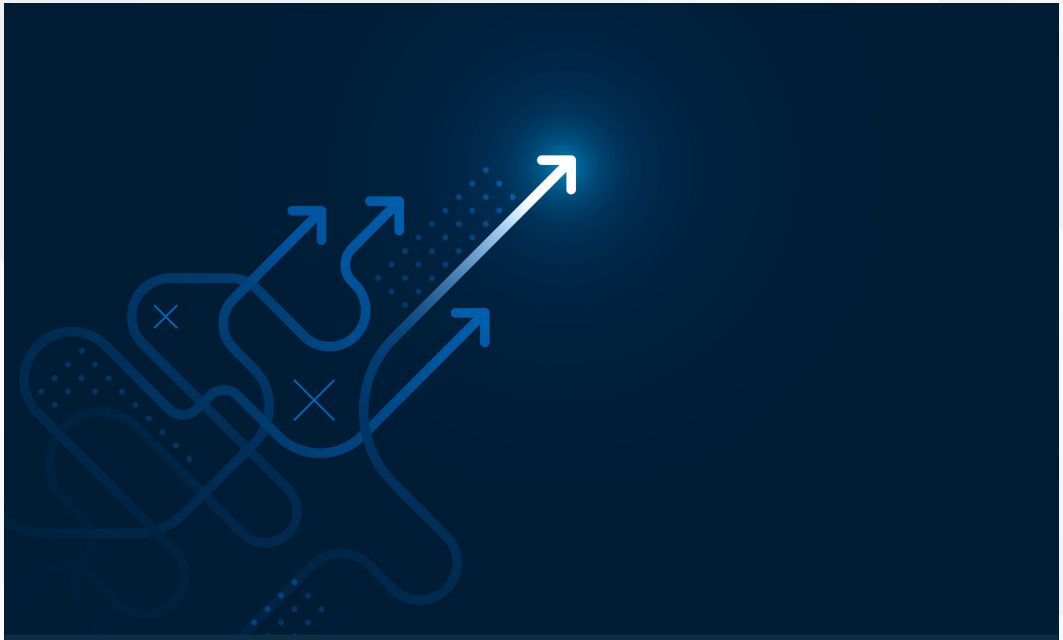

Universidad
Tecmilenio®

Universidad Tecmilenio

Software Testing

Diseños de casos de prueba

Semana 2



Vas a desarrollar tu primer software y es necesario que definas los casos de prueba, por lo cual, le pedirás a alguien que simule ser el usuario para que evalúe si el software funciona como debe y si cumple con todas las características.

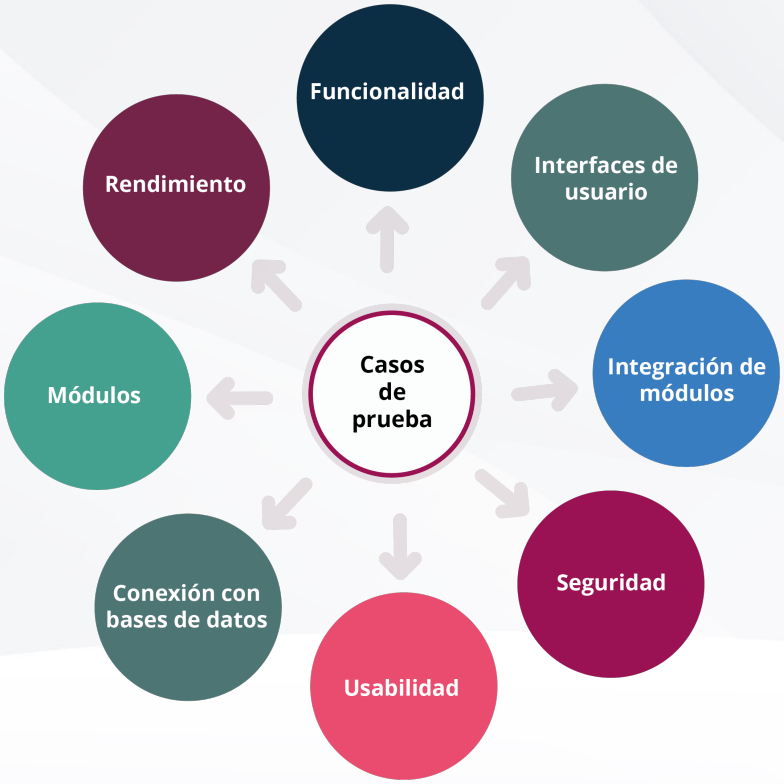
Para cada desarrollo, deberás crear casos de prueba diferentes, ya que cada uno tiene su propia complejidad. Los casos te permitirán encontrar de forma rápida los errores, no solo de codificación, sino también factores, como interfaces de usuario, seguridad, usabilidad, etc.

