha de comando, digitar “taskkill/PID NUMERO_PROCESSO/F”. O NUMERO_PROCESSO é o número anotado no passo anterior.

9) Indique a linha de comando (shell) para listar somente os processos node rodando no server.

10) Configure o cron abaixo para rodar toda sexta feira, as 10n da manha e as 22:22h da noite.

```
0  10  *  *  5, 22  22  *  *  5/usr/local/foo-bar.sh
```

11) Crie um docker compose com os 2 serviços desenvolvidos no exercício 3.