

Casos de Prueba: Estrategia y análisis

Introducción

En un contexto ideal, se desearía probar cada permutación posible de un programa, pero la complejidad hace que esto sea económicamente inviable. El problema influye en la economía de las pruebas, las suposiciones del tester y el diseño de casos de prueba. El objetivo es maximizar la eficacia de las pruebas para encontrar el mayor número de errores con un número finito de casos. La selección de casos de prueba es crucial, considerando la revisión regular y la creación de nuevas pruebas.

Cada caso de prueba debe ser preciso, apuntar a un propósito específico y ser lo suficientemente claro para que cualquier persona pueda ejecutarlo sin necesidad de conocer el sistema a fondo.

Confección del caso de Prueba



La confección de un caso de prueba, sin importar el tipo del que sea, es un proceso detallado y estratégico que implica la creación de un documento

preciso y completo para evaluar la funcionalidad de un componente específico del software. A continuación , revisaremos los elementos esenciales para garantizar la redacción de un caso de prueba efectivo. Recuerda que no todos los equipos redactan los casos de prueba del mismo modo, pero sí es importante mantener el formato y criterio a lo largo de todo el proceso de pruebas del software.

- **Identificador:** Código único para vincular el caso de prueba con informes de errores u otras referencias.
- **Conjunto de Prueba:** Identificador de los conjuntos de prueba donde se utiliza el caso de prueba.
- **Nombre de la Funcionalidad:** Descripción de la funcionalidad del Inventario de Prueba (IP) que se evalúa.
- **Prioridad:** Asignación de prioridad según escala utilizada (1-alta, 2-medio, 3-baja) al caso de prueba. Íntimamente ligada a la severidad del caso.
- **Requisitos:** Características del objeto de pruebas que deben evaluarse.
- **Necesidades de Ambientes:** Indicación del ambiente de prueba necesario para la ejecución.
- **Situación:** Registro de condiciones especiales que podrían afectar la ejecución o resultados del caso de prueba.
- **Marca de Tiempo y Evaluador:** Inclusión de marca de tiempo y nombre del evaluador en los comentarios para atribuir resultados a un momento específico.
- **HW** (Hardware): Lista de hardware requerido para ejecutar el caso de prueba.
- **SW** (Software): Lista de software necesario para ejecutar el caso de prueba.
- **Configuración** (Setup): Pasos necesarios para preparar el entorno de prueba.
- **Retroceder Acciones** (Cleanup): Pasos para revertir el entorno al estado previo a la prueba.
- **Precondiciones:** Situación inicial o acciones previas necesarias para la ejecución de pruebas.
- **Resultado Esperado:** Especificación clara del resultado esperado después de ejecutar las pruebas.

- **Dependencias:** Identificación y razón de las dependencias entre casos de prueba.
- **Valores de Entrada / Salida:** Descripción detallada de los datos utilizados en el caso de prueba, con claridad y sin ambigüedades.

Los siguientes datos se completan una vez que el caso de prueba se ejecutó:

- **Resultado real:** Especificación de lo que realmente ocurrió: Pasó/Falló/No se pudo ejecutar
- **Post-caso de prueba condiciones:** características de un objeto de prueba tras la ejecución de pruebas, descripción de su situación tras la ejecución de las pruebas. Puede ser un dibujo o texto explicando cómo se espera que los datos iniciales queden luego de ejecutar la acción. Debe ser claro y no dejar dudas.

Cada vez que se ejecuta un caso debe registrarse la información resultante para que quien sea responsable del código probado, sepa si está correcto o se produjo algún error o resultado no deseado.

Aquí se obtiene un resultado real de la ejecución del código que denominamos Resultado Final de la Ejecución del Caso de Prueba.

Las ejecuciones de casos se agrupan en "Corridas" que se deben identificar para saber cuándo (y a veces quién) realizó una ejecución.

Resultados de casos de prueba

Los resultados de los casos de prueba se dividen en dos categorías fundamentales: positivos y negativos.

1. Casos de Prueba Positivos: "El Camino Feliz"

Los casos de prueba positivos se conocen como "el camino feliz". Estos casos verifican el correcto funcionamiento del uso principal y esperado de una funcionalidad. Tomando como ejemplo la funcionalidad de inicio de sesión en un sistema, el camino feliz implicaría ingresar un nombre de usuario válido y la contraseña correspondiente. Estos casos confirman que la funcionalidad

principal opera según lo esperado, representando el escenario ideal de interacción.

