
HOME
QUALITY
ASSURANCE

11

Estratégia de Testes

Prof. Jonathan Rodrigo da Silva Santos
Prof. Marco Túlio Jeunon

Resumo

Nesta aula iremos abordar as principais estratégias de testes, bem como pirâmide de testes, e mocks.

11.1 Pirâmide de Testes

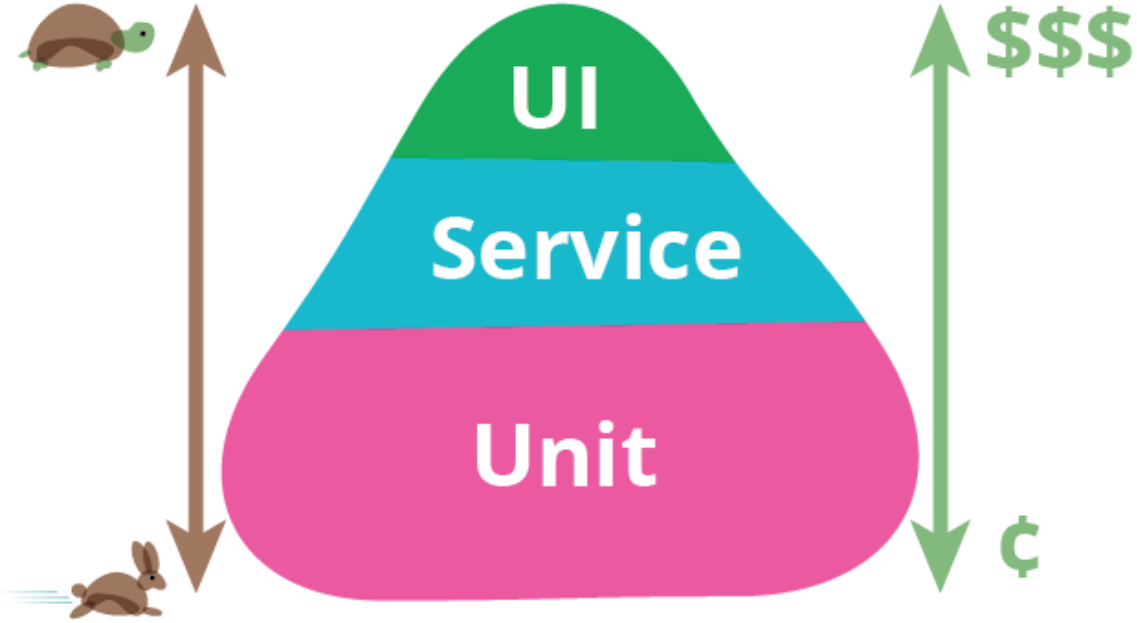
A "Pirâmide de Testes" é uma metáfora que nos diz para agrupar testes de software em diferentes granularidades é também uma forma gráfica de demonstrar de forma simplificada os tipos e níveis de testes, complexidade e velocidade de implementação dos testes a serem construídos e suas respectivas manutenções e execuções.

Ela é apresentada no formato de uma pirâmide pois provoca um direcionamento em relação a quantidade de testes a serem produzidos em cada um de seus níveis, em conjunto com o custo e complexidade de cada tipo de teste.

O assunto tomou grande proporção e se pesquisarmos chegaremos em pirâmides com várias divisões porém vamos tratar da divisão mais comum e tradicional, que é representada por 3 níveis, sendo eles:

- Base: Testes de Unidade
- Meio: Testes de Integração (Service)
- Topo: Testes Ponta a Ponta (UI, E2E ou Testes de Interface)

Figura 11.1. Pirâmide de Testes



Fonte: FOWLER, 2012.

Os níveis de teste geralmente são identificados por seus objetivos específicos e seu objeto de teste. Abaixo vamos ver de forma detalhada cada um dos níveis de forma mais profunda.

