Introducción a test unitarios y test de integración

DigitalHouse >
Coding School

Certified Tech Developer
The Ultimate Degree

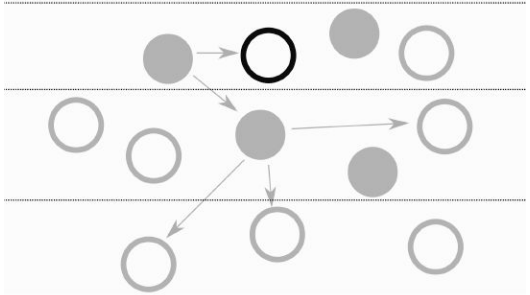
“

Conozcamos cuáles son las pruebas que debemos hacer para que nuestro software responda tal como queremos.

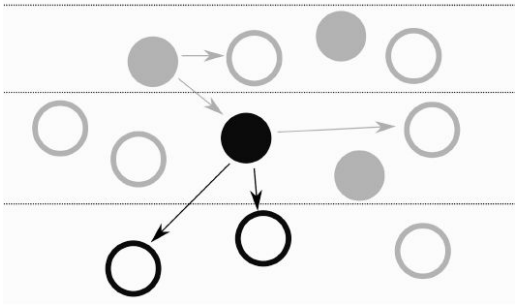
”

¿En qué se diferencian?

Test unitarios



Test de integración

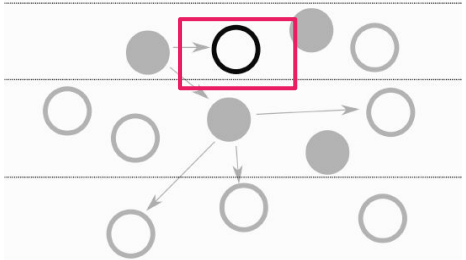


Test unitarios

Los test o pruebas unitarias tiene por objetivo tomar una pequeña parte de software, **aislandola** del resto del código, para determinar si se comporta/funciona tal como esperamos.

Cada unidad se prueba por separado antes de ser integrada en los componentes para probar las interfaces entre las unidades.

Cabe aclarar que, cualquier dependencia del módulo bajo prueba debe sustituirse por un mock o un stub, para acotar la prueba específicamente a esa unidad de código.



Test unitarios

Para llevar a cabo un correcto test unitario se debe seguir un proceso conocido como **3A:**

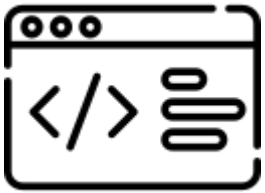
***A**rrange* (organizar) → En este paso se definen los requisitos que debe cumplir el código.

***A**ct* (actuar) → Aquí se ejecuta el test que dará lugar a los resultados que debemos analizar.

***A**ssert* (afirmar) → Se comprueban si los resultados obtenidos son los esperados. Si es así, se valida y se continúa. Caso contrario, se corrige el error hasta que desaparezca.

Ventajas de los test unitarios

Veamos algunos beneficios de utilizar este tipo de test:



Facilitar los cambios en el código

Al detectar el error rápidamente es más fácil cambiarlo y volver a probar.

Proveen documentación

Ayudan a comprender qué hace el código y cuál fue la intención al desarrollarlo.

Encontrar bugs

Probando componentes individuales antes de la integración. Esto genera que no impacten en otra parte del código.

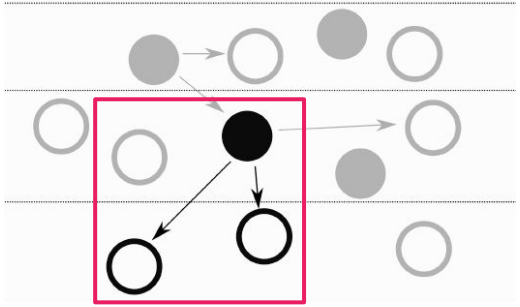
Mejoran el diseño y la calidad del código

Invitan al desarrollador a pensar en el diseño antes de escribirlo (Test Driven Development - TDD).

Test de integración

Las unidades individuales se **integran** para formar componentes más grandes, por ejemplo, dos unidades que ya fueron probadas se combinan en un componente integrado y se prueba la interfaz entre ellas. Esto nos permite cubrir un área mayor de código, del que a veces no tenemos control.

Por lo tanto, podemos concluir que este tipo de test tiene por objetivo **validar la interacción** entre los módulos de software.




DigitalHouse>
Coding School