2. Casos de Prueba Negativos: Caminos Alternativos

Contrariamente, los casos de prueba negativos abordan situaciones fuera del escenario ideal, explorando caminos alternativos y posibles errores. En el ejemplo del inicio de sesión, los casos negativos se manifiestan cuando se intenta ingresar con un nombre de usuario no registrado o con una contraseña incorrecta. Estos casos no siguen el "camino feliz" y evalúan la capacidad del sistema para manejar situaciones inesperadas o ingresos incorrectos.

Es esencial destacar que, en la categoría de casos negativos, una palabra clave fundamental es "INTENTAR". Esta palabra se emplea comúnmente para describir acciones en los casos de prueba negativos, subrayando que la acción descrita no debería ser posible de realizar. Utilizar la palabra "INTENTAR" proporciona claridad sobre la naturaleza de la prueba, indicando que se está evaluando cómo el sistema responde ante acciones que no deberían ser permitidas.

Mantener un enfoque BAOE

Mantener el enfoque de **BAOE** (**B**ásico, **A**lternativo, **O**pciones y **E**xcepciones) es crucial para lograr una cobertura exhaustiva del flujo funcional y de las características que se están probando. Este método garantiza una evaluación detallada tanto en situaciones estándar como en escenarios más complejos. A continuación, se detallan los elementos fundamentales de este enfoque, aplicables tanto a pruebas positivas como negativas:

- **Flujo Básico:** Se enfoca en la ruta principal del proceso, evaluando el comportamiento del sistema bajo condiciones normales. Por ejemplo, al analizar la pantalla de inicio de sesión, el flujo básico sería: "Ingrese la ruta URL del inicio de sesión en cualquier navegador, complete la información requerida e inicie sesión en la aplicación".
- **Flujo Alternativo:** Explora las trayectorias no convencionales o secundarias que los usuarios pueden seguir. Al probar estos caminos menos frecuentes, se verifica la capacidad del sistema para gestionar situaciones menos comunes. Siguiendo el ejemplo de la pantalla de inicio de sesión,

un flujo alternativo podría ser: "Instale la aplicación en un dispositivo móvil, complete la información requerida e inicie sesión en la aplicación".

- **Opciones:** Se centra en las diversas opciones y configuraciones disponibles en la funcionalidad, incluyendo evaluación de diferentes configuraciones y elecciones que los usuarios pueden realizar durante el uso regular del sistema. Para la pantalla de inicio de sesión, estas opciones podrían incluir: "¿Cuáles son las opciones disponibles para llegar a la misma pantalla de inicio de sesión? Por ejemplo, después de iniciar sesión, al hacer clic en 'Cerrar sesión', puede aparecer la misma pantalla o, si el tiempo de espera de la sesión expira, el usuario puede acceder a la pantalla de inicio de sesión".
- **Excepciones:** Examina cómo el sistema maneja situaciones inesperadas o errores, evaluando la capacidad del software para identificar y responder adecuadamente a condiciones excepcionales. Este enfoque asegura una experiencia de usuario robusta incluso en circunstancias imprevistas. Ejemplo si se ingresan credenciales incorrectas en la pantalla de inicio de sesión, si el usuario recibirá un mensaje de error o ninguna acción asociada.

Gestión de trazabilidad

Es imperativo que las pruebas sean trazables, lo que implica conocer qué casos de prueba han sido incorporados en el catálogo o listado, y sobre qué requisitos están fundamentados. Este enfoque facilita la identificación directa de las repercusiones que los cambios en los requisitos pueden tener sobre las pruebas planificadas.

La trazabilidad no sólo proporciona un registro claro de la relación entre casos de prueba y requisitos, sino que también se erige como una herramienta invaluable para evaluar la cobertura de requisitos. Al seguir la trazabilidad, se asegura que cada requisito sea respaldado por pruebas adecuadas, brindando una visión completa y detallada de la calidad y validez del software en desarrollo.

Controlar longitud para mejorar la comprobación

La longitud de los casos de prueba desempeña un papel crucial en la eficacia y precisión de la evaluación. Se recomienda que un caso de prueba ideal tenga entre 8 y 16 pasos, siguiendo el enfoque paso a paso.