Explicación

¿Qué son los casos de prueba y cómo se estructuran?

Son escenarios que permiten medir la funcionalidad del desarrollo, es decir, son ciertas acciones que aseguran que este haga lo esperado.

Es importante no confundir un caso de prueba con un *script* de prueba; el script de prueba es un programa que prueba la funcionalidad de cierta parte del desarrollo, y el caso de prueba es un documento extenso que contiene lo que se debe hacer paso a paso.



Explicación

Diseño de un caso de prueba

- Id de prueba.
- Descripción de la prueba.
- Supuestos y condiciones previas a la prueba.
- Datos de la prueba.
- Pasos a ejecutar.
- Resultado esperado.
- Resultado obtenido.

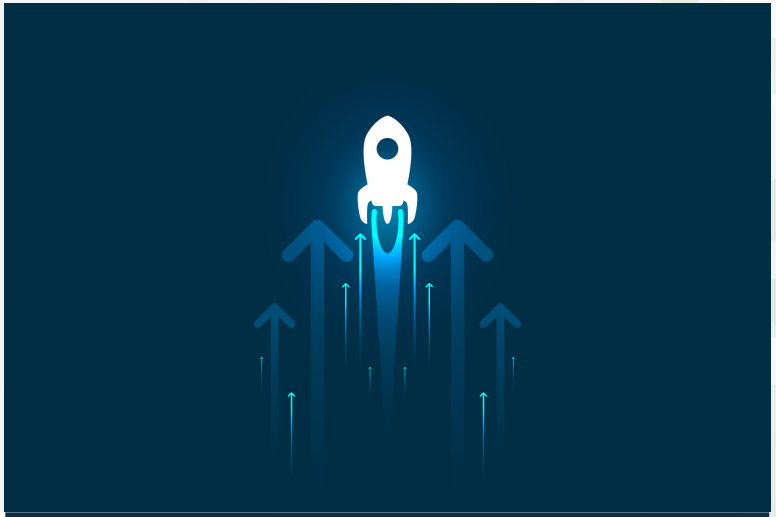
Universidad Tecmilenio

Explicación

Buenas prácticas y enfoques en el diseño

- Simples y claros: necesitarás que cualquier persona pueda entenderlos.
- Los casos de prueba no deben tener más de 15 pasos a ejecutar.
- Realiza los casos de prueba pensando en el usuario final.
- No repitas los casos de prueba para probar diferentes cosas.
- Asegúrate de que el conjunto de casos de prueba verifique todo lo relacionado con el software.
- Siempre coloca un ID a los casos de prueba para identificarlos fácilmente en caso de que lo requieras.
- Independientemente de quién ejecute el caso de prueba, los resultados deben ser siempre los mismos.
- Al finalizar la creación de los casos, es recomendable que alguien más los revise.

