INFORME DE TESTING Y PRUEBAS DE CÓDIGO

ÍNDICE

1.- INTRODUCCIÓN:

Cuando se empieza a desarrollar un software, se sabe que el objetivo final es que funcione correctamente y que sea utilizado para desarrollar el trabajo para el que fue creado. Los testing y pruebas de código, son herramientas imprescindibles que se deben utilizar paralelamente al desarrollo del software ya que, es la forma que hay de prevenir que una vez esté terminado, no sea operativo porque tenga errores, ya que gracias a estas herramientas se pueden corregir o modificar justo en la parte del proceso que dé el error y poder seguir avanzando en el proceso con fiabilidad.

2.- HABILIDADES Y GESTIÓN DE TIEMPO

2.1 Definición y diferencias entre testing y pruebas de código.

Un testing, también llamado software QA, es el método de verificación, de que todo está funcionando correctamente durante el desarrollo de un software, está basado en la realización de análisis, evaluaciones de productos, y observaciones, es una prueba de que el software se está desarrollando bien sin errores ya que en caso de encontrarlos se pueden ir reparando. Las herramientas que se utilizan para hacer estas pruebas pueden ser manuales o automatizadas.

Las pruebas de código son uno de los tipos de testing. Son las llamadas pruebas unitarias. Es el proceso de prueba de la unidad funcional de código más pequeña. La más básica. En el desarrollo del software se recomienda escribirlo con unidades pequeñas y funcionales porque así se puede hacer una prueba unitaria para cada unidad de código. Así, si una prueba sale fallida, se puede ver rápidamente el código que tiene el error y se puede reparar con rapidez. Las pruebas de código son muy importantes porque mejoran mucho la calidad del software.

La diferencia que hay entre el testing y las pruebas de código es que la segunda es uno de los tipos de la primera y se diferencia de todos los demás en que el resto de los métodos de prueba necesitan para verificarlas, herramientas

especializadas o procesos independientes y alguna de ellas no se pueden verificar hasta el final del proceso. Las pruebas de código en cambio, se ejecutan a la vez que se crea el código y no requieren herramientas especiales. Es la prueba más básica del testing.

2.2 Objetivos y beneficios de realizar pruebas.

El objetivo final en la creación de cualquier proyecto en este caso de un software, es que sea funcional y pueda utilizarse para lo que se ha creado de la manera más eficiente posible. Por tanto, la realización de pruebas para que esto ocurra es fundamental y eso se consigue con la utilización de testing.

Los beneficios que se consiguen son muchos como asegurar que es de fácil uso para los usuarios para los que está destinado, porque no todo el mundo tiene grandes conocimientos informáticos y hay que asegurarse que va a ser funcional.

Que se lanza al mercado con una garantía de que va a funcionar porque si ha sido probado da una mayor fiabilidad.

La garantía de que es accesible para todas las plataformas, porque con las pruebas se ha comprobado la accesibilidad y la compatibilidad.

Se ahorran sobrecostes que se producirían si no funcionase porque si hay que repararlo a posteriori sería más costoso que si se hace en el momento de la creación y además sería menos productivo porque tardaría más en poder utilizarse.

3.- TIPOS DE PRUEBAS.

3.1 descripción de los tipos de pruebas

No sabremos si el software funciona, hasta que no lo probemos. Para ello el testing tiene dos tipos diferentes de pruebas: Las manuales que las hacen personas con las herramientas adecuadas y que es un proceso más costoso y que además puede llevar a errores, y las automatizadas que son realizadas por

una máquina y pueden llegar a ser muy complejas desde verificar un solo código hasta desarrollar un método para analizar toda una secuencia.

Además se pueden clasificar de otro modo entre tipos: funcionales y no funcionales.

Los primeros analizan los requisitos comerciales. Es decir, que cumpla las exigencias que el usuario necesita. Hay varios tipos:

-De unidad: Son las pruebas de código, unitarias y que comprueban que funciona cada componente por separado.

-De integración que verifican que los módulos funcionan bien juntos

-De sistema Comprueban la eficiencia de funcionamiento entre componentes y el sistema

-De humo: Comprueban cosas básicas que puede hacer el usuario como apagar o iniciar sesión.

