Testing, la importancia sobre la fase de testeo de software

Como bien sabemos, el proceso de creación de software se compone de varias fases. Desde su diseño hasta su puesta en producción, debe pasar por varios momentos en los que este software va evolucionando.

Qué es el testing de software

El **testing de software o software QA** es una disciplina en la ingeniería de software que permite tener procesos de ejecución de un programa o aplicación y una metodología de trabajo con el objetivo de localizar errores de software. También puede describirse como el proceso de validación y verificación de un programa de software o una aplicación.
Es imprescindible tener en cuenta que el testing es paralelo al proceso de desarrollo del software. A medida que se está construyendo nuestro producto, tenemos que realizar tareas de testing de software, siguiendo los **7 principios del testing**, para prevenir incidencias de funcionalidad y corregir desviaciones del software antes de su lanzamiento.

Por qué el testing es importante

A un alto nivel, las pruebas de software son necesarias para **detectar los errores en el software** y para probar si el software cumple con los requisitos del cliente. Esto ayuda al equipo de desarrollo a corregir los errores y entregar un producto de buena calidad.

Hay varios puntos en el proceso de desarrollo de software en los que el error humano puede llevar a un software que no cumple con los requisitos de los clientes. Algunos de ellos se enumeran a continuación.

- El cliente/persona que proporciona los requisitos en nombre de la organización del cliente puede no saber exactamente qué es lo que se requiere o puede olvidarse de proporcionar algunos detalles, lo que puede llevar a que falten características.
- La persona que está recopilando los requisitos puede malinterpretarlos o no cumplirlos por completo al documentarlos.
- Durante la fase de diseño, si hay problemas en el diseño, esto puede conducir a errores en el futuro.
- Los errores pueden ser introducidos durante la fase de desarrollo durante un error humano, falta de experiencia, etc.
- Los probadores pueden perder errores durante la fase de prueba debido a errores humanos, falta de tiempo, experiencia insuficiente, etc.

- Es posible que los clientes no dispongan del ancho de banda necesario para probar todas las funciones del producto y que liberen el producto a sus usuarios finales, lo que puede dar lugar a que los usuarios finales encuentren errores en la aplicación.
- El negocio y la reputación de una organización depende de la calidad de sus productos y en algunos casos incluso los ingresos pueden depender de las ventas de productos de software.

Los usuarios pueden preferir comprar un producto de la competencia en lugar de un producto de baja calidad, lo que puede resultar en una pérdida de ingresos para la organización. En el mundo actual, la calidad es una de las principales prioridades de cualquier organización.

¿Cómo funcionan las pruebas de software?

La prueba de software es el proceso de evaluar y verificar que un producto o aplicación de software hace lo que se supone que debe hacer. Los beneficios de las pruebas incluyen la prevención de errores, la reducción de los costos de desarrollo y la mejora del rendimiento.

Qué es un tester

Los **probadores de software** (también conocidos como **testers**, su denominación en inglés) planifican y llevan a cabo pruebas de software de los ordenadores para comprobar si funcionan correctamente. Identifican el riesgo de sufrir errores de un software, detectan errores y los comunican. Evalúan el funcionamiento general del software y sugieren formas de mejorarlo.

En muchos casos, **la fase del testing se ha relegado a una fase final previa a salida a producción y con un tiempo tan limitado que, en muchos casos, no pueden garantizar un testing eficaz**. Hablando de pruebas funcionales, el testing puede ser más valorable, dado a los resultados que esto ofrece (pasado o no pasado), por el contrario, otras pruebas, como rendimiento o seguridad, quedan relegadas a un punto menos cuantificable, ya que no afecta a su funcionalidad directa y por parte de negocio no suelen venir unos requisitos específicos.

El testing es clave para poder garantizar la calidad y funcionalidad de cualquier producto o servicio que se quiera ofrecer al público y el software no es una excepción. Cuando un **producto de software** no es debidamente testado, puede presentar una amplia variedad de

problemas que frustran al usuario y disminuyen su confianza en la empresa que lo ha ofrecido.