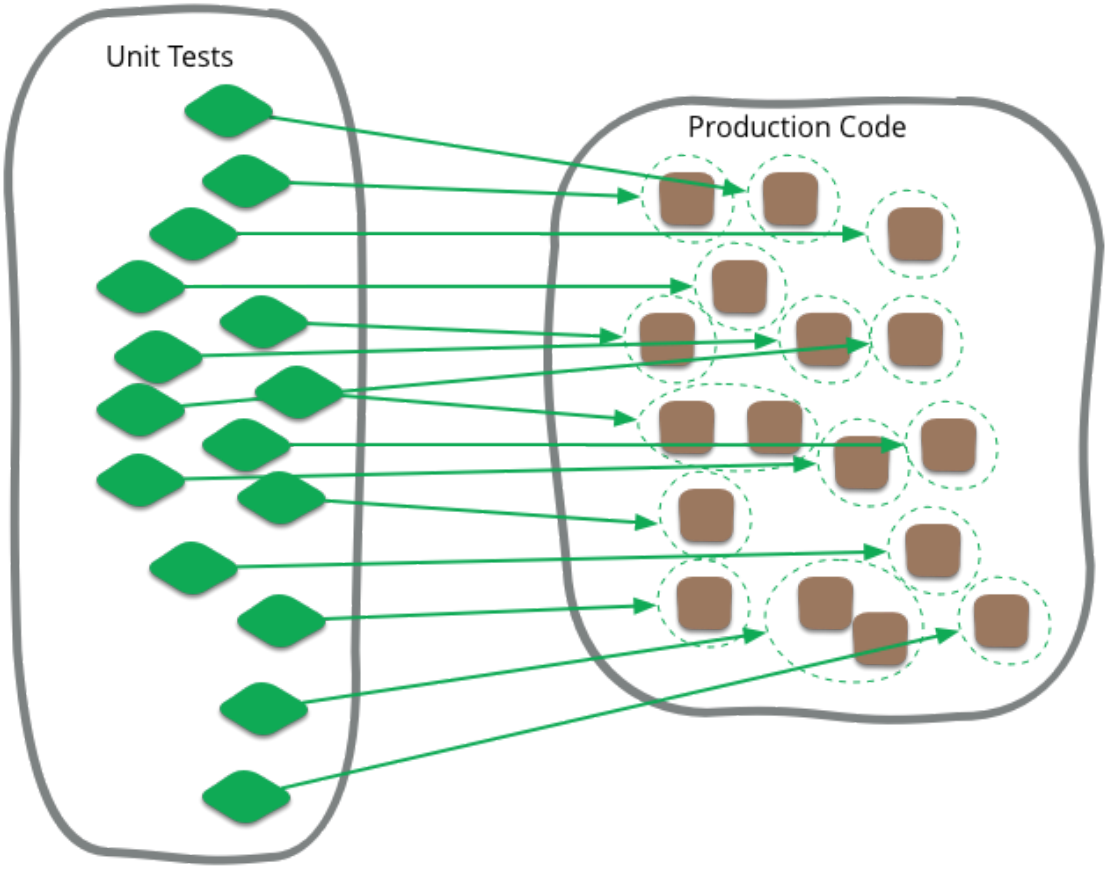
11.1.1 - Testes de Unidade

Os testes de unidade, ou como o mercado chama "testes unitários", são os testes realizados na menor parte testável de uma aplicação, independentemente de seu relacionamento com outras classes ou métodos do código. Em um contexto de OO - Orientação a Objetos , por exemplo, podemos classificar uma unidade como um método público disponível em uma determinada classe.

Por se tratar de um teste do menor pedaço possível do código de forma isolada, os testes unitários tem por característica ser pequenos, simples e de rápida criação e execução. Essa característica possibilita que quando um CT (Caso de teste) falhar, seja possível saber com exatidão a origem da mesma.

Como estratégia, para viabilizar a construção dos testes unitários de forma totalmente independente, os times utilizam-se de Mocks, pois eles imitam o comportamento de objetos reais, assim isolando a parte a ser testada e não tendo a necessidade de bater quente nos objetos reais.

Figura 11.2. Testes de Unidade



Fonte: FOWLER, 2012.

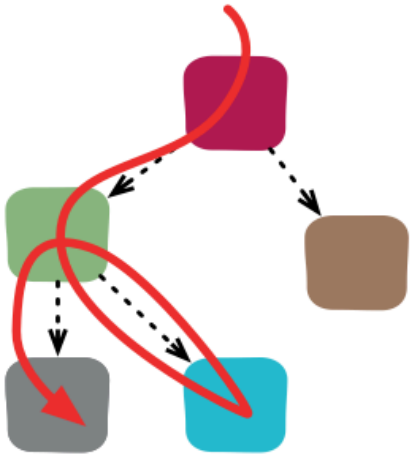
11.1.2 - Testes de Integração

Esses testes têm como principal objetivo exercitar com teste um conjunto de unidades que interagem.

