**Pruebas Unitarias**

**Silvana Patricia Zuluaga García**

**Rionegro-Antioquia**

**15 de Abril de 2024**

**ADSO 2697576**

**SENA**

**1. ¿Qué son las pruebas unitarias?**

Las pruebas unitarias son el proceso en el que se prueba la unidad funcional de código más pequeña, aislando una parte del código y comprobando que funcione a la perfección. Son pequeños tests que validan el comportamiento de un objeto y la lógica.

**2. ¿Cuál es el propósito de las pruebas unitarias?**

El propósito de las pruebas unitarias está en ayudar a garantizar la calidad del código para el buen desarrollo de un Software.

**3. ¿Qué ventajas tienen las pruebas unitarias?**

**→** **Proporciona un trabajo ágil**: Como procedimiento ágil que es, te permite poder detectar los errores a tiempo, de forma que puedas reescribir el código o corregir errores sin necesidad de tener que volver al principio y rehacer el trabajo. Puesto que las pequeñas se van haciendo periódicamente y en pequeños packs. Disminuyendo el tiempo y el coste.

**→** **Calidad del código:** Al realizar pruebas continuamente y detectar los errores, cuando el código está terminado, es un código limpio y de calidad.

**→** **Detectar errores rápido:** A diferencias de otros procesos, los tests unitarios nos permiten detectar los errores rápidamente, analizamos el código por partes, haciendo pequeñas pruebas y de manera periódica, además, las pruebas se pueden realizar las veces que hagan falta hasta obtener el resultado óptimo.

**→** **Facilita los cambios y favorece la integración:** Los tests unitarios, nos permiten modificar partes del código sin afectar al conjunto, simplemente para poder solucionar bugs que nos encontramos por el camino. Los tests unitarios, al estar desglosados en bloques individuales permiten la integración de nuevas aportaciones para hacer un código más complejo o actualizarlo en función de lo que el cliente demande.

**→ Proporciona información:** Gracias al continuo flujo de información y la superación de errores, se puede recopilar gran cantidad de información para evitar bugs venideros.

**→ Proceso debugging:** Los Tests unitarios ayudan en el proceso de debugging. Cuando se encuentra un error o bug en el código, solo es necesario desglosar el trozo de código testeado. Este es uno de los motivos principales por los que los tests unitarios se hacen en pequeños trozos de código, simplificando mucho la tarea de resolver problemas.

**→** **El diseño:** Si primero se crean los tests, es mucho más fácil saber con anterioridad cómo debemos enfocar el diseño y ver qué necesidades debemos cumplir. Testeando una pieza del código, también puedes saber que requisitos debe cumplir, y por eso mismo te será mucho más fácil llegar a una cohesión entre el código y el diseño.

**→** **Reduce el coste:** Partiendo de la base que los errores se detectan a tiempo, lo cual implica tener que escribir menos código, poder diseñar a la vez que se crea y optimizar los tiempos de entrega, vemos una clara relación con una reducción económica.

**4. Verifique cuantas pruebas automatizadas existen en el mercado y a que lenguajes de desarrollo se le aplican**

**-->Pruebas Funcionales:** Comprueba si el software produce los resultados correctos sin errores ni lagunas.

**--> Pruebas no funcionales:** Estas pruebas miden varias cosas, entre ellas:

El rendimiento del software en diversas circunstancias

Cómo funciona el software bajo las cargas de usuarios previstas, incluyendo los picos de uso

El funcionamiento del software en las condiciones de mayor carga.

**-->Análisis del código:** examina el código e identifica los problemas mediante pruebas como código que no funciona y código que tiene problemas de interfaz.

**-->Pruebas unitarias:** Las pruebas de integración garantizan que las unidades funcionan juntas cuando se conectan entre sí. Examina si los componentes cooperan lógicamente y producen los valores correctos. También comprueba si los módulos funcionan con herramientas de terceros.

**-->Pruebas de humo:** Los desarrolladores utilizan pruebas de humo para verificar la estabilidad de todo el sistema después de cada nueva compilación.

**-->Pruebas de rendimiento:** Las pruebas de rendimiento miden el funcionamiento del software. Su principal preocupación es la calidad general del software, los puntos en los que falla, la velocidad y la escalabilidad.

**-->Pruebas de regresión:** Las pruebas de regresión garantizan que el nuevo código, la reparación de errores o las actualizaciones no rompan la funcionalidad de los componentes previamente existentes en el software.

**-->Pruebas API:** Las pruebas de la API garantizan que dos componentes sean capaces de comunicarse entre sí de forma fiable y segura en diversos escenarios. Un marco de automatización de pruebas de API debe ser fácil de usar, escalable y reutilizable.

**Las pruebas de software pueden ser escritas en una variedad de lenguajes de programación tales como:**

**→ Java**: Es ampliamente utilizado en la industria para escribir pruebas de software, especialmente para herramientas de automatización de pruebas como Selenium y JUnit.

**→ Python:** Es otro lenguaje popular para escribir pruebas de software debido a su simplicidad y legibilidad. Se utiliza en herramientas de automatización como Selenium con Python, pytest, y unittest.

**→** **C#:** Es el lenguaje de programación principal utilizado en el entorno de desarrollo de Microsoft, y se utiliza para escribir pruebas de software en herramientas como Selenium con C# y NUnit.

**→** **JavaScript:** Es ampliamente utilizado para escribir pruebas de software, especialmente para pruebas de aplicaciones web. Herramientas como Mocha, Jasmine y Jest son populares para escribir pruebas en JavaScript.

**→** **Ruby:** Es el lenguaje de programación utilizado en el marco de automatización de pruebas Ruby on Rails, y también se utiliza en herramientas de prueba como Capybara y RSpec.

**→ PHP:** Es comúnmente utilizado en el desarrollo web y también se puede utilizar para escribir pruebas de software, especialmente para pruebas de aplicaciones web. PHPUnit es una herramienta popular de prueba de unidad en PHP.

**→C++:** Se utiliza en entornos donde se requiere un rendimiento muy alto, como pruebas de sistemas embebidos o aplicaciones de alto rendimiento.

**→Swift y Objective-C:** Se utilizan para escribir pruebas de software en el entorno de desarrollo de iOS.