Mantener los casos de prueba cortos presenta diversas ventajas, como la reducción del tiempo necesario, una menor probabilidad de errores y una menor dependencia de la asistencia o riesgos de pérdida de datos. La longitud de los casos de prueba también permite estimar con precisión el tiempo y el esfuerzo requeridos para la prueba, así como prever sus resultados.

Para casos de prueba en formato de matriz, se sugiere una longitud óptima que oscile entre 18 y 20 minutos, proporcionando un equilibrio efectivo entre exhaustividad y eficiencia en la evaluación.

Desafíos comunes en creación de casos de Prueba

En el ámbito de la calidad del software, abordar los desafíos en la redacción de casos de prueba es crucial. A continuación, exploraremos algunos problemas comunes surgidos durante el proceso:

➤ Pasos Compuestos:

La complejidad surge cuando los pasos de prueba se vuelven compuestos, es decir, se pueden desglosar en varias acciones individuales. Por ejemplo, instrucciones como "Ve al lugar XYZ y luego al ABC" pueden resultar confusas, ya que los usuarios se preguntarán: "¿Cómo llego a XYZ en primer lugar?". En el ámbito de las pruebas, esto se traduce en la necesidad de especificar claramente "cómo" realizar cada paso.

Por ejemplo, al redactar una prueba para Amazon.com, en lugar de tener un paso compuesto en el pago, como "Checkout y pago", es más efectivo dividirlo en pasos individuales, detallando cada acción, desde buscar un producto hasta la confirmación del pedido.

Ejemplo de Caso Anterior:

- Dirigirse a: Amazon.com
- Buscar un producto ingresando la palabra clave / nombre del producto en el campo 'Buscar'.
- Elegir el primer resultado.
- Hacer clic en "Agregar al carrito" en la página de detalles del producto.
- Hacer clic en "Pagar" en la página del carrito de compras.
- Ingresar la información de CC, envío y facturación.
- Hacer clic en "Pagar".
- Consultar la página de confirmación del pedido.

La ausencia de información documentada clara es un desafío común en proyectos actuales, lo cual obliga al equipo a depender del comportamiento de la aplicación, incluso en versiones anteriores, para la creación y ejecución de pruebas. Este escenario se presenta con mayor frecuencia en entornos de programación extrema y ciclos de desarrollo rápidos. En estas circunstancias, la toma del comportamiento de la aplicación como el esperado se convierte en una estrategia esencial para mantener la agilidad del equipo y garantizar la calidad del software.

- **Varias condiciones en un caso:**

Analizemos el siguiente ejemplo, de casos de prueba creados para un sencillo "Inicio de Sesión":

1. Ingrese detalles válidos y haga clic en Enviar.
2. Deje el campo Nombre de usuario vacío. Haga clic en Enviar.
3. Deje el campo de la contraseña vacío y haga clic en Enviar.
4. Elija un nombre de usuario / contraseña que ya haya iniciado sesión y haga clic en Enviar.

Supongamos, que deseamos unir estos 4 casos de prueba en uno solo, ¿trae algún tipo de problema?

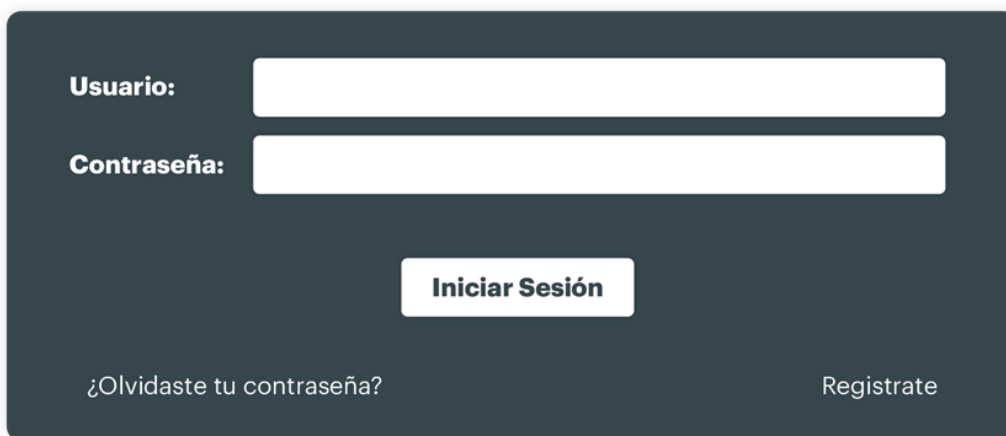
A menudo, nos enfrentamos a la tentación de combinar múltiples condiciones en un solo caso de prueba, buscando ahorrar tiempo y documentación. Sin embargo, este enfoque puede resultar problemático, ya que cada condición

debería representar un caso distinto. La complicación surge cuando una de estas condiciones falla, obligando a marcar toda la prueba como 'fallida', aunque no todas las condiciones estén afectadas.