-De cordura: comprueban el razonamiento lógico

-De Regresión: Si se han hecho cambios, comprueban que el sistema sigue funcionando.

-Beta: Tienen un sistema que permite que los usuarios puedan hacer pruebas y comunicar fallos o mejoras

-Interface: Permiten comprobar la comunicación correcta entre dos sistemas de software.

Las no funcionales no analizan aspectos técnicos sino otros como la fiabilidad, el rendimiento o cómo se va a usar.

Algunos de ellos son:

-Test de rendimiento, que analiza la confiabilidad y la capacidad de respuesta.

-Stress testing Prueba la solidez del programa si se le da un uso más allá de lo normal para lo que está programado.

-Prueba de volumen, Comprueba el comportamiento si se carga con una mayor cantidad de datos de lo usual.

-Security test, Evalúa y comprueba su protección ante los ataques

-Prueba de compatibilidad, funcionamiento en diferentes dispositivos y navegadores.

-Pruebas de recuperación: Comportamiento ante caídas y fallos del sistema.

-Reliability testing Qué confianza puede dar a una tarea sin fallos durante un tiempo prolongado.

-Usabilidad: Prueba la facilidad de uso para quien lo vaya a utilizar

-Localización testing. Verifica el comportamiento del producto según el entorno donde va a ser utilizado.

3.2 Herramientas populares.

Existen muchas herramientas disponibles de testing. La elección de cada una de ellas dependerá del tipo de test que se vaya a utilizar y del programa que se use.

En los testing de las pruebas de código

-JUnit: Es una herramienta para utilizar con Java. Es fácil de usar y se integra bien con los frameworks de desarrollo de Java.

-NUnit: herramienta de testing de unidad para .NET. Es parecida a JUnit y es fácil de usar.

-PyTest: Se usa con Python. Es fácil y también se integra bien con los frameworks de desarrollo de Python.

Testing de sistema

-Selenium: Es una herramienta de testing de integración para plicaciones web. Se utiliza para pruebas de interfaz de usuario o pruebas funcionales en aplicaciones web.

-HP Quality Center: Es una herramienta que ayuda a planificar, diseñar y ejecutar pruebas de sistema.Tienen una amplia gama de funcionalidades

-IBM Rational Functional Tester: Es para aplicaciones Java y .NET.

Testing de integración (Integration testing)

- Selenium: aunque se utiliza principalmente para pruebas en aplicaciones web, también se puede utilizar para pruebas de integración

-Apache JMeter: Es una herramienta de carga y de integración.

Testing de aceptación

-FitNesse: Es también para usar en aplicaciones web. Se utiliza para escribir pruebas en lenguaje normal y es fácil

-Cucumber: Al igual que la anterior Es para aplicaciones web.

Testing de humo (Smoke testing)

-Jenkins es una herramienta que se utiliza para realizar el testing de humo, es decir para probar cosas básica que puede hacer el ususario

Testing de regresión (Regression testing)

-TestComplete: herramienta de testing de regresión que se utiliza para probar aplicaciones de escritorio, web y móviles. Es fácil de usar y ofrece una amplia gama de funcionalidades para el testing de regresión.

-Ranorex: Se usa para probar aplicaciones de escritorio, web y móviles. Su uso es sencillo y puede ofrecer una amplia gama de funcionalidades.

Testing de carga

-Apache JMeter: como se mencionó anteriormente, es una herramienta de testing de carga y de integración. Se utiliza para probar el rendimiento de la aplicación bajo diferentes cargas y condiciones.

-LoadRunner: Se utiliza para probar aplicaciones web, de escritorio y móviles.

En las no funcionales como por ejemplo las pruebas de estrés , también se utilizan herramientas como soapUI que simula peticiones para servidores.

Cada una de las herramientas tienen sus fortalezas y sus debilidades, la elección de una o de otra dependerá de las necesidades que se tengan en la realización del proyecto y de la prueba que se quiera realizar pero elegir una herramienta correctamente puede ser esencial para garantizar un buen resultado final

4.- TÉCNICAS DE TESTING

4.1 Exploración de técnicas

Las dos principales técnicas de testing son las técnicas de la caja blanca y las técnicas de la caja negra, cada una de ellas engloba a su vez diferentes técnicas.

