Caja blanca y caja negra

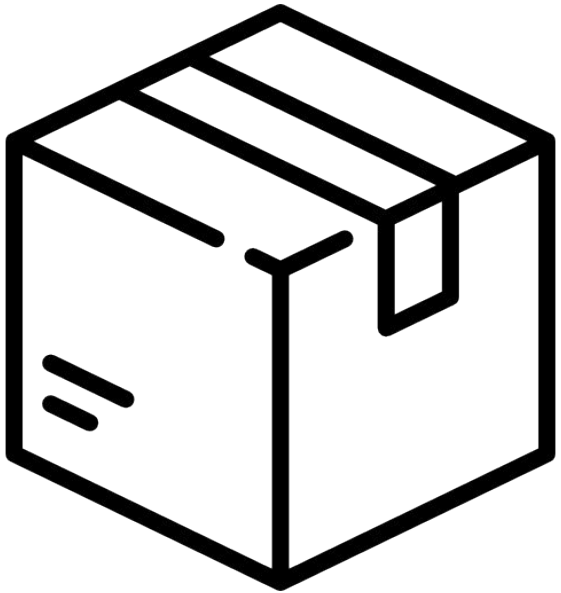
DigitalHouse >
Coding School

Certified Tech Developer
The Ultimate Degree

Índice

1 | Caja blanca

¿Qué es caja blanca?



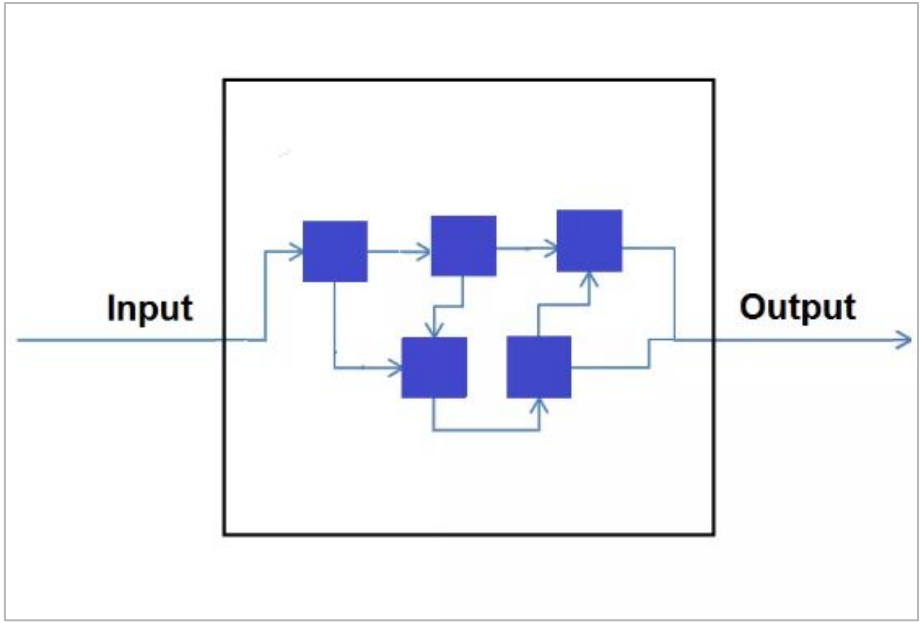
La usamos para probar la estructura interna, el diseño y la codificación de una solución de software. El código es visible para el tester y se enfoca principalmente en verificar el flujo de entradas y salidas a través de la aplicación, mejorando el diseño y la usabilidad, y también fortaleciendo la seguridad.

La prueba de caja blanca también se conoce como prueba de caja transparente, prueba de caja abierta, prueba estructural, prueba basada en código y prueba de caja de vidrio. Por lo general, la realizan desarrolladores.

¿Qué se verifica en la prueba de caja blanca?

- Agujeros de seguridad internos.
- Rutas rotas o mal estructuradas en los procesos de codificación.
- El flujo de entradas específicas a través del código.
- Rendimiento esperado.
- La funcionalidad de los bucles condicionales.
- Prueba de cada declaración, objeto y función de forma individual.

También se compone de una serie de niveles cuya esencia es la prueba cuidadosa de la aplicación a nivel de código fuente para reducir los errores ocultos más adelante.



Pasos para la implementación de caja blanca

Existen dos pasos básicos que llevan a cabo los testers cuando prueban una aplicación utilizando la técnica de prueba de caja blanca:

1) Entender el código fuente

El tester debe tener pleno conocimiento sobre el código de software y la lógica utilizados. Es recomendable que sea consciente de las prácticas de codificación segura y, por lo tanto, de la protección del código frente a los piratas informáticos.

Pasos para la implementación de caja blanca (cont.)