Para poder hacer un buen testing de software, se aplican **7 principios del testing** establecidos por el ISTQB -**International Software Testing Qualifications Board** es una entidad que define un esquema de certificación internacional para la **calidad del software**.

Los 7 principios del testing de software

Aunque cada empresa y cada desarrollador pueden tener sus métodos propios, el International Software Testing Qualifications Board propone **siete principios del testing** básicos a los que todo testing debería ajustarse. Se trata de una guía para homogeneizar el testing de software a nivel global y garantizar que se siguen los estándares de calidad del software.

1. El testing sirve para demostrar defectos

El testing muestra la **existencia de errores**, no su ausencia. Es muy importante partir de esta base, porque el objetivo principal de las pruebas es la identificación y resolución de errores. Esto reduce la posibilidad de que los usuarios finales encuentren errores no descubiertos. Por eso este es uno de los principales **principios del testing**.

2. No es posible realizar testing de software exhaustivo

Hay muchos tipos de pruebas de software pero el **testing exhaustivo** es imposible. Sin embargo, el desconocimiento hace que en muchos casos se pronuncie la famosa frase «¡*probadlo todo*!». Probar todas las combinaciones de un software es imposible excepto en productos extremadamente sencillos. Por eso es importante evaluar y delimitar qué y cómo se debe probar.

3. Necesidad de realizar pruebas tempranas

El tercero de los **7 principios del testing** incide en la importancia de realizar el testing en **fases tempranas** de desarrollo porque ahorra tiempo y dinero. Cuanto más se avanza en el ciclo de vida del producto, más costoso es solucionar los errores.

4. **Aglutinación de defectos**

Los defectos se pueden **aglutinar en *clusters***. Existen grupos funcionales que por su complejidad o especificidad de negocio reúnen a su alrededor la mayor parte de defectos sotfware.

5. **Paradoja del pesticida**

Se da la **paradoja del pesticida** que pierde efectividad a largo plazo: si se repiten siempre los mismos test llegará un momento en el que no se encontrarán defectos. Por eso es importante mantener y nutrir las baterías de pruebas con nuevos casos de manera regular.

6. **Hay que tener en cuenta el contexto**

El testing **depende del contexto. Las pruebas deben realizarse teniendo en cuenta el escenario, entorno y caso de uso**. ¿El usuario tendrá prisa? ¿Usará la aplicación en dispositivo móvil o escritorio? ¿La usará para sí mismo o para atender a otra persona? ¿Es una aplicación profesional o lúdica? ¿Cuántas personas utilizarán al mismo tiempo la aplicación cuando esté en producción?

7. **La ausencia de errores es una falacia**

Aunque es el sueño de cualquier usuario, programador responsable de proyecto, ¡el software siempre tiene errores! Por eso es importante tener muy presente la importancia del QA y diseñar una buena estrategia.

Tipos de pruebas de software: diferencias y ejemplos

Las pruebas de software son una parte integral del ciclo de vida del desarrollo de software (SDLC). Las pruebas son la forma en que puede estar seguro acerca de la funcionalidad, el rendimiento y la experiencia del usuario. Ya sea que realice sus pruebas manualmente o mediante automatización, cuanto antes y con mayor frecuencia pueda realizar pruebas, más probable es que identifique errores y errores, no solo salvándolo a usted y a su equipo de posibles simulacros de incendio más adelante, sino también asegurando que su aplicación de software haya sido revisada y auditada exhaustivamente antes de que esté frente a su usuarios. Si los problemas se arrastran al entorno de producción, los más caros y lentos que van a solucionar.

Las pruebas de software se pueden dividir en dos tipos diferentes: pruebas funcionales y no funcionales. Diferentes aspectos de una

aplicación de software requieren diferentes tipos de pruebas, como pruebas de rendimiento, pruebas de escalabilidad, pruebas de integración, pruebas unitarias y muchos más. Cada uno de estos tipos de pruebas de software ofrece una excelente visibilidad de la aplicación, desde el código hasta la experiencia del usuario. Vamos a entrar en los detalles de algunos de los tipos más comunes de pruebas de software.