La importancia de mantener un flujo coherente en las pruebas se evidencia al considerar el caso de inicio de sesión. Si, por ejemplo, el paso A tiene éxito y el tester inicia sesión, el siguiente paso (paso B) podría perder su contexto, ya que la opción 'iniciar sesión' ya no estaría disponible.

La clave para abordar esto radica en la escritura de pruebas modulares. Aunque puede parecer más laborioso, separar las condiciones en casos individuales garantiza un flujo continuo y preciso. Utilizar herramientas como Ctrl + C y Ctrl + V puede facilitar enormemente este proceso.

Para ilustrar este enfoque, consideremos cómo escribir casos de prueba eficientes para una pantalla de 'Inicio de sesión'. Ya sea para una interfaz simple o una más compleja, el procedimiento de prueba sigue siendo similar.



The image shows a dark-themed login form. It contains two input fields: 'Usuario:' and 'Contraseña:'. Below these fields is a button labeled 'Iniciar Sesión'. At the bottom left, there is a link '¿Olvidaste tu contraseña?' and at the bottom right, a link 'Registrate'.

El primer paso es obtener un prototipo de pantalla, si está disponible. Aunque no siempre es posible, si hay un documento de Especificación de Requisitos de Software (SRS), este puede proporcionar valiosos prototipos desarrollados por el equipo del proyecto. Estos prototipos simplifican la tarea del evaluador y mejoran la eficiencia de las pruebas.

Luego, se analizan las especificaciones de requisitos funcionales. Dependiendo de la organización, estas pueden estar en varios documentos. La elección del

mejor documento para redactar casos dependerá de la disponibilidad y claridad del flujo funcional.

Con el prototipo y las especificaciones funcionales en su lugar, el tester puede comenzar a escribir casos de prueba, teniendo en cuenta los 8 tipos de casos de prueba para un enfoque integral y efectivo. Este método garantiza pruebas más precisas y facilita la adaptabilidad a cambios en condiciones específicas.

➤ **Falta de recopilación de datos de prueba:**

En la redacción de casos de prueba, la tarea primordial para cualquier evaluador radica en la recopilación de datos de prueba. A menudo, este paso crucial es pasado por alto, con la suposición errónea de que los casos de prueba pueden ejecutarse con datos de muestra o ficticios y que los datos necesarios se pueden proporcionar durante la ejecución. Este malentendido conlleva a la alimentación de datos desde la memoria en el momento de ejecutar casos de prueba, un enfoque altamente problemático.

Si los datos no se recopilan y actualizan en el documento de prueba al escribir los casos, el tester enfrentará desafíos significativos al recopilar los datos en el momento de la ejecución. La recopilación de datos de prueba es esencial tanto para casos positivos como negativos, asegurando una perspectiva completa del flujo funcional de la característica.

Esto no solo optimiza el proceso de prueba, sino que también reduce la probabilidad de errores y ahorra tiempo valioso durante la ejecución. La meticulosa recopilación de datos establece una base sólida para pruebas eficientes y efectivas.

➤ **Trabajar con una única condición de prueba:**

Es esencial que los casos de prueba aborden una gama diversa de condiciones que el software pueda encontrar. Cada caso debe ser capaz de evaluar de manera exhaustiva el módulo de software, considerando prácticamente todas las combinaciones posibles de condiciones principales.

Para lograr una prueba exhaustiva, el probador de software debe desarrollar un enfoque que permita presentar estas condiciones de manera clara y comprensible. Esto facilitará la revisión, seguimiento y, si es necesario, la modificación de las pruebas por parte de otros miembros del equipo. Este enfoque se vuelve crucial para afrontar situaciones del mundo real que demandan acciones específicas y adaptabilidad en el proceso de prueba.

➤ **Cubrir una pequeña parte de la funcionalidad:**

Es común que los casos de prueba se centren en funciones específicas, a menudo dictadas por el diseño técnico interno del software. Este enfoque es especialmente evidente en aplicaciones monolíticas extensas como SAP u Oracle ERP, donde los probadores pueden carecer del conocimiento integral del proceso comercial. Esto lleva a que los casos de prueba no reflejen completamente lo que el diseñador de pruebas no conoce, pero debería esforzarse por entender.

