Las pruebas unitarias (o unit tests en inglés) son un tipo de pruebas de software donde las unidades más pequeñas del código, como funciones, métodos, o clases, se prueban individualmente para asegurarse de que funcionan correctamente. El objetivo es verificar que cada componente del software realiza su función según lo esperado y cumple con los requisitos especificados.

Características de las Pruebas Unitarias

Aisladas: Las pruebas unitarias se diseñan para probar una sola unidad de código, aislándola de otras partes del sistema. Se utilizan "stubs" o "mocks" para simular las dependencias y mantener el foco en la unidad bajo prueba.

Rápidas: Como se prueban unidades pequeñas, las pruebas unitarias son rápidas de ejecutar. Esto permite ejecutar muchas pruebas en poco tiempo y hacer pruebas frecuentes.

Automatizadas: Las pruebas unitarias suelen ser automatizadas y forman parte del proceso de integración continua, permitiendo la ejecución automática tras cada cambio en el código.

Determinísticas: Los resultados de las pruebas unitarias deben ser consistentes, dando el mismo resultado cada vez que se ejecutan bajo las mismas condiciones.

Conceptos Relacionados con Pruebas Unitarias

Cobertura de Pruebas: Como se explicó anteriormente, esto mide cuánto del código ha sido ejecutado por las pruebas unitarias. Puede ser cobertura de declaraciones, ramas, funciones, etc.

Mocks y Stubs: Herramientas para simular dependencias externas. Los "mocks" simulan el comportamiento de un componente externo y permiten controlar las respuestas, mientras que los "stubs" proporcionan respuestas predefinidas para llamadas específicas.

Asserts (Afirmaciones): Son condiciones que se verifican en las pruebas unitarias. Si una afirmación falla, la prueba se considera fallida.

Test Fixtures: Conjuntos de datos o configuraciones necesarias para que las pruebas unitarias se ejecuten de manera consistente. Esto incluye configuración antes de la prueba y limpieza después de la prueba.

Refactorización y Pruebas Unitarias: La refactorización es la modificación del código sin cambiar su funcionalidad. Las pruebas unitarias permiten refactorizar el código con confianza, ya que proporcionan una manera de comprobar que el comportamiento sigue siendo el mismo después de la modificación.

TDD (Test-Driven Development): Enfoque en el que las pruebas unitarias se escriben antes del código de producción. El desarrollo está guiado por la creación de pruebas que definen la funcionalidad esperada.

Ventajas de las Pruebas Unitarias

Detección Temprana de Errores: Permiten identificar errores a nivel de unidad antes de que afecten al sistema completo.

Facilitan la Refactorización: Con un conjunto sólido de pruebas unitarias, los desarrolladores pueden refactorizar con confianza.

Fomentan el Código Modular: Para hacer pruebas unitarias efectivas, el código debe ser modular y tener dependencias claras.

Integración Continua: Las pruebas unitarias permiten la integración continua y el despliegue continuo, facilitando el desarrollo ágil.

Desafíos de las Pruebas Unitarias

Mantenimiento de Pruebas: A medida que el código evoluciona, las pruebas unitarias también deben actualizarse.

Pruebas Insuficientes: Pruebas unitarias mal diseñadas pueden no cubrir todas las rutas o condiciones.

Difícil de Configurar: En algunos casos, aislar las unidades y crear las pruebas adecuadas puede ser complejo.