Realizar teste de integração é muito importante, pois às vezes os testes unitários não são suficientes, pois podemos ter duas unidades que foram testadas de forma unitária funcionando conforme esperado, porém ao interagir-se, podem não apresentar o resultado esperado.

Alguns casos comuns de cobertura de testes de integração são testes realizados na comunicação com o banco de dados, comunicação de interfaces, APIs, micro-serviços.

Figura 11.3. Testes de Integração



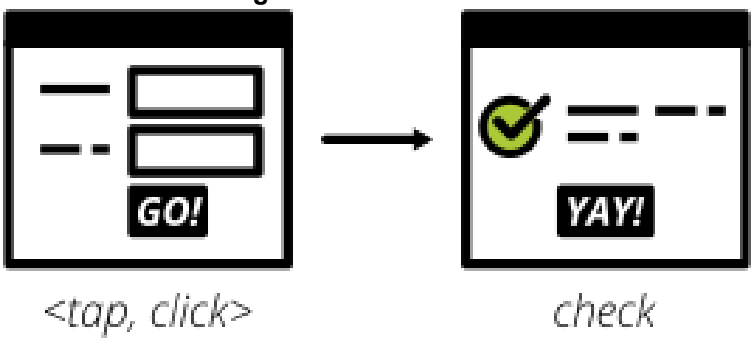
Fonte: FOWLER, 2012.

11.1.3 - Testes de UI

Os testes de UI têm como objetivo principal simular o comportamento de um usuário final na aplicação.

São testes que simulam o ambiente mais próximo do real, ou seja, abre o navegador, navegar nas funcionalidades do site ou sistema, clicar nos botões, popular campos e formulários e, ao fim, validar se está conforme ao comportamento esperado para a funcionalidade.

Figura 2.3. Testes de UI

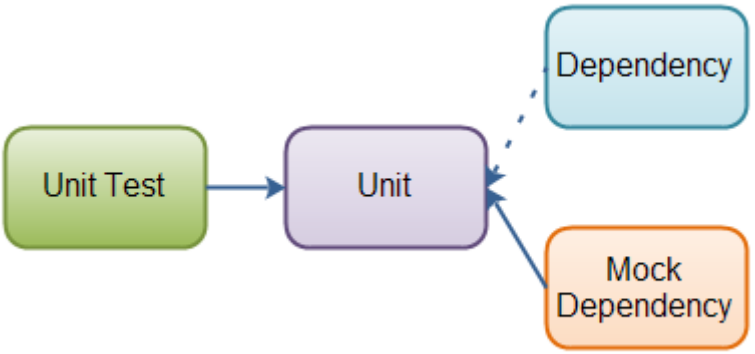


Fonte: FOWLER, 2012.

2.2 Mock

Ao planejar e escrever testes, muitas vezes necessitamos simular partes de nossa aplicação para tornar as execuções dos testes possíveis e os resultados reproduzíveis e alcançáveis. E a utilização de Mocks caem como uma luva neste contexto, pois eles são imitações ou unidades falsas que simulam o comportamento de unidades reais.

Figura 2.4 Esquema de Mock



Fonte: FOWLER, 2012.

Referências

BARTIE, A. **Garantia da Qualidade de Software.** Rio de Janeiro: Elsevier, 2002. ISBN: 978-85-352-1124-5.

DEMING, W. Edwards. **Qualidade:** a revolução da administração. Marques Saraiva, 1990.

FOWLER, Martin. **TestPyramid.** MartinFowler, 01 mai. 2012. Disponível em: <https://martinfowler.com/bliki/TestPyramid.html>. Acessado em: 21 mar. 2022.

GARVIN, D. **Competing on the eight dimensions of quality.** Harv. Bus. Rev. 1987, November/December, p. 101-109.

PALADINI, E. P. **Gestão da Qualidade:** teoria e prática. 2. ed. São Paulo: Atlas, 2004.

PEZZE, M.; YOUNG, M. **Teste e Análise de Software:** processos, princípios e técnicas. Porto Alegre: Bookman, 2008. ISBN: 978-85-778-0262-3.