

Capítulo 17

Pruebas unitarias

Objetivo

Al finalizar el capítulo, el alumno:

- Implementar pruebas unitarias al proyecto creado.

Temas

1. ¿Por qué las pruebas unitarias?
2. Desafíos con las pruebas unitarias
3. ¿Cómo se crea un unit test?
4. ¿Qué es un unit test?
5. Ventajas de los unit test


1. ¿Por qué las pruebas unitarias?

¿Por qué las pruebas unitarias?

- “Inicia con el final en mente.”
-- Stephen Covey (The 7 Habits of Highly Effective People)
- Conjunto de herramientas útiles para ayudar a crear una nueva base para el producto que es altamente estable, cambiable y utilizable.
- Algo que podemos vender con confianza y todavía dormir por la noche.

17 - 4

Copyright © Todos los Derechos Reservados - Cibertec Perú SAC.



Una prueba unitaria es una forma de comprobar el correcto funcionamiento de una unidad de código. Por ejemplo, en diseño estructurado o en diseño funcional una función o un procedimiento, en diseño orientado a objetos una clase. Esto sirve para asegurar que cada unidad funcione correctamente y eficientemente por separado. Además de verificar que el código hace lo que tiene que hacer, comprobamos que sea correcto el nombre, los nombres y tipos de los parámetros, el tipo de lo que se devuelve, que si el estado inicial es válido entonces el estado final es válido

La idea es escribir casos de prueba para cada función no trivial o método en el módulo, de forma que cada caso sea independiente del resto. Luego, con las pruebas de integración, se podrá asegurar el correcto funcionamiento del sistema o subsistema en cuestión.

2. Desafíos con las pruebas unitarias

Desafíos con las pruebas unitarias

- ¡No tenemos tiempo para hacer esto!
- No hará una diferencia significativa.
- ¡Mi código no necesita pruebas!
- Sólo haz que funcione ahora!
- ¿Cuántas pruebas de unidad necesitamos?
- ¿Qué conjunto de herramientas usar?
- ¿Hasta dónde llegar? (Por ejemplo, objetos simulados, CI)
- Plazos vs mejora de procesos
- ¿Podemos reducir nuestro personal de pruebas si hacemos la prueba de unidad?

17 - 5

Copyright © Todos los Derechos Reservados - Cibertec Perú S.A.C.



Muchas veces uno de los problemas con los que se tienen que lidiar al momento de querer implementar pruebas unitarias en el desarrollo de una aplicación empresarial, es el poco conocimiento de nuestros desarrolladores en la creación de los mismos.

Así mismo, tenemos que indicar que el pensar en pruebas unitarias demanda un tiempo adicional, lo cual para muchos proyectos no es viable, pero eso cambia cuando es visible la utilidad de estas pruebas, que permiten evitar inconvenientes posteriores y la creación de futuros errores en la aplicación.

3. ¿Qué es un unit test?

¿Qué es un unit test?

Una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado.



La prueba unitaria es un proceso de desarrollo de software en el que las partes más pequeñas comprobables de una aplicación, denominadas unidades, se examinan de forma individual e independiente para su correcto funcionamiento. La prueba de la unidad se puede hacer manualmente, pero se automatiza a menudo.

Las pruebas unitarias son un componente del desarrollo impulsado por pruebas (TDD), una metodología pragmática que adopta un enfoque meticuloso para construir un producto mediante pruebas y revisiones continuas. El desarrollo basado en pruebas requiere que los desarrolladores escriban primero las pruebas de unidades fallidas. Luego, escriben código y refactorizan la aplicación hasta que pase la prueba. TDD típicamente resulta en una base de código explícita y predecible.

Las pruebas unitarias implican sólo aquellas características que son vitales para el rendimiento de la unidad bajo prueba. Esto anima a los desarrolladores a modificar el código fuente sin preocupaciones inmediatas acerca de cómo dichos cambios pueden afectar el funcionamiento de otras unidades o del programa en su conjunto. Una vez que se ha comprobado que todas las unidades de un programa funcionan de la manera más eficiente y libre de errores posibles, los componentes más grandes del programa pueden evaluarse mediante pruebas de integración.

Las pruebas unitarias tienen una curva de aprendizaje muy pronunciada. El equipo de desarrollo necesita aprender lo que es la prueba de unidad, cómo realizar la prueba de unidad, qué prueba de unidad y cómo usar herramientas de software automatizadas para facilitar el proceso en forma continua. El gran beneficio de las pruebas unitarias es que cuanto antes se identifique un problema, menos errores compuestos se producen. Un error compuesto es uno que no parece romper nada al principio, pero eventualmente entra en conflicto con algo en la línea y resulta en un problema.

4. Ventajas de los unit test

Ventajas de los unit test

- Fomentan el cambio
- Simplifica la integración
- Documenta el código
- Separación de la interfaz y la implementación
- Los errores están más acotados y son más fáciles de localizar



Ventajas

El objetivo de las pruebas unitarias es aislar cada parte del programa y mostrar que las partes individuales son correctas. Proporcionan un contrato escrito que el trozo de código debe satisfacer. Estas pruebas aisladas proporcionan cinco ventajas básicas:

Fomentan el cambio

Las pruebas unitarias facilitan que el programador cambie el código para mejorar su estructura (lo que se ha dado en llamar refactorización), puesto que permiten hacer pruebas sobre los cambios y así asegurarse de que los nuevos cambios no han introducido errores.

Simplifica la integración

Puesto que permiten llegar a la fase de integración con un grado alto de seguridad de que el código está funcionando correctamente. De esta manera se facilitan las pruebas de integración.

Documenta el código

Las propias pruebas son documentación del código puesto que ahí se puede ver cómo utilizarlo.

Separación de la interfaz y la implementación

Dado que la única interacción entre los casos de prueba y las unidades bajo prueba son las interfaces de estas últimas, se puede cambiar cualquiera de los dos sin afectar al otro, a veces usando objetos mock (mock object) para simular el comportamiento de objetos complejos.

Los errores están más acotados y son más fáciles de localizar

Dado que tenemos pruebas unitarias que pueden desenmascararlos.