Tests unitarios con y sin Mock- Tests de Integración

 Repaso módulo Testing  
// Conceptos básicos

**¿Qué testeamos?**

Debemos testear el comportamiento de nuestro sistema. Para eso nos centramos en los requerimientos del negocio.

Podemos testear las siguientes capas:

* **Controller**: Validamos parámetros de entrada, y testeamos como se los pasamos al service.
* **Service**: La capa de servicio es el lugar donde seguramente tengamos más casos a testear, ya que allí creamos los métodos que hacen a nuestro negocio.
* **Repository**: La testeamos si nosotros hicimos su lógica, cuando usamos librerías que gestionan mi capa de Repo no necesito testearlo (si tenemos una base de datos no testeamos).

importante: Para que un componente sea testeable debe cumplir con el principio SOLID.

**Test unitarios**

// A tener en cuenta

Los test unitarios nos ayudan a ver si cada unidad del sistema que programe funciona como espero.

Deben cumplir con el principio FIRST.

**Test de integración**

// A tener en cuenta

Los test de integración me permiten ver cómo interactúan los módulos en el software que programé. A partir de un test de integración puedo ver si las unidades dependientes entre sí funcionan correctamente.

**¿Qué es un Mock?**

// A tener en cuenta

Los mocks me permiten simular el comportamiento de los componentes para poder realizar pruebas de forma aislada.

Lo usamos cuando necesitamos verificar los resultados y las interacciones de mis módulos.

**¿Cómo testeamos con Mocks?**

* Elijo la capa a testear
* Mockeo las dependencias de las clases con @Mock .
* Declaro una instancia de mi dependencia a testear (con @injectMock si uso @Autowired o por contructor)
* Mockeo las dependencias de las clases con @Mock para testear los métodos públicos (los métodos que estan expuestos)
* Dentro de cada método debemos cubrir todos los casos bordes (escenarios) posibles para lograr una cobertura de calidad y que mi red de testing sea lo más granular posible.
* Preparo mi setUp, debo usar *MockitoAnnotations.initMocks(this)*
* instanciamos el setUp necesario.(arrange)
* Definimos el comportamiento del mock con *when(methodCall).thenReturn(result)*
* llamamos al método que vamos a testear (act)
* checkeo los resultados esperados con los recibidos (assert : expected vs current)

// Consejos

No es necesario testear cosas que hayamos validado, pero podríamos testear los throws que catcheamos.

Usar Verify de Mockito para comprobar que un método ha sido llamado.