22 de enero de 2024

INFORME DE TESTING Y PRUEBAS DE CÓDIGO

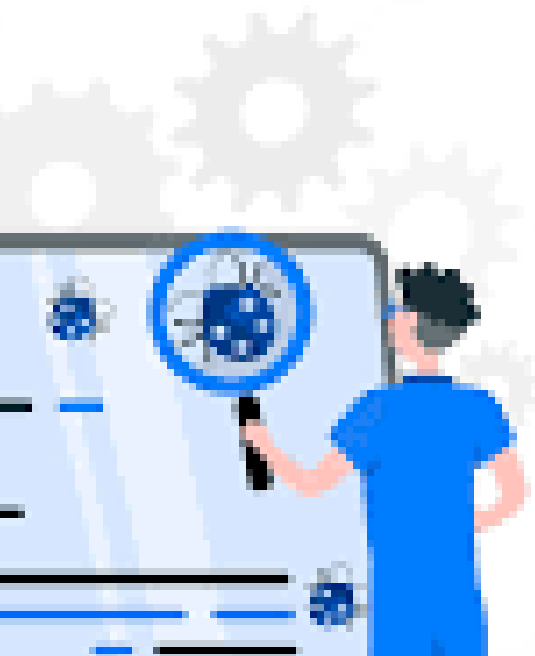
Computer Science & AI

Preparado por: Pablo Nicolás Soto Irago

INTRODUCCIÓN

En el ciclo de vida del desarrollo de software, el testing y las pruebas de código son componentes críticos para garantizar la calidad y confiabilidad del producto final. Estas actividades no solo identifican defectos y errores, sino que también mejoran la eficiencia y la mantenibilidad del código.

La evaluación de software o Aseguramiento de Calidad del Software (QA, por sus siglas en inglés) es un procedimiento para comprobar y validar la funcionalidad de un programa o aplicación con el propósito de asegurar que el producto esté libre de defectos. El objetivo final es que coincida con los requisitos esperados para entregar un producto de calidad. Las pruebas de código se centran específicamente en verificar la funcionalidad y calidad del código fuente. Mientras que el testing evalúa el sistema en su conjunto, las pruebas de código se adentran en el nivel del código fuente.
Los objetivos de las pruebas abarcan validar la funcionalidad, identificar errores y asegurar que el software cumpla con los requisitos. Los beneficios comprenden la detección temprana de defectos, la reducción de costos de corrección y el mejoramiento de la confiabilidad del software.



TIPOS DE PRUEBAS

- Pruebas manuales: Aquellas que en las que se prueba la navegación por la aplicación interactuando con la misma.
- Pruebas automáticas: Se utiliza una herramienta para realizar los test de código. Hay varios tipos de pruebas automáticas:

Pruebas unitarias: Evalúan cada unidad individual del código por separado y analizarla fuera de su contexto para validar su correcto funcionamiento.

Pruebas de integración: Verifican la interacción del sistema. Se podrían hacer estas pruebas teniendo pruebas unitarias para comprobar si funciona la conexión entre sí.

Pruebas estáticas: Revisión de documentos con el objetivo de seguir los flujos de la aplicación. Se suelen realizar sin ejecutar el código.

Pruebas dinámicas: Permiten recopilar información sobre el funcionamiento del programa. Estas pruebas necesitan que el código este en ejecución.

Pruebas ESRE: Las realizan los betatester para validar si el sistema se comporta de la forma que indican los requerimientos.

- Pruebas funcionales: Es una prueba basada en la ejecución, revisión y retroalimentación de las funcionalidades. (Pruebas de sistema, humo, Alpha beta...)
- Pruebas no funcionales: Tienen como objetivo la verificación de un requisito que especifica los criterios de requisitos. (Pruebas de compatibilidad, seguridad, estrés, rendimiento...)
- Pruebas de servicios web (API): Tipo de prueba de software que valida los servicios web
- Pruebas de carga: Las pruebas de carga son la práctica de simular el uso del mundo real, o cargar, en cualquier software, sitio web, aplicación web, API o sistema para analizar e identificar factores como la capacidad de respuesta, degradación y escalabilidad.
- Pruebas de rendimiento: Análisis de comprobación del funcionamiento del sistema frente a múltiples escenarios de ensayo, con el fin de examinar los componentes de la aplicación.
- Pruebas de compatibilidad del navegador: Garantizan que una aplicación o sitio web funcione en varios navegadores, lo cual es vital porque utilizan motores de diseño distintos.

HERRAMIENTAS POPULARES ASOCIADAS A CADA TIPO DE PRUEBA

Según las pruebas, podemos usar diversos tipos de herramientas. Algunas son:

- Selenium: Pruebas automáticas, Pruebas manuales, Pruebas de integración, Pruebas funcionales.
- SoapUI: Pruebas automáticas, Pruebas de integración, Pruebas de servicios web (API).
- Loadunner: Pruebas automáticas, Pruebas de carga y rendimiento.
- LambdaTest: Pruebas automáticas, Pruebas de compatibilidad del navegador.
- WebLOAD Professional: Pruebas automáticas, Pruebas de carga y rendimiento.

TÉCNICAS DE TESTING

Las técnicas de testing son diferentes estrategias o enfoques utilizados para probar una aplicación y garantizar que se comporte y visualice según lo esperado. Estas pruebas ayudan a los equipos a evaluar el software con los requisitos e información recopilada de los usuarios y el propietario del producto, simplificando la vida de los desarrolladores. Exploraremos técnicas como TDD, BDD y otras relevantes:

- TDD (Desarrollo Dirigido por Pruebas): Metodología agile de desarrollo de software donde se documenta y diseña una aplicación en función del comportamiento que los usuarios esperan experimentar al interactuar con ella.
- BDD (Desarrollo Dirigido por Comportamiento): Metodología de programación en la que se escriben primero las pruebas, generalmente unitarias, y luego se escribe el código fuente de manera que supere esas pruebas.

