

# PRUEBAS DE SOFTWARE

**Semana 2: Metodologías, Tipos y Técnicas de Pruebas de Software.**  
**Principios del testing de software. Metodologías de Pruebas de Software. Tipos de testing de software: pruebas de caja negra, pruebas de caja blanca y pruebas de caja gris.**

Ing. Quiroz Angulo Christian Janderson



Universidad  
Tecnológica  
del Perú

# Logro de la sesión

Al finalizar la sesión, el estudiante **reconoce y diferencia los tipos de pruebas de software (caja negra, caja blanca y caja gris)**, comprendiendo sus características, aplicaciones y ejemplos prácticos, con el fin de aplicarlos en el diseño de casos de prueba dentro de un proyecto de desarrollo de software.

# Dudas sobre tema el tema anterior

**¿En qué casos conviene usar pruebas de caja negra en lugar de caja blanca?**

Muchos estudiantes se confunden sobre cuándo es más eficiente validar solo entradas/salidas sin revisar el código interno.

**¿Las pruebas de caja blanca siempre requieren conocimientos de programación?**

Se suele dudar si un tester funcional puede ejecutarlas o si es exclusivo de desarrolladores.



# Conocimientos previos

Para comprender los tipos de pruebas de software, el estudiante debe conocer el ciclo de vida del software y la importancia de la calidad, qué es un caso de prueba (entradas, acciones y salidas esperadas), la diferencia entre requisitos funcionales y no funcionales, conceptos básicos de programación como condicionales, bucles y lógica simple, además del uso básico de entornos de prueba y la noción de automatización.



