INFORME DEL TESTING Y PRUEBAS DE CÓDIGO

INTRODUCCIÓN:

- El testing y las pruebas de código desempeñan un papel crucial en el ciclo de vida del desarrollo de software, asegurando la calidad y confiabilidad del producto final. Estas prácticas son esenciales para identificar y corregir errores, garantizar el cumplimiento de requisitos y mejorar la experiencia del usuario.

CONCEPTOS BÁSICOS:

- DEFINICIÓN Y DIFERENCIA ENTRE TESTING Y PRUEBAS DE CÓDIGO:

    - El testing es un proceso de evaluación de un sistema para identificar los defectos, sin embargo las pruebas de código se centran en verificar el correcto funcionamiento de unidades individuales de código.
    - El objetivo de las pruebas de testing es la corrección de posibles errores, entrega de un software más fiable, además de mejorar la calidad del código.
      Los beneficios más notables podrían ser la identificación de problemas hasta la mejora del mantenimiento.

- TIPOS DE PRUEBAS:

    - Pruebas unitarias: son de muy bajo nivel y se realizan cerca de la fuente de la aplicación. Consisten en probar métodos y funciones individuales de las clases, componentes o módulos que usa tu software. En general, las pruebas unitarias son bastante baratas de automatizar y se pueden ejecutar rápidamente mediante un servidor de integración continua.

    - Pruebas de integración: las pruebas de integración verifican que los distintos módulos o servicios utilizados por tu aplicación funcionan bien en conjunto. Por ejemplo, se puede probar la interacción con la base de datos o asegurarse de que los microservicios funcionan bien en conjunto y según lo esperado. Estos tipos de pruebas son más costosos de ejecutar, ya que requieren que varias partes de la aplicación estén en marcha.

    - Pruebas de sistema: Se centran en evaluar el sistema completo. Selenium es comúnmente utilizado para pruebas de interfaz de usuario.

- Pruebas de aceptación: son pruebas formales que verifican si un sistema satisface los requisitos empresariales. Requieren que se esté ejecutando toda la aplicación durante las pruebas y se centran en replicar las conductas de los usuarios. Sin embargo, también pueden ir más allá y medir el rendimiento del sistema y rechazar cambios si no se han cumplido determinados objetivos.

- Pruebas de carga y estrés: Evalúan la capacidad del sistema para manejar carga normal y situaciones de estrés. Apache JMeter es ampliamente utilizado para pruebas de carga.

- TÉCNICAS DE TESTING:

  - TDD (Test Driven Development): Implica escribir pruebas antes del código de producción.

    Algún beneficio del TDD podría ser la mejora de la comunicación. Puesto que los problemas llegan en cortos periodos de tiempo.

  - BDD (Behaviour Driven Development): Se basa en el comportamiento del sistema desde el punto de vista del usuario.

    Algún beneficio de BDD puede ser la automatización de pruebas debido a la claridad y comprensión compartida de los escenarios.

- CASOS DE USO Y EJEMPLOS:

  - Pruebas de Integración con Postman: en una situación de desarrollo de API, lo utilizamos para realizar pruebas de integración entre los distintos servicios.
  - TDD con Phyton y pytest: seguimos la metodología TDD al escribir pruebas antes de desarrollar funcionalidades en un proyecto Python utilizando pytest.
  - Pruebas unitarias con JUnit: en un proyecto de Java, implementamos pruebas unitarias con JUnir para verificar funciones individuales.

- CONCLUSIÓN:

  - En conclusión, las pruebas de testing son un proceso muy importante, para localizar posibles errores en nuestra estructura y así poder mejorarla en todo lo que se pueda. A parte de solucionarnos los posibles problemas que nos surjan, también contribuye a la confiabilidad y satisfacción de los clientes.