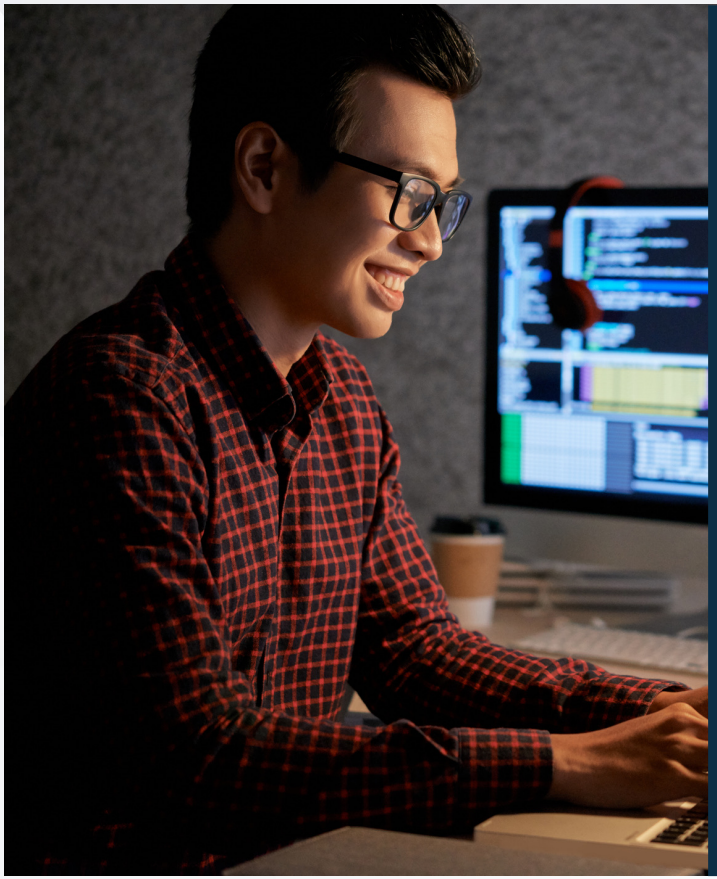
Actividad



¿Consideras que los casos de prueba son útiles?

¿Qué es más fácil: realizar el código y luego los casos de prueba o viceversa?

Cierre



Gracias a los casos de prueba podrás verificar no solo la funcionalidad del software, sino todas las siguientes características:

- Funcionalidad.
- Interfaces de usuario.
- Integración de módulos.
- Seguridad.
- Usabilidad.
- Conexión con bases de datos.
- Módulos o unidades.
- Rendimiento del desarrollo.

Para que los casos de prueba estén bien realizados, toma en cuenta todo lo que contienen, al igual que las buenas prácticas mencionadas anteriormente.

Universidad Tecmilenio

Software Testing

Administración de defectos

Semana 2

Conforme vas aplicando el ciclo de vida del software, puedes reflexionar sobre lo fundamental que es la detección de los defectos, ya que, sin esto, el desarrollo podría retrasarse, tu reputación se vería afectada, la inversión para el desarrollo sería mayor y perderías la confianza del cliente.

Explicación

¿Qué es un defecto y cómo se estructura?

Error

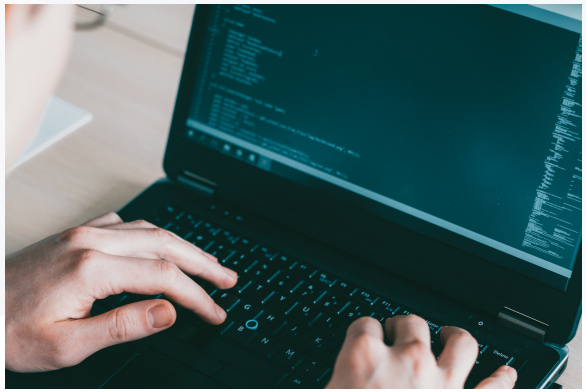
Algo que ocurre debido a una acción humana.

Fallo

Cuando ejecutas el software y se presenta un resultado no deseado debido a que el software tiene un defecto.

Defecto

Ocurre debido a un error de implementación.



Explicación

¿Para detectar los defectos, utilizarás las pruebas estáticas, ya que las dinámicas se centran en encontrar fallos.

Una vez que el *tester* haya detectado el defecto, deberás llenar el documento de defectos, el cual contiene toda la información del defecto para que el equipo de desarrollo pueda corregirlo. Su estructura es la siguiente:

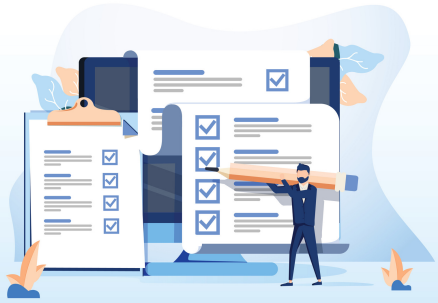
Elemento	Definición
ID del defecto	Se coloca para que sea más fácil de identificar el defecto del que se está hablando.
Nombre del defecto	Se coloca el nombre del defecto y una descripción a detalle sobre lo que está pasando.
Ruta de área	En qué módulo o en qué parte del software se encontró el defecto.
Número de compilación	Versión del software donde se encontró el defecto.
Gravedad	Puede ser clasificada como alta, media o baja.
Prioridad	Puede ser clasificada como alta, media o baja.

Universidad Tecmilenio

Explicación

Estructura del documento de defectos

Elemento	Definición
Asignado a	El desarrollador será el encargado de solucionar el defecto.
Reportado por	Nombre del tester que encontró el defecto.
Reportado en	Fecha en la que se encontró el defecto.
Razón	Se debe colocar por qué es un defecto.
Estado	Estado en el cual se encontrará el defecto, en este caso, será “nuevo”.
Ambiente	Desarrollo donde se hicieron las pruebas y se encontró el defecto.
Pasos para encontrar el defecto	Pasos que debe seguir el desarrollador para llegar al defecto.
Capturas de pantalla	Presentar capturas de pantalla donde se muestre el defecto.