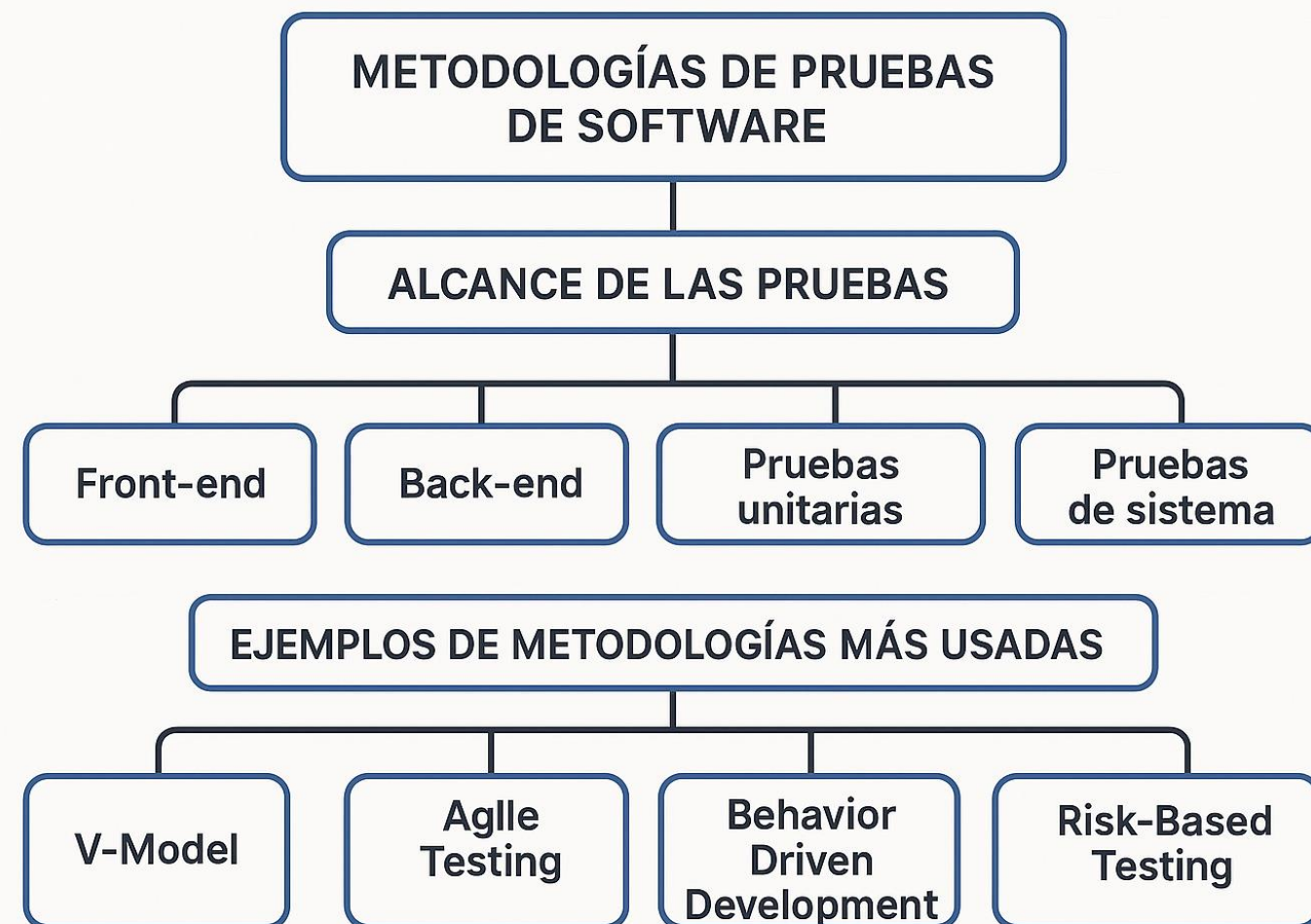


# Utilidad del tema

La utilidad de estudiar los tipos de pruebas de software radica en que permite seleccionar el enfoque adecuado para validar la calidad de un sistema, garantizando que cumpla con los requisitos funcionales y no funcionales. Además, ayuda a identificar errores de manera más eficiente, optimizar recursos durante el proceso de testing y asegurar que el software entregue valor real a los usuarios y organizaciones.