La caja blanca es una técnica para realizar las pruebas de software en la que los métodos de comprobación del funcionamiento de la estructuras, se refieren a la parte interna y el diseño del software. Por otro lado, las pruebas de caja negra, no se ocupan de las operaciones internas del software, se ocupan de comprobar los resultados externos.

-Análisis del valor límite (BVA)

Se realizan pruebas en los límites entre particiones. Con límites máximos, mínimos, externos e internos y valores típicos de error. Es una técnica de caja negra se basa en que, si un sistema funciona bien para estos valores particulares, funcionará igualmente bien para todos los valores que se encuentran entre los dos valores límite.

-Particionamiento de clase de equivalencia

Es una técnica que permite dividir el conjunto de condiciones de prueba en una partición que debe considerarse igual. Su principio es que el caso de prueba de un valor representativo de cada clase es igual a una prueba de cualquier otro valor de la misma clase. así, se pueden identificar equivalencias válidas y no válidas.

-Transición de estado

En esta los cambios en la entrada, cambian el estado de la aplicación bajo prueba (AUT). así se puede probar el comportamiento de un AUT.En esta técnica, se proporcionan valores de prueba de entrada positivos y negativos para evaluar el comportamiento del sistema.

- Prueba basada en la tabla de decisiones

También se llama tabla causa-efecto, es una técnica para funciones que responden a una combinación de entradas o eventos. Por ejemplo, prueba que el botón de envío esté habilitado si el usuario llena todos los campos obligatorios.

-Error adivinando

Es una técnica que se utiliza para adivinar un error que puede mantenerse en el código. Se basa en la experiencia de la persona que la prueba para poder encontrar errores. Cuenta con una lista de posibles errores o situaciones propensas a errores. quien la está probando se encarga de exponer esos errores a través de un caso de prueba.

-Test Driven Development (TDD)

Es una metodología de desarrollo de software que se centra en escribir pruebas automatizadas antes de escribir el código de la aplicación. Se realizan pruebas unitarias. crea casos de pruebas para cada funcionalidad que se quiera desarrollar, se testea, y en caso de detectar algún fallo, se reescribe un código libre de errores. Así, el desarrollo es más rápido porque elimina la duplicación del código. Esta técnica se basa en un test de pruebas que guían el proceso , no se realizan en un proceso posterior y se desarrolla en tres fases:

Desarrollo y escritura de la prueba, validación de la prueba y refactorización.

-Behavior Driven Development (BDD).

Es una técnica orientada al comportamiento que los usuarios de la aplicación esperan al utilizarla, se analiza desde su perspectiva.

Esta técnica combina y perfecciona la técnica anterior (TDD).Utiliza un lenguaje estructurado GWT para escribir los casos. GIVEN-WHEN-THEN.

4.2 Beneficios y retos asociados a las técnicas.

La utilización de técnicas de testing tiene muchas ventajas, en el caso del TDD mejora la calidad del código porque fomenta su escritura modular y cohesiva y como son pruebas automatizadas garantizan que el código funcione según está previsto.

Detecta de forma rápida y muy pronto los errores ya que como se escriben pruebas antes de poner la funcionalidad se pueden identificar los problemas y los errores antes de que cuente más corregirlos. Además, facilita la colaboración entre los equipos y los desarrolladores porque especifican claramente el comportamiento esperado.

Todo ello lleva a mejorar la productividad porque aunque puede llevar más tiempo desarrollar el software reducirá el tiempo dedicado a la corrección de errores y depuración.

Permiten hacer cambios en el código con confianza porque las pruebas son automatizadas y aseguran que las funcionalidades no se vean afectadas negativamente.

Los retos que se encuentran por delante a la hora de utilizar los testing es que requiere un aprendizaje y hay que adaptarse a los cambios para acostumbrarse a escribir pruebas antes del código. Se puede tener la sensación de que se pierde el tiempo y requiere mucho esfuerzo porque es necesario escribir pruebas antes de la implementación.