Universidad Tecmilenio

Explicación

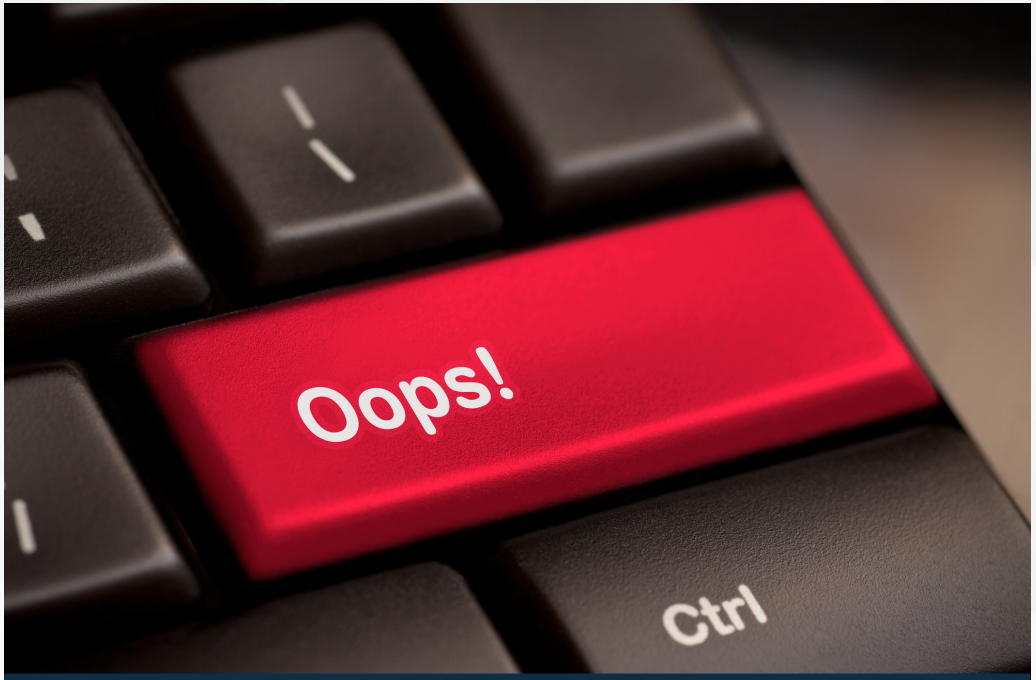
Ciclo de vida del defecto

- Nuevo
- Asignado
- Abierto o activo
- Solucionado
- Prueba pendiente
- Volver a la prueba
- Volver a abrir
- Verificado
- Cerrado

Actividad

¿Conoces alguna herramienta que administre los defectos?

¿Consideras que los defectos tienen una repercusión grave en el desarrollo?



Un defecto es cualquier situación provocada por una mala implementación. Si el defecto no es detectado y corregido a tiempo, el proyecto tendrá un retraso, además de provocar defectos o errores más delicados e involucrarán una mayor inversión de tiempo y un mayor retraso del proyecto.

Universidad Tecmilenio

Software Testing

Categorías de técnicas de pruebas

Semana 2

Universidad Tecmilenio

Introducción

Te encuentras desarrollando un videojuego para una empresa muy importante.

La empresa te solicita que tenga tres niveles y que en cada uno vaya aumentando la dificultad. Junto con tu equipo, se encuentran desarrollando y terminando el primer nivel, y deciden comenzar a utilizar las técnicas de pruebas existentes, sin embargo, no tenían el conocimiento de que una de estas técnicas debe ser ejecutada antes de que el desarrollo esté terminado y hay una gran probabilidad de que en las pruebas se encuentre un gran número de defectos.

Se pudo haber prevenido esta situación, si tan solo hubieran tenido el conocimiento de cómo, cuándo y dónde se deben ejecutar dichas técnicas.

Explicación

Elección de técnicas de pruebas

Técnicas estáticas

Se usan para:

- Detección temprana y corrección de defectos.
- Reducir el tiempo de desarrollo.
- Reducir el costo y tiempo de las pruebas.
- Mejorar la productividad del desarrollo.
- Reducir el número de fallas que se encontrarán en un futuro.