# Metodologías de pruebas de software

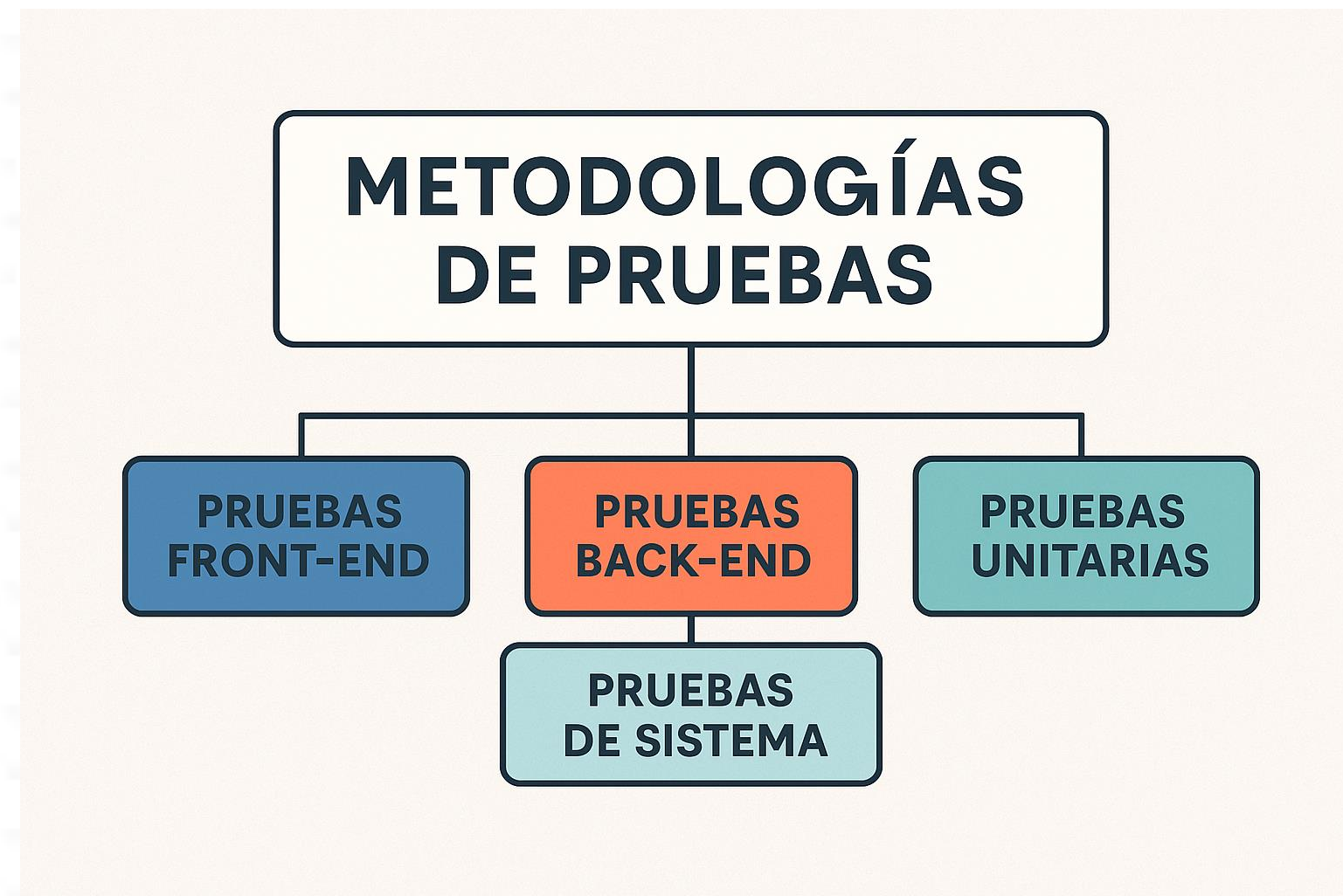
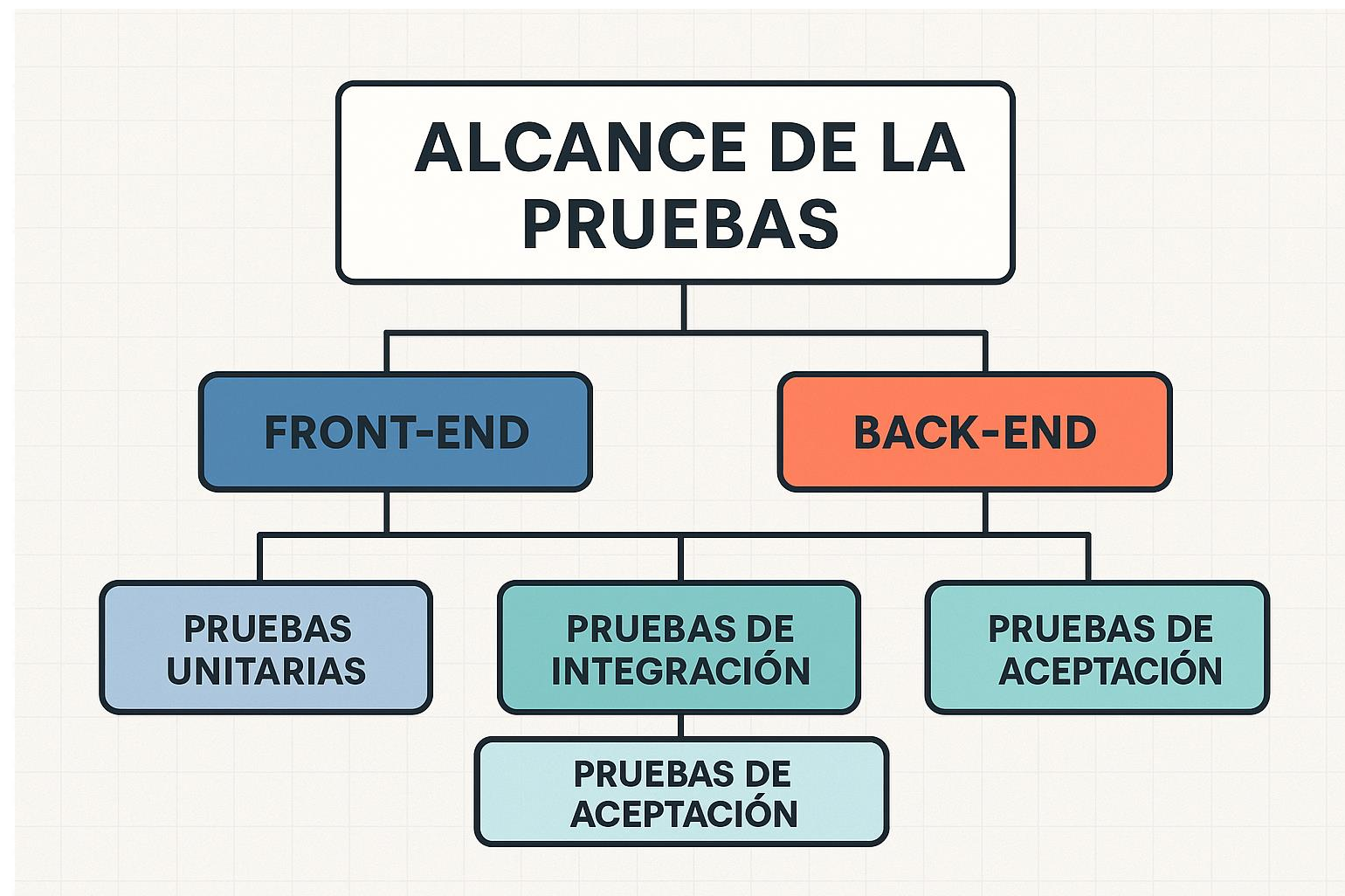
Las metodologías de pruebas de software son los diferentes **enfoques, estrategias y prácticas organizadas** que permiten verificar y validar que una aplicación cumpla con los requerimientos funcionales, de calidad y de usabilidad establecidos. Estas metodologías abarcan desde pruebas **unitarias**, donde se valida el correcto funcionamiento de componentes individuales, hasta pruebas **de sistema**, que comprueban la integración completa de la aplicación.





