# Desafio VR Desenvolvimento

1. Recebemos um código desenvolvido por terceiros de um sistema que possui alto volume de lógica de negócio e apresenta as seguintes características:

- O sistema recebe requisições REST, está dividido em camadas e possui classes de domínio;

- O controller recebe a requisição e está com toda lógica de negócio. Monta e repassa o domínio para a aplicação;

- A aplicação tem a responsabilidade de repassar o objeto pronto para o repositório;

- O repositório apenas persiste os objetos mapeados do hibernate através de spring data;

- O domínio apenas faz o mapeamento para o BD;

- Nenhum teste unitário foi escrito.

- O sistema está escrito em java para rodar como spring boot.

Apresente observações/problemas sobre essa solução.

Comente qual(is) a(s) sua(s) estratégia(s) para melhorar este sistema em termos de qualidade e manutenção. Justifique suas decisões.

1. Retirar a lógica de negócio dos controllers e passar para a camada de aplicação.
2. Faria os testes unitários das classes que fizessem alguma alteração relevante.
3. Faria testes de integração para verificar a consistência das informações persistidas.
4. Desacoplar o mapeamento do banco de dados do domínio, utilizando padrões como Data Transfer Objects (DTOs) para representar as informações que transitam entre as camadas.
5. Utilizar padrões de projeto adequados à arquitetura, como o padrão de Repositório para abstrair o acesso ao banco de dados e reduzir o acoplamento com o Hibernate e o Spring Data.
6. Documentar o código, fornecendo explicações claras sobre a lógica de negócio, a estrutura do sistema e quaisquer decisões importantes tomadas durante o desenvolvimento.
7. Descreva quais são as principais limitações ao se adotar servidores de aplicação em uma arquitetura orientada a microsserviços.

Overhead e Consumo de Recursos – os servidores de aplicação geralmente incluem uma ampla gama de recursos e funcionalidades, mas nem todos eles serão necessários para cada microsserviço. Isso pode levar a um consumo excessivo de recursos, como memória e CPU, uma vez que cada servidor de aplicação executará um conjunto completo de recursos, mesmo que nem todos sejam utilizados.

Escala Vertical em vez de Escala Horizontal – servidores de aplicação geralmente seguem uma abordagem de escala vertical, onde você aumenta os recursos de uma única máquina para lidar com mais carga. Isso pode limitar a escalabilidade da aplicação em um ambiente de microsserviços, onde o ideal é ter a capacidade de dimensionar horizontalmente, adicionando instâncias independentes de um serviço para lidar com o aumento da demanda.

Ponto único de falha – ao utilizar servidores de aplicação, você pode criar um ponto único de falha para todos os microsserviços hospedados nele. Se o servidor de aplicação enfrentar problemas ou falhar, todos os serviços hospedados nele serão afetados, o que vai contra o princípio de isolamento e resiliência da arquitetura de microsserviços.

1. Atualmente, diversas aplicações escritas em Java estão deixando de serem desenvolvidas para rodarem em servidores (JBoss, Tomcat), adotando ferramentas que disponibilizam um servidor embutido na própria ferramenta. Quais são os principais desafios ao se tomar uma decisão dessas? Justifique sua resposta.

Para o desenvolvedor é melhor. Mais fácil de implantar, configurar e tem uma menor sobrecarga de recursos.

Os desafios que precisam ser considerados são:

Flexibilidade tecnológica – ao utilizar um servidor embutido, você está adotando uma tecnologia específica que a ferramenta suporta.

Desempenho e escalabilidade – servidores embutidos são otimizados para ambientes de desenvolvimento e testes, mas podem não ser tão eficientes quanto servidores de aplicação dedicados em ambientes de produção com alto tráfego.

Monitoramento e gerenciamento – servidores embutidos podem oferecer menos opções de monitoramento e gerenciamento em comparação com servidores de aplicação independentes, o que pode dificultar a identificação e solução de problemas em tempo real.

1. Teste prático (em anexo)