2) Crear casos de prueba y ejecutarlos

Las pruebas de caja blanca implican probar el código fuente de la aplicación para determinar el flujo y la estructura adecuados. Una forma es escribir más código para probar el código fuente de la aplicación y desarrollar pequeñas pruebas para cada proceso de la aplicación. Este método requiere que el tester tenga un conocimiento profundo del código, como ya mencionamos. Otros métodos incluyen pruebas manuales, pruebas de prueba y error y el uso de herramientas de prueba.

Herramientas para probar con caja blanca

- Parasoft Jtest
- EclEmma
- NUnit
- PyUnit
- HtmlUnit
- CppUnit

3 niveles de testing en caja blanca

Unit testing: garantiza que el código funcione según lo previsto antes de que ocurra la integración con el código probado previamente.

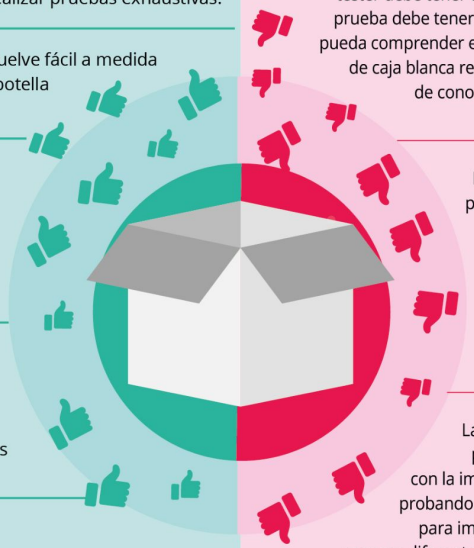
Integration testing: se prueban las interacciones de las interfaces entre sí, asegurando que el código funcionará en el entorno de integraciones del sistema.

Regression testing: se reciclan los casos utilizados en los niveles anteriores.

Ventajas y desventajas de caja blanca

Las pruebas de caja blanca son uno de los dos métodos de prueba más importantes que se utilizan en la actualidad. Tiene varias ventajas importantes:

1. Los efectos secundarios de tener conocimiento del código fuente son beneficiosos para realizar pruebas exhaustivas.
2. La optimización del código se vuelve fácil a medida que se exponen los cuellos de botella que pasan desapercibidos.
3. Otorgan mayor nivel de introspección al programador porque los desarrolladores describen cuidadosamente cualquier implementación nueva.
4. Proporcionan trazabilidad de las pruebas desde la fuente, lo que permite que los cambios futuros se capturen fácilmente en las pruebas recién agregadas o modificadas.
5. Fáciles de automatizar.
6. Proporcionan reglas claras basadas en ingeniería sobre cuándo detener las pruebas.



Aunque las pruebas de caja blanca tienen grandes ventajas, no son perfectas y contienen algunas desventajas:

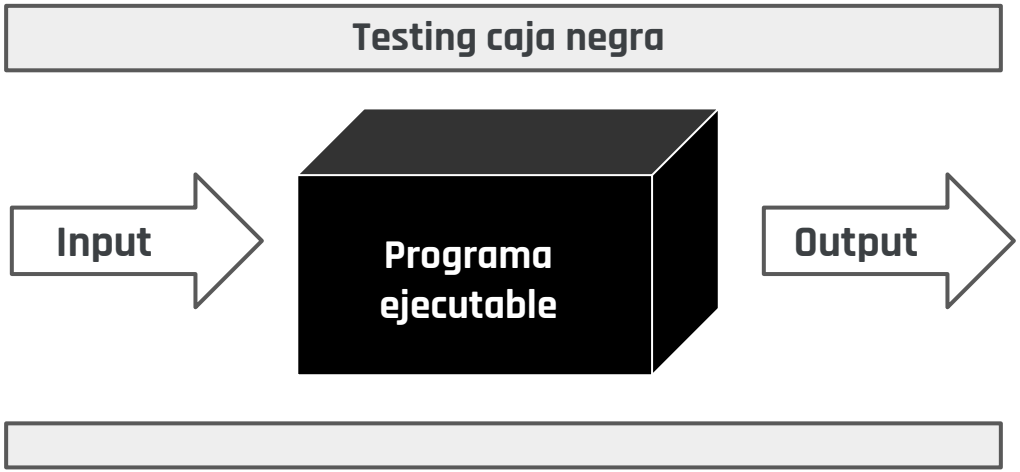
1. Las pruebas de caja blanca aportan complejidad porque el tester debe tener conocimiento del programa o el equipo de prueba debe tener al menos un muy buen programador que pueda comprender el programa a nivel de código. Las pruebas de caja blanca requieren un programador con un alto nivel de conocimiento debido a la complejidad del nivel de pruebas que se debe realizar.
2. En algunas ocasiones, no es realista poder probar todas y cada una de las condiciones existentes de la aplicación y algunas condiciones no se probarán.
3. Las pruebas se centran en el software tal como existe y es posible que no se descubra la funcionalidad faltante.
4. Las pruebas resultantes pueden ser frágiles porque están estrechamente relacionadas con la implementación específica de lo que se está probando. El código bajo prueba podría reescribirse para implementar la misma funcionalidad de una manera diferente que invalide los supuestos incluidos en la prueba. Esto podría resultar en pruebas que fallan innecesariamente o, en el peor de los casos, pruebas que ahora dan falsos positivos y enmascaran errores en el código.

