TDD- Desenvolvimento Guiado por Testes

Qual o propósito? Imagine um cenário onde temos
1 - Requisitos incompletos
2 - Potenciais dívidas técnica (não conseguimos ainda - por falta de informação - implementar todos os cenários)
3 - Ainda não temos a implementação completa porque dependemos de outra equipe

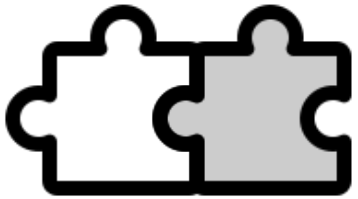
Como proceder?

Podemos simular um determinado componente sem ter sua implementação concluída

MOCK! (dubê, simulação)

Cenário 1 - Não tenho a implementação do SERVICE nem acesso ao Repository

Posso então, testar uma "simulação" do service?



Teste Mock

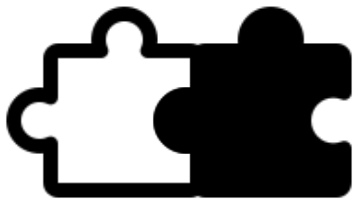
Mock

Eu não defino nada de lógica, eu apenas defino que, dada uma determinada entrada, eu gero uma determinada saída

When(método).then(resultado)

Cenário 2 - Agora tenho a implementação

Se eu substituir o Mock pela implementação oficial do service, os testes precisam manter-se consistentes (ou seja, aquilo que falhava com o Mock precisa falhar com a implementação e aquilo que passava no teste como Mock precisa continuar passando no teste com a implementação)



Teste Implementação Oficial

Cenário 1

Defini meus scripts de testes
Declarei a interface
Ao invés de fazer a injeção de dependência da impelmentação, eu a anotei como um "Bean Mockado" (um objeto simulado)
Escrevi o comportamento da chamada dos métodos deste objeto (com o Mockito)

Rodei os testes e deveria passar nos cenários que eu defini

Cenário 2

Defini meus scripts de testes
Declarei a interface
Ao invés de fazer a injeção de dependência da impelmentação, eu a anotei como um "Bean Mockado" (um objeto simulado)
Escrevi o comportamento da chamada dos métodos deste objeto (com o Mockito)

Rodei os testes e deveria passar nos cenários que eu defini

Testes de Integração

Em geral quando criamos nossos controllers, eles seguem +/- o seguinte caminho (numa arquitetura multicamadas)

controller <--> service <--> repository

Colmo costumamos testar os Controllers?

1. Subimos a API
2. Usamos uma ferramenta externa (Postman, Insomnia, Thunder Client, CURL, etc)
3. Montamos nossas collections de requisições e testamos conferindo os resultados visualmente

É possível automatizar isso? SIM

Usando a Lib MockMVC

Alguns cuidados

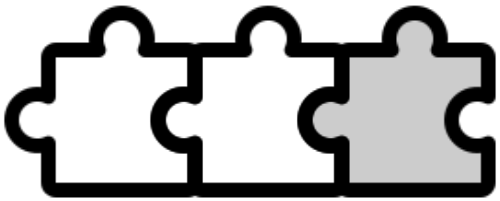
1. Anotar sua classe de testes como @AutoConfigureMockMVC
2. Injetar o componente MockMVC
3. Usar os recursos dele da seguinte maneira

```
mvc.perform(MockMvcRequestBuilders.get("/produtos/-1"))
      .andExpect(MockMvcResultMatchers.status().isBadRequest());
```

O Objeto `MockMvcRequestBuilders` oferece a chamada do método HTTP
O Objeto `MockMvcResultMatchers` oferece a conferência do retorno da requisição

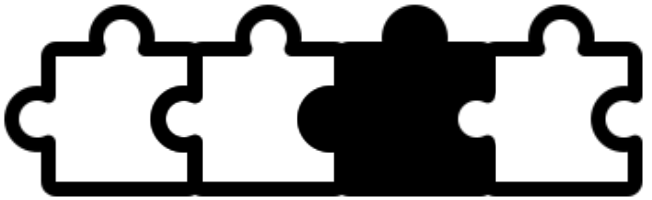
Posso Mockar o serviço acessado pelo Controller? SIM

Como? Da mesma forma que usamos nos cenários 1 e 2 anteriormente



Cenario 1
Mock do serviço sem acessar repositório

Teste **Controller** **Mock**



Teste **Controller** Repository

Implementação Oficial

Cenario 2
Uso da implementação oficial acessando o repositório

Vantagens de automatizar testes de integração

Você não precisa de várias ferramentas (tudo é feito via código)
você acaba tendo um "esforço inicial" de criar vários roteiros de testes
Você acaba documentando (se você usar uma boa prática de nomenclatura de métodos e clean code) todos os requisitos que foram testados na sua aplicação