Tipos de pruebas de software: pruebas funcionales y no funcionales

Pruebas funcionales

Las pruebas funcionales se llevan a cabo para comprobar las características críticas para el negocio, la funcionalidad y la usabilidad. Las pruebas funcionales garantizan que las características y funcionalidades del software se comportan según lo esperado sin ningún problema. Valida principalmente toda la aplicación con respecto a las especificaciones mencionadas en el documento Software Requirement Specification (SRS). Los tipos de pruebas funcionales incluyen pruebas unitarias, pruebas de interfaz, pruebas de regresión, entre otras.

Pruebas unitarias

Las pruebas unitarias se centran en probar piezas/unidades individuales de una aplicación de software al principio del SDLC. Cualquier función, procedimiento, método o módulo puede ser una unidad que se someta a pruebas unitarias para determinar su corrección y comportamiento esperado. Las pruebas unitarias son las primeras pruebas que los desarrolladores realizan durante la fase de desarrollo.

Pruebas de integración

Las pruebas de integración implican probar diferentes módulos de una aplicación de software como grupo. Una aplicación de software se compone de diferentes submódulos que trabajan juntos para diferentes funcionalidades. El propósito de las pruebas de integración es validar la integración de diferentes módulos juntos e identificar los errores y problemas relacionados con ellos.

Pruebas no funcionales

Las pruebas no funcionales son como pruebas funcionales; sin embargo, la principal diferencia es que esas funciones se prueban bajo carga para el rendimiento de los observadores, fiabilidad, usabilidad,

escalabilidad, etc. Además de las pruebas de rendimiento, los tipos de pruebas no funcionales incluyen pruebas de instalación, pruebas de confiabilidad y pruebas de seguridad.

Performance Testing

Las pruebas de rendimiento son un tipo de prueba no funcional, que se lleva a cabo para determinar la velocidad, estabilidad y escalabilidad de una aplicación de software. Como su nombre indica, el objetivo general de esta prueba es verificar el rendimiento de una aplicación contra los diferentes puntos de referencia del sistema y la red, como la utilización de la CPU, la velocidad de carga de la página, el manejo del tráfico máximo, la utilización de recursos del servidor, etc. Dentro de las pruebas de rendimiento, hay varios otros tipos de pruebas, como las pruebas de carga y las pruebas de esfuerzo.

Cómo estos tipos de prueba difieren entre sí

Es posible que tenga alguna idea sobre los diferentes tipos de pruebas anteriores. Todas las pruebas se centran en la confiabilidad y la preparación de la aplicación de software, sin embargo, entendamos mejor las diferencias entre ellas a través de algunos ejemplos. Supongamos que tiene un sitio web/aplicación de comercio electrónico con funcionalidades estándar. Estos son algunos ejemplos de pruebas de rendimiento, pruebas funcionales, pruebas de integración y pruebas unitarias:

Si desea comprobar cómo funcionará su sitio web cuando un alto número de usuarios acudan a su sitio web, por ejemplo, durante la temporada de ventas, debe realizar pruebas de carga, que entran dentro de la categoría de pruebas de **rendimiento.** Le ayudará a detectar problemas de velocidad y estabilidad y eliminar posibles cuellos de botella de rendimiento.

Supongamos que desea validar la entrada y salida para cada funcionalidad, como registro, inicio de sesión, agregar al carrito, pago, procesamiento de pagos, entradas de base de datos, etc., de acuerdo con los casos de prueba escritos en el documento SRS. En ese caso, debe realizar **pruebas funcionales.**

Si desea validar la funcionalidad del carrito con la integración del módulo de pago y pago para ver si el número de artículos agregados al

carrito se compra correctamente con el pago correcto, debe realizar **pruebas de integración.**

Si ha escrito un módulo para la carga del producto y desea comprobar si es correcto y los productos se agregan correctamente sin ningún error o defecto, debe realizar **pruebas unitarias** para el módulo de carga del producto.