Garantizar una cobertura completa de pruebas puede ser un gran desafío, porque como lleva mucho tiempo, algunas partes del código pueden quedar sin probar.No hay que verlo como una restricción a la libertad de los creadores.

Tener un conjunto sólido de las herramientas de prueba automatizadas para poder tener una implantación efectiva.

En general compromiso y práctica por parte de los desarrolladores.

5.- AUTOMATIZACIÓN DE PRUEBAS.

5.1 Automatización y sus ventajas

La automatización de pruebas, también es llamada testing automatizado .Es un proceso en el que se usan herramientas y scripts de software para ejecutar pruebas en un sistema de manera automatizada, en lugar de hacerlo manualmente. Esta forma de testear es preferida a la forma manual en el desarrollo de software debido a sus muchas ventajas.

La automatización de pruebas permite una ejecución más rápida y eficiente de las pruebas en comparación con los métodos manuales. Al eliminar la necesidad de intervención humana en cada paso del proceso de prueba, se pueden ejecutar pruebas complejas en un corto período de tiempo, así se acelera el desarrollo de software y se mejora la productividad.

Además, con el uso de pruebas automatizadas, se garantiza una mayor precisión en los resultados porque los scripts realizan las pruebas de forma exacta y reproducible, lo que reduce la posibilidad de cometer los errores que una persona podría cometer.

Otra ventaja importante es la capacidad de realizar pruebas de regresión de manera eficiente. Si se automatizan las pruebas, se puede volver a ejecutar fácilmente un conjunto completo de pruebas cada vez que se realizan cambios en el código, así se puede ver rápidamente cualquier regresión o error no deseado.

También permite que unaa mayor capacidad para realizar pruebas de carga y estrés, porque permite hacer pruebas como si hubiese un gran número de usuarios utilizándolo al mismo tiempo o en condiciones extremas de uso, lo que ayuda a evaluar el rendimiento y la estabilidad del sistema.

En resumen, la automatización de pruebas ofrece numerosas ventajas, como mayor precisión, ejecución más eficienciete y rápida, y consistencia en los resultados, capacidad para realizar pruebas de regresión y carga, y por tanto, mejora de la productividad. Estas ventajas hacen que la automatización de pruebas sea una práctica esencial en el desarrollo de software actual.

5.2 Herramientas y Frameworks populares

Existen numerosas herramientas y frameworks populares para la automatización de pruebas. En apartados anteriores ya se han señalado algunos de ellos .

Un framework es una infraestructura predefinida que proporciona un conjunto de herramientas y pautas para facilitar el desarrollo y la implementación de aplicaciones. Estos están diseñados para ayudar a los desarrolladores a crear aplicaciones de manera más eficiente ya que proporcionan una estructura general, y funcionalidades comunes.

Los frameworks son herramientas muy poderosas que aceleran el desarrollo al proporcionar una base sólida y un conjunto de prácticas recomendadas, permitiendo a los desarrolladores centrarse más en la lógica específica de su aplicación en lugar de tener que abordar problemas comunes y estructurales desde el principio .

Algunos ejemplos de herramientas son

-Selenium es una herramienta de código abierto ampliamente utilizada para la automatización de pruebas en aplicaciones web. Soporta varios lenguajes de programación como Java, Python, C#, entre otros.

- Cucumber es una herramienta de automatización de pruebas basada en comportamientos (BDD). Utiliza lenguajes como Gherkin para escribir escenarios en un formato legible por humanos y es compatible con varios lenguajes de programación como Java, Ruby, C#, entre otros

- Jenkins además de una herramienta es un servidor de automatización de código abierto que se utiliza para la integración continua y la entrega continua (CI/CD). Puede ser configurado para ejecutar pruebas automáticamente después de cada cambio en el código fuente.

Ejemplos de Framework

-Appium es un framework de automatización de pruebas que se centra en aplicaciones móviles, permitiendo pruebas en dispositivos Android e iOS. Es compatible con varios lenguajes de programación como Java, Python, C#, entre otros.

