Prova Técnica – Avaliação de Desenvolvedor Java Sênior

Instruções Gerais:
- Duração estimada: até 4 horas.
- Submeta a solução final em um repositório Git com instruções claras de execução e testes.
- A prova avalia: clareza de código, arquitetura, uso de boas práticas, conhecimento técnico e habilidade com ferra

1. Desenvolva um microsserviço Java usando Quarkus responsável pelo gerenciamento de cadastro de clientes e
Funcionalidades:
- Cadastro, edição e consulta de clientes.
- Criação de pedidos com múltiplos itens, com vínculo ao cliente.
- Salvar o pedido com status “CRIADO”.
- Publicar um evento “PedidoCriado” em um tópico Kafka com os dados essenciais do pedido.

Tecnologias obrigatórias:
- Java 17+
- Quarkus (JAX-RS, Hibernate Panache, Kafka)
- Banco de dados em memória (H2)
- Testes unitários com JUnit ou REST Assured
- Uso de DTOs com mapeamento claro

2. Implemente um segundo serviço para controle logístico.
Funcionalidades:
- Consumir os eventos “PedidoCriado”.
- Simular a separação de itens (com delay artificial).
- Atualizar o status para “EM_TRANSPORTE” chamando o serviço de pedidos via API.

3. Crie uma DAG do Apache Airflow (arquivo .py) que:
- Baixe dados de um endpoint HTTP (pode usar JSON placeholder ou outro público).
- Converta o conteúdo para CSV e salve localmente ou em bucket simulado.
- Envie o resultado para um tópico Kafka para ser processado por um consumidor.

4. Elabore uma estratégia de monitoramento e rastreamento distribuído para os serviços acima.
Indique:
- Ferramentas (Prometheus, Grafana, Loki, Jaeger, etc).
- Quais métricas e logs seriam capturados e onde.
- Como relacionar chamadas entre microsserviços.

5. Descreva como garantiria alta disponibilidade e resiliência nos serviços.
Inclua práticas como: retries, circuit breakers, timeouts, fallback e mensageria assíncrona.

6. Compartilhe um desafio técnico que você enfrentou na sua experiência, explicando a decisão tomada, trade-off

7. Análise de Código:
Analise e refatore o código abaixo seguindo princípios de SOLID, tratamento de exceções e boas práticas de orien

```
public class ClienteService {
public void cadastrar(Cliente cliente) {
if (cliente.getNome().isEmpty()) {
throw new RuntimeException("Nome inválido");
}
System.out.println("Cliente cadastrado: " + cliente.getNome());
}
}
```