En resumen, se realizan pruebas de rendimiento para verificar el rendimiento del sitio web. Las pruebas funcionales se realizan para validar todas las funcionalidades. Las pruebas de integración se realizan para validar la interacción entre diferentes módulos, y se realizan pruebas unitarias para comprobar si son correctos las piezas de código individuales.

Ventajas de estos tipos de prueba

Performance Testing

- Evalúa la velocidad y escalabilidad del sitio web/aplicación.
- Identifica los cuellos de botella para las mejoras de rendimiento.
- Detecta errores que se pasan por alto en las pruebas funcionales.
- Optimización del sistema y mejoras de características
- Garantiza la fiabilidad del sitio web bajo una gran carga.

Pruebas funcionales

- Se asegura de que el sitio web / aplicación está libre de defectos.
- Garantiza el comportamiento esperado de todas las funcionalidades.
- Garantiza que la arquitectura sea correcta con la seguridad necesaria.
- Mejora la calidad y las funcionalidades generales.
- Minimiza los riesgos empresariales asociados con el sitio web/aplicación.

Pruebas de integración

- Se asegura de que todos los módulos de aplicación estén bien integrados y funcionen juntos según lo esperado.
- Detecta problemas y conflictos interconectados para resolverlos antes de crear un gran problema.
- Valida la funcionalidad, fiabilidad y estabilidad entre diferentes módulos.

- Detecta excepciones ignoradas para mejorar la calidad del código.
- Admite la canalización de CI/CD.

Pruebas unitarias

- Detección temprana de errores en las nuevas funcionalidades o características desarrolladas.
- Minimiza los costos de las pruebas a medida que se detectan problemas desde el principio.
- Mejora la calidad del código con una mejor refactorización del código.
- Apoya el proceso de desarrollo ágil.
- Simplifica la integración y permite una buena documentación.

Desventajas de estos tipos de pruebas

Como todos estos tipos de prueba mejoran las funcionalidades y mejoran la experiencia del usuario, no hay desventajas al hacerlo. Lo único que puede considerar una desventaja, en general, es el tiempo y el costo asociados con la prueba. Las pruebas requieren esfuerzos y recursos, y existe un riesgo relacionado con resultados de pruebas inexactos. Sin embargo, no hacer pruebas de sitio web / aplicación le pondrá en una posición comprometedora que puede obstaculizar su negocio y reputación significativamente.

El momento adecuado para realizar este tipo de pruebas

Las pruebas de rendimiento son imprescindibles en todos los entornos de desarrollo y producción para garantizar que su sitio web o aplicación esté al día y pueda soportar la carga de usuario esperada. Las pruebas funcionales deben realizarse con cada compilación para validar todos los cambios y funcionalidades con respecto a las especificaciones y requisitos. Las pruebas de integración deben realizarse al integrar un nuevo fragmento de código con algún otro módulo para asegurarse de que no hay conflictos y trabajar juntos correctamente. Las pruebas unitarias deben realizarlas siempre que terminen de escribir cualquier código para validar la entrada y la salida correctas.

Consejo: Un enfoque de mano en mano

Aunque cada tipo de prueba parece una tarea independiente, puede combinarlas de forma inteligente para lograr una mayor calidad del producto. Tomemos un ejemplo.

Supongamos que ha creado una nueva página web, ejecutando una prueba de carga (prueba de rendimiento) para esa página web como prueba unitaria se asegurará de que cuando realice su compilación final con todas las páginas, el sitio web ya esté optimizado para manejar una alta carga de usuarios en escenarios de tráfico pico. Esto significa que tiene su rendimiento probando una parte de las pruebas unitarias. Un enfoque mano a mano como este le ayudará a reducir los problemas en una etapa temprana y le ahorrará una gran cantidad de costo y tiempo a largo plazo.

Tomado de:

Los 7 principios del testing. ¿Qué dice ISTQB? - Blog de Hiberus Tecnología

Testing, la importancia sobre la fase de testeo de software - Blog de Hiberus Tecnología

Tipos de pruebas de software: diferencias y ejemplos - LoadView (loadview-testing.com)