La estrategia más efectiva implica que los casos de prueba se orienten hacia patrones y flujos de uso. En lugar de centrarse exclusivamente en funciones técnicas internas, cada caso de prueba debe esforzarse por abarcar la mayor parte del flujo operativo posible. Este enfoque trasciende los límites técnicos y modulares de la aplicación, brindando una cobertura más completa y realista.

➤ **Cobertura de un rol específico:**

Con frecuencia, observamos la redacción de casos de prueba centrados en un rol de usuario particular, excluyendo a otros usuarios de la aplicación. Esta práctica restringe el alcance de los casos de prueba, comprometiendo su efectividad de manera significativa. Estos casos de prueba, aunque pretenden ser completos, prueban solo pequeños aspectos de la aplicación de manera engañosa.

➤ **Escrito para demostrar que los casos de uso más comunes están bien cubiertos en la aplicación:**

Abordar de manera efectiva los casos de uso más comunes va más allá de simplemente convertir los requisitos en casos de prueba. Este enfoque, que califico como "perezoso", es uno de los problemas más recurrentes en el diseño de pruebas.

El diseñador de pruebas debe adoptar una estrategia más proactiva, buscando los "casos de esquina" o las "condiciones de contorno". Mientras la mayoría de los desarrolladores puede manejar los casos de uso convencionales, las dificultades emergen cuando se enfrentan a condiciones ligeramente diferentes o inesperadas. Un caso de prueba bien diseñado debe ser capaz de detectar estas variaciones de manera fácil y consistente, proporcionando así una cobertura más robusta y efectiva de los casos de uso en la aplicación.

Tipos de Mantenimiento

Cuando un usuario identifica un problema en el sistema, comunica la incidencia al administrador, quien inicia el proceso de diagnóstico para determinar el tipo de mantenimiento requerido. Los tipos de mantenimiento pueden categorizarse de la siguiente manera:

- a. **Correctivo:** Enfocado en corregir errores específicos del producto de software.
- b. **Evolutivo:** Incluye adiciones, modificaciones o eliminaciones necesarias para adaptarse a cambios o expansiones en los requisitos del usuario.
- c. **Adaptativo:** Modificaciones relacionadas con los entornos operativos, como cambios en configuraciones de hardware, software base, gestores de base de datos, comunicaciones, entre otros.
- d. **Perfectivo:** Acciones destinadas a mejorar la calidad interna del sistema, abordando aspectos como la reestructuración de código, definición más clara del sistema y optimización del rendimiento.

Una vez identificado el tipo de mantenimiento, se establece un plazo razonable para la modificación y prueba, informando al usuario. En el caso de mantenimiento correctivo, se verifica y reproduce el problema, evaluando la viabilidad de los cambios propuestos. El plazo y urgencia de la solución se determinan según el tipo de mantenimiento.

Las tareas necesarias se definen según los componentes del sistema afectados, abarcando actividades de análisis, diseño, desarrollo y pruebas. Finalmente, antes de la aceptación del usuario, se establece un plan de pruebas de regresión para garantizar la integración del sistema de información afectado.

Revisiones periódicas

La vigilancia continua del sistema es esencial para la identificación y satisfacción oportunas de las necesidades de mantenimiento. En situaciones donde el sistema tiene una vida útil prolongada, se puede establecer un mecanismo efectivo para recopilar retroalimentación de los usuarios, facilitando así la identificación de requerimientos de mantenimiento y modificaciones.

El mantenimiento de los sistemas es crucial para garantizar que continúen operando al nivel demostrado durante la etapa de prueba. El deterioro de los sistemas conlleva el riesgo de un rendimiento por debajo de los estándares requeridos, destacando la importancia de intervenciones oportunas para mantener su funcionalidad óptima.

¿Buscas un Ejemplo? Te ofrecemos [acceso a un documento](#) que presenta los casos de prueba para la pantalla de inicio de sesión. Este documento se destaca por su formato completo de cobertura y trazabilidad, proporcionando información detallada. La secuencia lógica y la numeración identificativa, conocida como 'ID de caso de prueba', resultarán especialmente valiosas para una rápida identificación en el historial de ejecución de casos de prueba.