2 | Caja negra

¿Qué es caja negra?

En este modelo de testing probamos la funcionalidad de la aplicación sin mirar la estructura del código interno y solo nos enfocamos en las entradas y salidas del sistema de software. Este tipo de prueba se basa completamente en requisitos y especificaciones de software. También se conoce como pruebas conductuales, de caja opaca, de caja cerrada, basadas en especificaciones u ojo a ojo.

Este método de prueba también se conoce como prueba de comportamiento y prueba funcional y es fundamental durante las etapas del ciclo de vida de las pruebas de software, como las pruebas de regresión, aceptación, unidad, sistema, integración y desarrollo de software.



Pasos para la implementación de caja negra

1. Reunir los requisitos y las especificaciones del software, los testers comprueban el software con entradas positivas y negativas y las detectan.
2. Testear con casos de prueba seleccionados.
3. Se ejecutan todos los casos de prueba.
4. Los testers verifican todas las salidas para las entradas dadas.
5. Los defectos, si los hubiere, se reparan y se vuelven a probar.

Testeo funcional y no funcional con caja negra

Testeo funcional

El testeo funcional se ocupa de los requisitos funcionales o las especificaciones de una aplicación. Aquí se probarán, entonces, diferentes funciones del sistema proporcionando la entrada y comparando la salida real con la salida esperada.

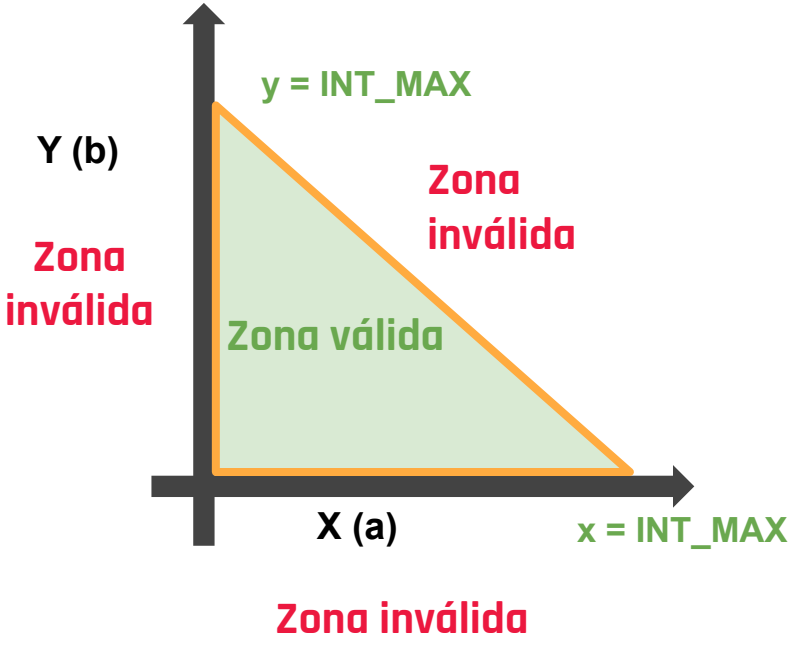
Testeo no funcional

El testeo no funcional se ocupa de la calidad y el rendimiento de la aplicación.

Técnicas de testing en caja negra

1) Partición de equivalencia

Los valores de entrada al sistema o la aplicación se dividen en diferentes clases o grupos en función de su similitud en el resultado. Por lo tanto, en lugar de usar todos y cada uno de los valores de entrada, ahora podemos usar cualquier valor de la clase para probar el resultado, lo que nos permite mantener la cobertura de la prueba mientras podemos reducir una gran cantidad de retrabajos y, lo más importante, el tiempo invertido.



2) Análisis de valor límite

Comprueba el sistema en los límites donde cambia la salida del sistema. Por ejemplo, si un sistema tiene un rendimiento de 50 segundos, ¿cómo actuará a los 49 segundos o 51 segundos y cómo cambia su salida?

