# Desafio VR Desenvolvimento

November 10th 2022

Maurício de Araújo Del Rei

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. O controller recebe a requisição e está com toda lógica de negócio. Monta e repassa o domínio para a aplicação - Criar um pacote service e para cada regra de negócio, criar uma classe de serviço, deixando cada serviço somente com uma responsabilidade.
2. A aplicação tem a responsabilidade de repassar o objeto pronto para o repositório - Criar um pacote DTO para transporte de dados.
3. Nenhum teste unitário foi escrito - [o software é colocado em produção mas a taxa de defeitos é tão alta que ele acaba não sendo utilizado, o software é colocado em produção com sucesso, porém após algum tempo o custo de se fazer modificações ou a taxa de defeitos aumenta tanto que o sistema precisa ser substituído]. Posto isto, o uso do TDD é fundamental neste caso.
4. O sistema está escrito em java para rodar como spring boot - Para a criação do projeto em Java com Gradle ou Maven, indica-se utlizar o <https://start.spring.io/> , pois as bibliotecas necessárias para rodar o projeto já são baixadas automaticamente, bem como o tomcat embutido(embedded server).
5. Descreva quais são as principais limitações ao se adotar servidores de aplicação em uma arquitetura orientada a microsserviços - A implantação de um App Server para cada um tipo de microsserviço adiciona uma sobrecarga considerável. Um número maior de instâncias são necessárias comparado ao modelo de configuração tradicional. Além disso, os microsserviços se concentram em infraestrutura especializada. Microsserviços aumentam o número de componentes independentemente deployados. Isso significa que uma infraestrutura complexa deve ser evitada.
6. Atualmente, diversas aplicações escritas em Java estão deixando de serem desenvolvidas para rodar 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. - Partindo do pressuposto que os App Servers apenas causam mais complexidade, não há bons motivos para usá-los. Usando embedded servers, podemos empacotar nossa app em arquivos JAR que contém uma classe principal e, portanto, podem ser iniciados a partir da linha de comando. O arquivo JAR contém toda a infra-estrutura necessária, tanto as bibliotecas, como um servidor HTTP incorporado para uma aplicação web. O deploy pode ser feito com ferramentas de automação como Puppet, Chef ou Ansible. Elas são bem fáceis de configurar. É difícil encontrar razões técnicas convincentes para justificar o uso de App Servers hoje em dia. O suporte para deploy e monitoramento pode ser feito com ferramentas genéricas às quais as operações já estão acostumadas. As bibliotecas fornecem os recursos que os desenvolvedores precisam. Sem um App Server, o desenvolvimento de aplicativos é mais fácil - e também o deploy, teste e debug.
7. Teste prático (em anexo) - Ok.

Testes:

POST <http://localhost:8080/cartoes>

{

"numeroCartao": "6549873025634501",

"nomeDoTitular": "VR Alimentação",

"codigoDeSeguranca": 123,

"senha": "3456",

"saldo":"500.00"

}

GET [http://localhost:8080/cartoes/numeroCartao/{numeroCartao](http://localhost:8080/cartoes/numeroCartao/%7BnumeroCartao)}

{

"saldo": Integer

}

POST <http://localhost:8080/transacoes>

{

"numeroCartao": "6549873025634501",

"senha": "3456",

"valor":"300.00"

}

Obs: Após realizar uma transação, ao chamar o endpoint [http://localhost:8080/cartoes/numeroCartao/{numeroCartao](http://localhost:8080/cartoes/numeroCartao/%7BnumeroCartao)} , o saldo retorna com o valor alterado.