Técnicas dinámicas

Se usan para:

- Verificar el comportamiento funcional del software.
- Encontrar y corregir fallas.
- Realizar el proceso de validación.
- Revelar errores o “cuellos de botella” en el software.
- Probar casos para su finalización.

Explicación

Diferencias entre pruebas dinámicas y estáticas

Pruebas dinámicas	Pruebas estáticas
Se realizan una vez terminado el desarrollo.	Se realizan sin haber completado el desarrollo.
Realizan el proceso de validación.	Realizan el proceso de verificación.
Revelan errores.	Dan una evaluación del código y la documentación
Encuentran y corrigen las fallas.	Intentan prevenir los defectos.
Es más costoso encontrar las fallas.	Es menos costoso encontrar las fallas.

Explicación

Categorías de técnicas de pruebas y características

Actividad

¿Qué pasaría si no aplicas estas técnicas?

¿Consideras que son relevantes en las pruebas?

Universidad Tecmilenio

Cierre

Las técnicas de software se utilizan en diferentes momentos del desarrollo y te brindarán diferentes defectos que se deberán corregir para asegurar el correcto desempeño del desarrollo.

Universidad Tecmilenio

Software Testing

Pruebas de caja negra

Semana 2

Introducción

Al hacer las pruebas de software, es importante verificar que el desarrollo funcione correctamente, tanto por dentro como por fuera.

Para las validaciones del desarrollo desde fuera, existen las pruebas de caja negra, las cuales verifican el comportamiento del desarrollo como si fueran el usuario final.

Dentro de estas pruebas existen diferentes tipos, que permitirán detectar los diferentes defectos que se podrían encontrar en el desarrollo.

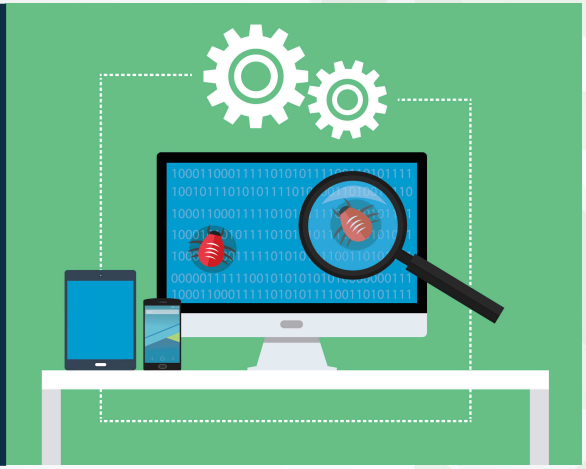
Universidad Tecmilenio

Definición y tipos de caja negra

Explicación

Las pruebas de caja negra y caja blanca detectan diferentes defectos, por lo cual, no puedes sustituir una con la otra, es decir, se complementan para verificar que el desarrollo esté al 100% (por dentro y por fuera).

Las pruebas de caja negra encuentran los siguientes defectos:



- Funciones incorrectas o faltantes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a otros sistemas o aplicativos.
- Errores de rendimiento.
- Errores de inicialización y de finalización.

Explicación

Pasos para realizar una caja negra de forma efectiva

1. Realiza un análisis de los requisitos y especificaciones del desarrollo.
2. Diseña las entradas. Serán válidas de forma que la prueba tenga un resultado positivo, y diseña las entradas que no serán válidas y den un resultado negativo.
3. Con las entradas que has planteado, establece cuáles serán las salidas esperadas para cada entrada.
4. Realiza el diseño de los casos de prueba.
5. Ejecuta los casos de prueba.
6. Verifica si el resultado obtenido es el mismo que el resultado esperado.
7. Si ambos resultados coinciden, el desarrollo funciona correctamente. En caso de haber encontrado un defecto, deberá ser corregido.

Explicación

Actividad

¿Qué pasaría si no aplicas las pruebas de caja negra?

¿Consideras que son relevantes todos los tipos de estas pruebas?

Universidad Tecmilenio

Cierre

Al momento de hacer las pruebas de software es importante verificar que el desarrollo funcione correctamente por dentro y por fuera.

Para las validaciones del desarrollo desde fuera se utilizan las pruebas de caja negra, ya que verifican el comportamiento del desarrollo como si fuera un usuario final.

Cada tipo de prueba dentro de esta te ayudará a detectar diferentes defectos, por lo tanto, los tipos de pruebas se complementan unos con otros.