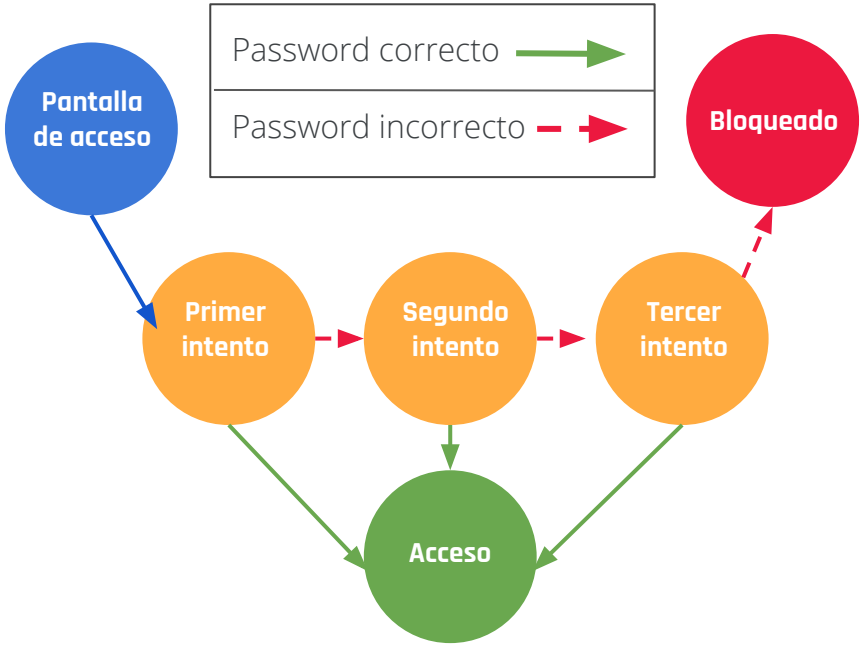
3) Prueba de la tabla de decisión

Actúa para cualquier sistema en el que tenga que tomar una decisión de acuerdo con una condición particular, como la del caso if else o switch.

	Regla 1	Regla 2	Regla 3	Regla 4
Condición				
Fin de mes	No	Sí	Sí	Sí
Salario transferido	N/A	No	Sí	Sí
Fondo de previsión	N/A	N/A	No	Sí
Acción				
Impuesto a la renta	No	No	Sí	Sí
Fondo de previsión	No	No	No	Sí

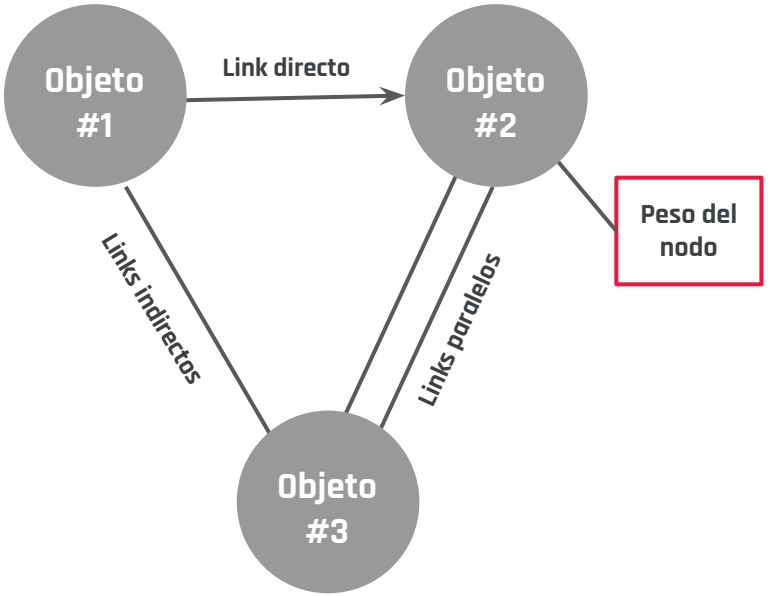
4) Prueba de transición de estados

Cuando un sistema tiene muchos estados (como el de un sitio web: página de inicio, página de inicio de sesión, página de cierre de sesión, etc.), entonces, tenemos que verificar cómo el sistema otorga salidas para todas las entradas.



5) Métodos de prueba basados en gráficos

Todas y cada una de las aplicaciones son una acumulación de algunos objetos. Todos estos objetos se identifican y se prepara el gráfico. A partir de este gráfico de objeto, se identifica cada relación de objeto y los casos de prueba se escriben en consecuencia para descubrir los errores.



6) Error guessing

La prueba se realiza de acuerdo con la experiencia del tester. Se prueban todos los errores comunes que normalmente cometen los desarrolladores y se prueba la codificación de excepciones.

7) Prueba de comparación

Se utilizan diferentes versiones independientes del mismo software para comparar entre sí.


DigitalHouse>
Coding School