## Alcance de las pruebas

- **Front-end:** Validan la interfaz gráfica, la experiencia del usuario (UX) y la interacción.
- **Back-end:** Evalúan servicios, bases de datos, APIs y lógica del negocio.
- **Pruebas unitarias:** Se enfocan en módulos o funciones específicas.
- **Pruebas de integración:** Validan cómo interactúan los componentes entre sí.
- **Pruebas de sistema:** Evalúan el software como un todo.
- **Pruebas de aceptación:** Determinan si el sistema satisface los requisitos del cliente o usuario final.



# Metodologías de pruebas funcionales y no funcionales

El objetivo de utilizar diversas metodologías de prueba en el proceso de desarrollo es garantizar que el software funcione correctamente en múltiples entornos y plataformas. Estas suelen dividirse en pruebas funcionales y no funcionales.

## Métodos de pruebas funcionales

Las pruebas funcionales implican probar la aplicación según los requisitos del negocio. Abarca todos los tipos de pruebas diseñados para garantizar que cada componente de un software se comporte según lo previsto mediante casos de uso proporcionados por el equipo de diseño o el analista de negocios. Estos métodos de prueba suelen realizarse en orden e incluyen:

- Pruebas unitarias
- Pruebas de integración
- Pruebas del sistema
- Pruebas de aceptación
- Pruebas de rendimiento
- Pruebas de seguridad
- Pruebas de usabilidad
- Pruebas de compatibilidad



# Métodos de prueba no funcionales

Los métodos de pruebas no funcionales abarcan todos los tipos de pruebas centradas en los aspectos operativos de un software. Estos incluyen:

- Pruebas de rendimiento
- Pruebas de seguridad
- Pruebas de usabilidad
- Pruebas de compatibilidad

La clave para lanzar software de alta calidad que los usuarios finales puedan adoptar fácilmente es construir un marco de pruebas sólido que implemente metodologías de pruebas de software tanto funcionales como no funcionales.

## Non-Functional Testing



Performance Testing



Security Testing



Usability Testing



Compatibility Testing

### FUNCTIONAL TESTING

#### EXAMPLE

Verifying login functionality

#### EXAMPLE

Checking search feature

### NON-FUNCTIONAL TESTING

Evaluating response time

Assessing reliability

# Principios del Testing de Software

- **Las pruebas muestran la presencia de defectos, no su ausencia;** El testing puede evidenciar errores, pero no garantiza que el software esté 100% libre de fallos.
- **Las pruebas exhaustivas son imposibles;** No se puede probar todo; se deben seleccionar pruebas prioritarias y representativas.
- **Las pruebas tempranas ahorran costos;** Mientras antes se detecten los defectos (desde etapas de requisitos y diseño), más barato y rápido será corregirlos.
- **La concentración de defectos es desigual;** La mayoría de los errores suelen encontrarse en unos pocos módulos o funciones críticas.
- **La paradoja del pesticida;** Si se repiten siempre las mismas pruebas, dejan de encontrar errores. Es necesario revisarlas y variarlas.
- **Las pruebas dependen del contexto;** El enfoque de pruebas varía según el tipo de sistema: no es igual probar un banco que un videojuego.
- **La ilusión de ausencia de errores;** Un software sin errores detectados no necesariamente es útil; también debe cumplir con las necesidades del usuario.



- Las pruebas muestran la presencia de defectos, no su ausencia



- Las pruebas exhaustivas son imposibles



- Las pruebas tempranas ahorran costos



- La concentración de defectos es desigual



- La paradoja del pesticida



- Las pruebas dependen del contexto



# Pruebas de Caja Negra

Se enfocan en **qué hace el sistema**, sin importar cómo lo hace internamente.

- El tester no conoce el código fuente.
- Se prueban entradas y salidas, validando el comportamiento esperado.
- Útiles para pruebas funcionales y de aceptación.

**Ejemplo:** Ingresar un usuario y contraseña en un login y comprobar si el sistema da acceso o muestra error.

## PRUEBAS DE CAJA NEGRA



se enfocan en  
las entradas  
y salidas



# Pruebas de Caja Blanca

Se enfocan en **cómo funciona el sistema internamente**, revisando la lógica del código.

- El tester conoce el código fuente y estructuras internas.
- Se prueban condiciones, bucles, flujos y estructuras de datos.
- Útiles para pruebas unitarias y de seguridad.

**Ejemplo:** Revisar que un algoritmo de cálculo de impuestos siga correctamente cada rama condicional y devuelva resultados válidos.

