# 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.

Uma das principais coisas para se facilitar a manutenção é tirar a lógica de negócio do controller e criar as classes de services de acordo com a necessidade.

A classe DAO e a classe Repository devem ser criadas e coexistirem, pois é de responsabilidade da DAO é com a infra (select, insert, update) e a resposabildiade do repository é de dar apoio ao domínio.

Entendo que a ideia de ter testes unitários é bastante trabalhoso e as vezes complexo quando se há necessidade de mock. Mas, se o dia a dia permitir é interessante a implementação dos testes para evitar futuros bugs, ou até mesmo implementar em alguns projetos o Test-driven development (TDD) que facilitam na hora de fazer os testes.

1. Descreva quais são as principais limitações ao se adotar servidores de aplicação em uma arquitetura orientada a microsserviços.

Alguns pontos a se pensar:

* Testar a aplicação se torna mais difícil – Claro que você pode fazer os testes unitários dentro de cada microserviço, mas você testar o conjunto todo é uma tarefa trabalhosa e muito complexa.
* Implementar mecanismos de comunicações – Alem de implementar uma mensageria, temos que tratar casos de indisponibilidade ou até lentindão nas respostas.
* A complexidade nos sistemas distribuídos é mais difícil de ser adotada nos times.
* Uma maior complexidade na hora de fazer a implantação quando se há muitos serviços.

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 empresas com casos de uso de microsserviços em que se apegam a pequenas soluções, o modelo de servidor embutido é ótimo porque acelera as coisas e torna os ciclos de desenvolvimento mais rápidos, pois não há ajuste de serviço nesse tipo de solução. Mas o modelo tradicional não vai morrer. Muitas empresas não usarão microsserviços, muitas continuarão usando uma única solução de médio porte e muitas precisarão continuar ajustando servidores e serviços. Para aqueles, este modelo embutido não funciona. Então, na verdade, teremos esses dois estilos que chegarão ao mercado e teremos que escolhê-los com base no contexto da solução.

1. Teste prático (em anexo)