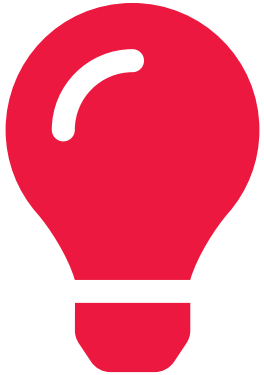
Introdução aos testes unitários e testes de integração

Certified Developer
The Ultimate Tech Degree

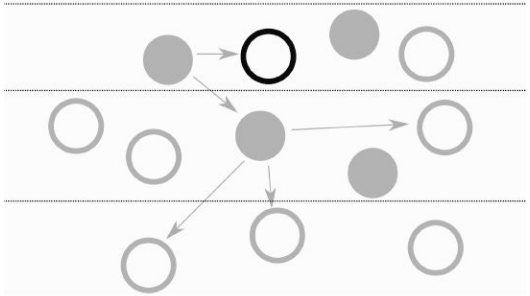
DigitalHouse >
Coding School

Vamos saber quais são os testes que devemos fazer para que nosso software responda como queremos.

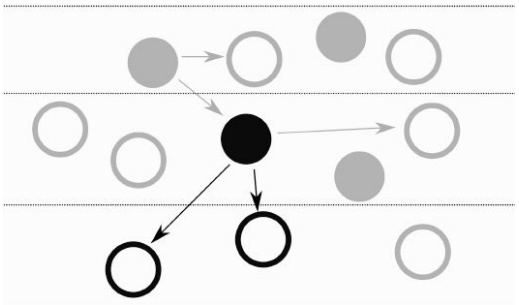


Qual a diferença entre eles?

Testes unitários



Testes de integração

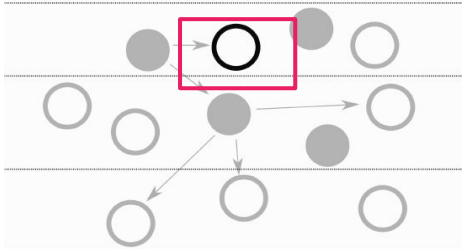


Testes unitários

Os testes unitários têm como objetivo pegar uma pequena parte do software, **isolando-a** do resto do código, para determinar se ela se comporta / funciona conforme o esperado.

Cada unidade é testada separadamente antes de ser integrada aos componentes para testar as interfaces entre as unidades.

Deve-se observar que qualquer dependência do módulo em teste deve ser substituída por um mock ou um stub, para limitar o teste especificamente a essa unidade de código.



Testes unitários

Para realizar um correto teste unitário, um processo conhecido como **3A** deve ser seguido:

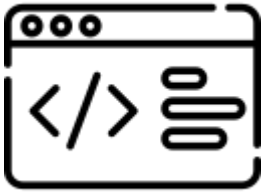
***A**rrange* (Preparar o teste) → Esta etapa define os requisitos que o código deve atender.

***A**ct* (Rodar o teste) → Aqui é executado o teste que dará origem aos resultados que devemos analisar.

***A**ssert* (Verificar as asserções) → Se comprovam se os resultados obtidos são os esperados. Nesse caso, ele é validado e continuado. Caso contrário, o erro é corrigido até desaparecer.

Vantagens dos Testes Unitários

Vejamos alguns benefícios de utilizar este tipo de teste:



Facilitar as mudanças de código

Ao detectar o erro rapidamente, é mais fácil alterá-lo e testá-lo novamente.

Fornecem documentação

Ajudam a entender o que o código faz e qual era a intenção ao desenvolvê-lo.

Encontrar bugs

Testando componentes individuais antes da integração. Isso garante que eles não afetem outra parte do código.

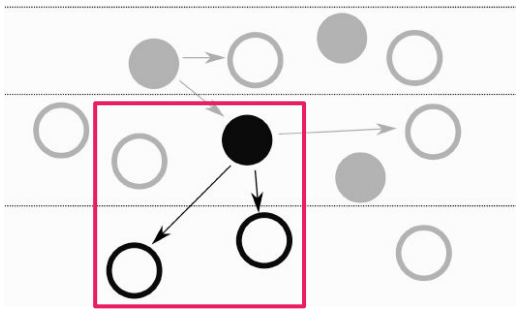
Melhoram o design e a qualidade do código

Convidam o desenvolvedor a pensar sobre o design antes de escrevê-lo (Test Driven Development - TDD).

Teste de Integração

As unidades individuais são **integradas** para formar componentes maiores, por exemplo, duas unidades que já foram testadas são combinadas em um componente integrado e a interface entre elas é testada. Isso nos permite cobrir uma área maior do código, sobre a qual às vezes não temos controle.

Portanto, podemos concluir que este tipo de teste visa **validar a interação** entre os módulos do software.




DigitalHouse>
Coding School