Informe de Testing y Pruebas de Código - Aitor Alguacil Maroto

Introducción:

En el mundo del desarrollo de software, el testing y las pruebas de código desempeñan un papel esencial. Este informe proporciona una breve descripción sobre por qué estas prácticas son fundamentales para asegurar la calidad y confiabilidad de las aplicaciones que creamos.

Conceptos Básicos:

- Definición y diferencia entre testing y pruebas de código:

Testing se refiere a evaluar el sistema en busca de errores, mientras que las pruebas de código se centran específicamente en validar el código fuente.

- Objetivos y beneficios de realizar pruebas:

Los objetivos incluyen la detección temprana de errores, mejorar la calidad del software y reducir costos. Los beneficios abarcan desde la identificación de problemas hasta la validación de requisitos.

Tipos de Pruebas:

- Pruebas unitarias: se evalúan unidades individuales de código para verificar su funcionamiento correcto.
- Pruebas de integración: se centran en la interacción entre unidades, mientras que las pruebas de sistema evalúan el sistema completo.
- Pruebas de aceptación: verifican si el software cumple con los requisitos del usuario.
- Pruebas de regresión: aseguran que las actualizaciones no afecten funcionalidades existentes.

Ejemplos de herramientas según el tipo de pruebas:

- JUnit para pruebas unitarias
- Selenium para pruebas de aceptación
- Apache JMeter para pruebas de carga.

Técnicas de Testing:

- TDD (Test Driven Development): El Desarrollo Guiado por Pruebas (TDD) implica escribir pruebas antes de implementar el código funcional. Las pruebas guían el desarrollo, asegurando que cada unidad de código cumpla

con requisitos específicos. TDD promueve modularidad, claridad y detección temprana de errores, mejorando la calidad y mantenibilidad del código.

- BDD (Behavior Driven Development): El Desarrollo Guiado por Comportamiento (BDD) se enfoca en el comportamiento del software desde la perspectiva del usuario. Utiliza un lenguaje natural para describir el comportamiento esperado, facilitando la comunicación entre equipos y stakeholders. BDD traduce estos escenarios en pruebas automatizadas, asegurando que el software cumpla con los requisitos del usuario.

Ambas técnicas promueven la claridad en los requisitos y mejoran la calidad del código.

Automatización de Pruebas:

La automatización de pruebas implica el uso de scripts y herramientas para ejecutar pruebas de manera eficiente y sus principales ventajas son:

- Mejora la velocidad
- Repetibilidad y cobertura de las pruebas
- Reducción de errores humanos

Ejemplos de herramientas:

- Selenium
- Appium para pruebas móviles
- JUnit/TestNG para automatización de pruebas unitarias

Casos de Uso y Ejemplos:

Ejemplo:

- Implementación de pruebas unitarias en un módulo crítico del sistema utilizando TDD, demostrando cómo estas prácticas contribuyen a la calidad y mantenibilidad del código.

Conclusión:

En conclusión, el testing y las pruebas de código son esenciales para asegurar la calidad y confiabilidad del desarrollo de software. La diferencia entre testing y pruebas de código se centra en evaluar el sistema y validar el código fuente, respectivamente. Con objetivos claros, herramientas como JUnit y Selenium, y técnicas como TDD y BDD, se logra una evaluación completa. La automatización de pruebas, con herramientas como Selenium y frameworks como JUnit/TestNG, mejora la eficiencia y reduce errores.

Referencias:

- https://www.hiberus.com/crecemos-contigo/testing-fase-de-testeo-de-software/#:~:text=Por%20qu%C3%A9%20el%20testing%20es,un%20producto%20de%20buena%20calidad.
- https://profile.es/blog/que-es-el-testing-de-software/
- https://trycore.co/transformacion-digital/tipos-de-pruebas-funcionales/
- https://www.arsys.es/blog/behavior-driven-development-vs-test-driven-development