## PRUEBAS DE CAJA BLANCA



se enfocan en la  
estructura interna

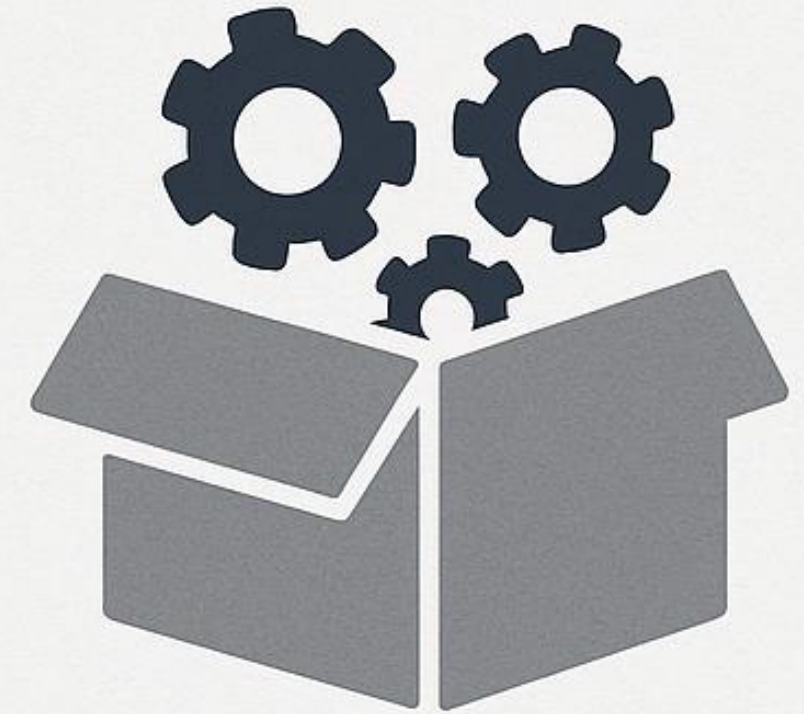
# Pruebas de Caja Gris

Combinan elementos de **caja negra** y **caja blanca**.

- El tester tiene conocimiento parcial del código o arquitectura.
- Permiten validar tanto la funcionalidad externa como algunos aspectos internos.
- Útiles para pruebas de integración y sistemas complejos.

**Ejemplo:** Probar un servicio web (API) validando la respuesta final, pero también verificando logs y bases de datos para asegurar que el proceso interno sea correcto.

## PRUEBAS DE CAJA GRIS



combinan  
ambos enfoques



## **Lo que debemos saber:**

Caja Negra → Se centra en qué hace el sistema.

Caja Blanca → Se centra en cómo lo hace internamente.

Caja Gris → Una mezcla de ambos enfoques.



# ¿Alguna consulta?



# Actividad: Grupal

En grupos; los alumnos explican con un ejemplo sobre los temas y brindan una conclusión:



1. **Grupo 1:** Explicar un caso, donde se demuestre las Pruebas de rendimiento.
2. **Grupo 2:** Explicar un caso, donde se demuestre las Pruebas de seguridad.
3. **Grupo 3:** Explicar un caso, donde se demuestre las Pruebas de usabilidad.
4. **Grupo 4:** Explicar un caso, donde se demuestre las Pruebas de compatibilidad.
5. **Grupo 5:** Explicar un caso, donde se demuestre las Pruebas de unitarias.
6. **Grupo 6:** Explicar un caso, donde se demuestre las Pruebas de integración.

