# 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 – Controller:

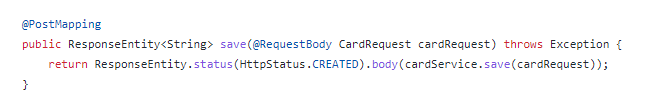
Controller jamais deve conter lógica de negócio, isso deve ficar em uma classe de service, o controller apenas a camada de API, a porta de entrada e saída.

2 – Teste uniátios:

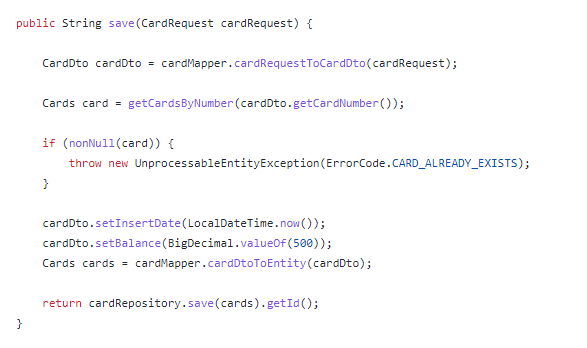
Sistema sem testes unitários é extremamente vuneravel a qualquer mudança em código, um simples IF pode quebrar uma lógica ou validação trazendo grandes risco a aplicação. Todo sistema que se preze tem que estar completamente coberto por testes eficazes.

Exemplo de uma arquitetura correta:

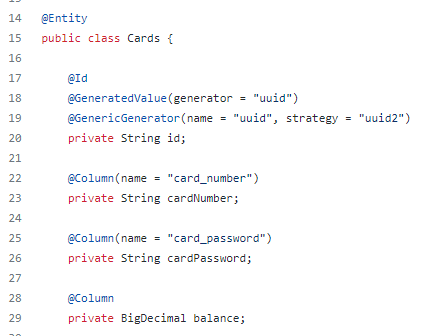
Controller apenas recebe a requisição e devolve a resposta:



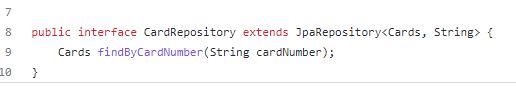
Service: implementa toda a lógica de negócio e devolve o resultado ao controller



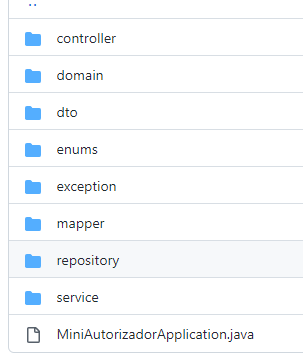
Classes de domínio apenas mapeiam as tabelas, sem nenhuma regra de negócios



Repositories: interfaces responsáveis pelo CRUD da aplicação



Exemplo de uma estrutura saldável



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

Uma das principais vantagens é da independência dos serviços entre si, de sua flexibilidade e escalabilidade. Como cada serviço é um componente separado, é possível expandir uma única função ou serviço, com um servidor de aplicação isso fica limitado ao todo

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.

Existe muitos motivos e bons motivos para se usar micros serviços, mas os principais desafios são:

- Custo do projeto, a quantidade de horas do desenvolvedor para essa migração.

- Retorno do investimento, mesmo usando jboss ou tomcat, um sistema pode ser robusto, muitas vezes a migração para micros serviços tendem a ser subestimado .

- Custo com novas ferramentas e também processos ainda mais levando em conta se não tiver uma infraestrutura madura para micros serviços em produção, deverá ser considerado o custo da adoção de novas tecnologias

- Se o sistema ‘legado’ receber contínuas mudanças, como o novo em micro serviços irá acompanhar as mudanças? Esse é um ponto muito crítico ao se fazer uma migração dessas

Os sistemas legados sempre farão parte de nossas vidas, o monolito é uma daquelas dívidas técnicas que conhecemos, pode ou não ser um problema urgente, mas caso seja, precisamos nos esforçar para superá-lo gradualmente e ter a adesão de todos os envolvidos.

1. Teste prático (em anexo)