La diferencia clave entre ambos enfoques es el alcance. TDD es una práctica de desarrollo, mientras que BDD es una metodología de equipo. En TDD, los desarrolladores escriben las pruebas, mientras que en BDD, las especificaciones automatizadas son creadas por usuarios o testers.

- TDD mejora la calidad del código, pero puede requerir un cambio cultural.
- BDD facilita la comunicación entre desarrolladores y partes interesadas, aunque la definición de comportamientos puede ser compleja.

AUTOMATIZACIÓN DE PRUEBAS

La automatización de Pruebas es el proceso de utilizar herramientas de software que ejecutan software recién desarrollado o actualizaciones a través de una serie de pruebas para identificar posibles errores de codificación, cuellos de botella y otros obstáculos para el rendimiento. La automatización de pruebas acelera el proceso, mejora la cobertura y proporciona resultados consistentes.

HERRAMIENTAS Y FRAMEWORKS POPULARES

- Selenium: Herramienta muy útil y eficaz con integración con otras herramientas y capacidad de pruebas cruzadas en diferentes navegadores y sistemas operativos. Es altamente flexible en cuanto a lenguajes de programación y navegadores. Se suele usar para probar la funcionalidad y el rendimiento de una aplicación web.
- Cypress: Utilizado principalmente para probar aplicaciones web modernas. Se usa Javascript como lenguaje a la hora de crear script de pruebas. Nos permite escribir diferentes tipos de pruebas de la pirámide de Cohn.
- Appium: Framework de código abierto compatible con múltiples plataformas (Android, iOS...). Cuenta con una amplia variedad de funciones que permiten realizar pruebas complejas, incluyendo la capacidad de interactuar con el DOM, realizar pruebas de gestos, hacer pruebas de localización, entre otros. Es compatible con múltiples lenguajes de programación, incluyendo Java, Python y Ruby.
- Playwright: Framework creado por Microsoft para automatización de pruebas para aplicaciones web, diseñado para trabajar con los principales navegadores (Chrome, Firefox, Safari y Edge) y que proporciona una API unificada para interactuar con ellos. Se enfoca en proporcionar una experiencia de usuario más rápida y confiable en comparación con otras herramientas de automatización de pruebas web.
- Robot Framework: Este framework genérico y de código abierto se utiliza para probar sistemas de software de cualquier tipo. Debido a esto se usa en una gran variedad de industrias, y con él puedes realizar pruebas de aceptación y pruebas de unidad. los usuarios pueden crear sus propias bibliotecas y plug-ins para Robot Framework, o utilizar bibliotecas de terceros para una amplia gama de funcionalidades, como la simulación de dispositivos móviles o la automatización de pruebas de API.

CONCLUSIÓN

La relevancia de la evaluación y verificación del código en el desarrollo de software es esencial. Estas acciones no solo detectan errores, sino que también fomentan la confianza en la calidad del producto final. La inversión en pruebas es crucial para asegurar el éxito a largo plazo de cualquier proyecto de software.

REFERENCIAS

Novoseltseva, E (2024): Técnicas De Testeo De Software Y Herramientas. Disponible en: https://apiumhub.com/es/tech-blog-barcelona/tecnicas-de-testeo-de-software/

KeeepCoding Team (2024) ¿Qué son las pruebas de rendimiento? Disponible en: https://keepcoding.io/blog/que-son-las-pruebas-de-rendimiento

(2024) Pruebas de carga. Disponible en: https://www.loadview-testing.com/es/pruebas-de-carga/

ZAPTEST (2024) Pruebas de compatibilidad: qué son, tipos, proceso, características, herramientas y mucho más. Disponible en: https://www.zaptest.com/es/pruebas-de-compatibilidad-que-son-tipos-proceso-caracteristicas-herramientas-y-mucho-mas

ZAPTEST (2024) ¿Qué es la automatización de pruebas? Una guía sencilla y sin jerga. Disponible en: https://www.zaptest.com/es/que-es-la-automatizacion-de-pruebas-una-guia-sencilla-y-sin-jerga

Chaverra, A (2024) Cypress: Una verdadera experiencia de automatización. Disponible en: https://www.ceiba.com.co/ceiba-blog/cypress-una-verdadera-experiencia-de-automatizacion

Martínez, M (2023): Qué es el testing de software. Disponible en: https://profile.es/blog/que-es-el-testing-de-software/

Hamilton, T (2023) Tutorial de prueba de servicios web: ¿Cómo realizar la prueba? Aprende con el ejemplo. Disponible en: https://www.guru99.com/es/webservice-testing-beginner-guide.html

MKT (2023) Conoce Las Técnicas de Software Testing. Disponible en: https://trans-ti.com/2020/11/23/conoce-las-tecnicas-de-software-testing/

Fuentes, F (2023) TDD vs BDD: Test-driven development vs. Behavior-driven development. Disponible en: https://www.arsys.es/blog/behavior-driven-development-vs-test-driven-development

Valls, M (2023) 5 Frameworks comunes en Automatización de Pruebas y cuándo usarlos. Disponible en: https://www.cleveritgroup.com/blog/5-frameworks-comunes-en-automatizacion-de-pruebas-y-cuando-usarlos

García, G (2021): 5.Control de versiones y testing: Testing y prueba de código. Disponible en: https://www.linkedin.com/learning/fundamentos-esenciales-de-la-programacion-2/testing-y-prueba-de-codigo?resume=false