# Cierre



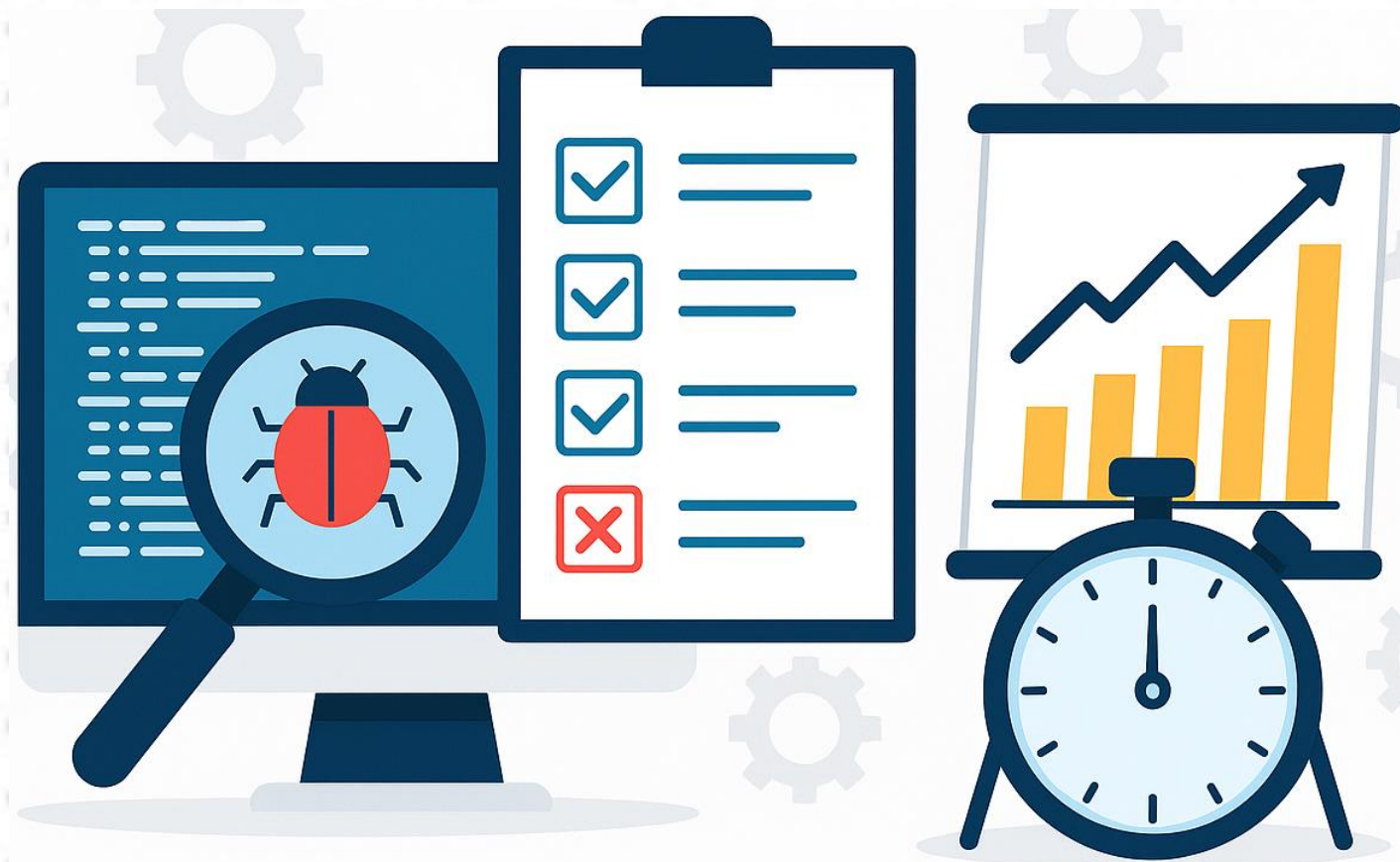
## ¿Qué hemos aprendido hoy?

Hoy hemos aprendido que las pruebas de software son fundamentales para garantizar la calidad de un sistema. Conocimos los principios del testing, entendimos los tipos de pruebas según el acceso a la información: caja negra, caja blanca y caja gris, y vimos cómo cada una se aplica en diferentes contextos.

Reconocimos también la importancia de los conocimientos previos para aplicar estos enfoques y comprendimos la utilidad del tema, que se centra en detectar errores de manera eficiente y asegurar que el software cumpla con los requisitos y necesidades del usuario.



# Conclusión



El testing de software es una actividad esencial para garantizar calidad y confianza en los sistemas. Los enfoques de **caja negra**, **caja blanca** y **caja gris** permiten analizar el software desde distintas perspectivas, asegurando tanto la correcta funcionalidad externa como la validez de la lógica interna.

Aplicar estos métodos de manera adecuada no solo ayuda a detectar errores, sino que también asegura que el producto final responda a las necesidades reales de los usuarios.



**Universidad  
Tecnológica  
del Perú**