-Matraz es un micro framework para Python. Se usa para el desarrollo web y es conocido por su facilidad y simplicidad. es una buena opción para proyectos pequeños

Laravel es un framework de aplicaciones web PHP, tiene una gran sintaxis y crea aplicaciones modernas y de alto rendimiento. Proporciona una estructura bien definida para el desarrollo de aplicaciones

-JUnit.- Es un framework de pruebas unitarias para Java. Se utiliza comúnmente en el desarrollo de software basado en Java y es compatible con la automatización de pruebas a nivel unitario.

- TestNG es otro framework de pruebas para Java que se inspira en JUnit pero agrega características adicionales, como configuración flexible de pruebas y paralelismo. Es especialmente útil para pruebas de integración y de extremo a extremo.

- Robot franework Es un framework de automatización de pruebas de código abierto que utiliza una sintaxis fácil de entender. Es compatible con pruebas de aceptación, pruebas de aceptación de interfaz de usuario y pruebas de procesos de negocio.

Hay muchas más herramientas y frameworks y la elección de cada una de ella dependerá del proyecto que se quiera realizar y los requisitos que se quieran cumplir.Además esta es una industria que está en continuo desarrollo y en el que se debe investigar según cada proyecto.

6.- CASOS DE USO Y EJEMPLOS

Un ejemplo que podríamos seguir relativo a un uso real de las técnicas de testing, sería la realización de una página web de comercio electrónico como la que se ha desarrollado en ese trabajo.

Primero se harían pruebas unitarias. Por ejemplo para desarrollar una función de cálculo de descuentos. Se crearían pruebas unitarias para verificar que la función de cálculo de descuentos devuelve resultados correctos para diferentes combinaciones de productos y cantidades.

Pruebas de Integración, para probar la funcionalidad de pago con elementos de pago externo.Se realizan pruebas para garantizar que la aplicación se integre correctamente con las pasarelas de pago y que los datos de transacción se manejen de manera segura.

Pruebas de Sistema para Implementar un sistema de gestión de inventario. Se llevarían a cabo pruebas de sistema para verificar que la funcionalidad de gestión de inventario permite la actualización y consulta correcta de los niveles de stock.

Pruebas de Aceptación del Usuario para desplegar una nueva interfaz de usuario. Los usuarios finales realizan pruebas de aceptación para evaluar la usabilidad y la experiencia general de la nueva interfaz de usuario.

Pruebas de Rendimiento: Gran cantidad de usuarios accediendo al mismo tiempo durante una oferta especial. Se ejecutan pruebas de carga para evaluar el rendimiento del sistema bajo condiciones de tráfico intenso y se identifican posibles cuellos de botella.

Pruebas de Seguridad. Implementación de autenticación de dos factores. Se realizan pruebas de seguridad para evaluar la robustez de la autenticación de dos factores y asegurar que los datos de los usuarios estén protegidos.

Pruebas de Regresión Actualización del botón de búsqueda de productos. Se ejecutan pruebas de regresión para garantizar que la actualización no afecte negativamente a las funcionalidades existentes.

Pruebas de Usabilidad Rediseño de la página de inicio. Los suarios participan en pruebas de usabilidad para evaluar la facilidad de uso y la eficacia del nuevo diseño de la página de inicio.

Estas son algunas de las pruebas que se podrían hacer para garantizar que nuestra página es fiable, segura y eficiente.

7.- CONCLUSIONES.

Como se ha dicho a lo largo de todo el informe, las pruebas de testing, son imprescindibles a la hoara de garantizar la v¡creación de un software fiable y productivo ya que aunque al realizarlas el proceso de creación sea más lento, a la larga los beneficios serán mucho mayores ya que nos garantizará que el producto cumple con los requisitos y expectativas que hemos marcado al principio-

8.- REFERENCIACIÓN (Webgrafía)

https://pacifitic.org/que-es-el-testing-de-software-y-por-que-es-tan-importante-en-el-desarrollo-de-software/

https://www.iebschool.com/blog/software-testing-que-es-tipos-digital-business/

https://programacionymas.com/blog/tipos-de-testing-en-desarrollo-de-software

https://www.hiberus.com/crecemos-contigo/las-mejores-herramientas-para-cada-tipo-de-testing/

https://www.tecnologias-informacion.com/software-testing.html

https://keepcoding.io/blog/que-es-el-testing-automatizado/