Tipos de pruebas

💡 Las pruebas de software son una parte integral del ciclo de vida de desarrollo de software **(SDLC)**. Las pruebas brindan confianza en la funcionalidad, el rendimiento y la experiencia del usuario. Ya sea que las pruebas se realicen de forma manual o automática, cuanto antes y con más frecuencia se realicen, más probable será identificar fallas y errores, debido a que una vez el problema llega a producción, se vuelve más costoso y lleva más tiempo solucionarlo.

▼ Pruebas funcionales

💡 Las pruebas funcionales se definen en función de los requisitos del sistema, estas pruebas se utilizan para confirmar y garantizar que el producto cumple con las especificaciones, que hace lo que se supone que debe hacer y cómo se supone que debe hacerlo, y da una idea de la calidad del software.

▼ Pruebas unitarias

Las pruebas unitarias aseguran que todas las líneas de código desarrolladas en un componente proporcionen resultados adecuados. En estas pruebas, el desarrollador observa la interfaz y la especificación del componente por separado, siempre que la documentación de desarrollo del código se haya probado exhaustivamente antes de pasar a otra unidad.

Las pruebas unitarias respaldan las pruebas funcionales ejercitando el código que es más probable que se rompa. Por lo tanto, el uso de pruebas funcionales sin pruebas unitarias puede causar problemas en el diagnóstico de pruebas fallidas, así que deben ser tenidas en cuenta.

▼ Prueba de componentes

Las pruebas de componentes se ejecutan por separado para garantizar que los resultados cumplan con lo requerido. Su objetivo es validar la funcionalidad y usabilidad del componente, pero no se limita solo a esto.

Un ejemplo de prueba de componente podría ser cualquier elemento que tenga una entrada y esté destinado a producir una salida.

Un componente puede ser un módulo de código, una página web, una pantalla o incluso un sistema dentro de un sistema más grande.

Estos son algunos usos del componente que se pueden probar:

- **Prueba de UI:** Para usabilidad y accesibilidad
- **Prueba de carga:** Para asegurar el rendimiento
- **Inyección de SQL a través de componentes de UI:** Para asegurar la seguridad
- **Prueba de login:** Con credenciales válidas e inválidas

▼ Prueba de humo

Se ejecuta una prueba de humo para verificar si la funcionalidad principal de la aplicación está funcionando. Esta prueba asegura de que el software más básico funcione correctamente con pruebas simples y rápidas.

Esta es una de las pruebas funcionales más importantes y debería ser lo primero en una nueva compilación. Las pruebas de humo se usan ampliamente y no se tratan de realizar pruebas exhaustivas, sino que se trata de asegurar de que las funciones críticas del sistema funcionan realmente bien y si pasan las pruebas, su compilación es estable.

Según la situación, el equipo de control de calidad realizará posteriormente pruebas funcionales o de regresión de la funcionalidad recién agregada. Por otro lado, si no es estable y no se puede compilar, por lo general se devuelve al

equipo de desarrollo para solucionar los problemas de compilación y crear uno nuevo.

▼ Prueba de integración

Las pruebas de integración son uno de los tipos más comunes de pruebas funcionales y están automatizadas. Son para probar componentes individuales y verificar cómo funcionan los módulos que trabajan individualmente cuando están integrados.

El propósito de realizar estas pruebas es mantener a los desarrolladores enfocados en construir diferentes módulos del sistema al mismo tiempo y no en otros módulos.

Las pruebas de integración permiten el flujo de datos y comandos operativos entre módulos, haciendo que todo funcione como parte de un solo sistema en lugar de aplicaciones aisladas.

Por lo general, ayuda a identificar problemas como las operaciones de la interfaz de usuario, los formatos de datos, las llamadas API, el acceso a la base de datos, etc.

Algunas de las comprobaciones realizadas durante las pruebas de integración son:

- **Prueba de interfaz:** Se prueba la transferencia de datos entre dos componentes, asi como tambien interfaces como servicios web, API, etc. Se ejecuta para garantizar que los componentes estén sincronizados entre sí. Esto ayuda a determinar varias funciones, como la transferencia de datos entre diferentes elementos del sistema y se realizan de acuerdo con la forma en que fueron diseñados.

▼ Prueba de regresión

Es normal que en el ciclo de desarrollo se cambien, arreglen y mejoren funcionalidades. Por lo tanto, existe una alta posibilidad de que estas acciones tenga un "efecto" inesperado.

Las pruebas de regresión se realizan para garantizar que los cambios o adiciones no hayan alterado o eliminado la funcionalidad existente.

El propósito de las pruebas de regresión es encontrar errores que pueden haberse introducido inadvertidamente en compilaciones existentes y así evitar la creación de nuevos errores y en la reapertura de antiguos errores.

▼ Prueba de cordura

Las pruebas de cordura se realizan para verificar que una compilación de software funciona correctamente con todos los cambios de código realizados en el software.

El objetivo principal de una prueba de cordura no es una prueba detallada sobre su aplicación, sino una prueba estrecha y profunda que verifica características específicas y correcciones de errores de su aplicación (pruebas de que los cambios en su código no han introducido nuevos errores).

Es importante no confundir las pruebas de humo con las pruebas de cordura. El objetivo de las pruebas de humo es confirmar la estabilidad del producto y el de las pruebas de cordura son enfocarse en la funcionalidad planificada y la correccion de errores.

▼ Pruebas de aceptación del usuario

Cuando ya se hayan realizado las pruebas requeridas en el producto, el siguiente paso es realizar las pruebas de aceptación. Estas son parte de las etapas finales de este proceso de prueba. Aquí es donde los usuarios reales del software se aseguran de utilizarlo para realizar las tareas que desean en un entorno "real". Esto se puede hacer en la entrega del producto "como punto de control final entre todos los tipos de pruebas funcionales".

▼ Pruebas no funcionales

Las pruebas de software no funcionales son aquellas que se realizan desde una perspectiva completamente diferente a las pruebas automatizadas. Este tipo de plan de pruebas es una herramienta de control de calidad que se realiza en las aplicaciones de software para asegurarse de que todo funciona bien y saber en qué condiciones pueden fallar.

▼ Pruebas de carga

Estas pruebas se ejecutan para determinar y validar la respuesta de una aplicación cuando se somete a una cantidad específica de usuarios o carga de solicitudes.

Ejemplo: Comprobar si el producto puede soportar una carga de 100 usuarios simultáneamente. Este resultado se compara con el volumen esperado.

▼ Pruebas de rendimiento

El objetivo principal de este tipo de prueba no funcional es calcular la respuesta de la aplicación a varias acciones o solicitudes del usuario.

Ejemplo: Saber la respuesta cuando los parámetros son 10, 100 y 1000 entradas de usuario. Este resultado se compara con el resultado esperado.

▼ Pruebas de estrés

Estas pruebas se realizan para determinar los tiempos soportados por los usuarios, las solicitudes o la aplicación. Este tipo de pruebas no funcionales es muy similar a las pruebas de carga y rendimiento, pero se diferencia en que tenemos que superar los límites esperados en el entorno de producción o definidos en las pruebas.

Ejemplo: Encontrar la cantidad de usuarios que puede admitir simultáneamente hasta que la aplicación deje de responder (se congela o se agota el tiempo de espera) haciéndolo correctamente para todas las solicitudes.