

Pruebas de software

Entornos de desarrollo

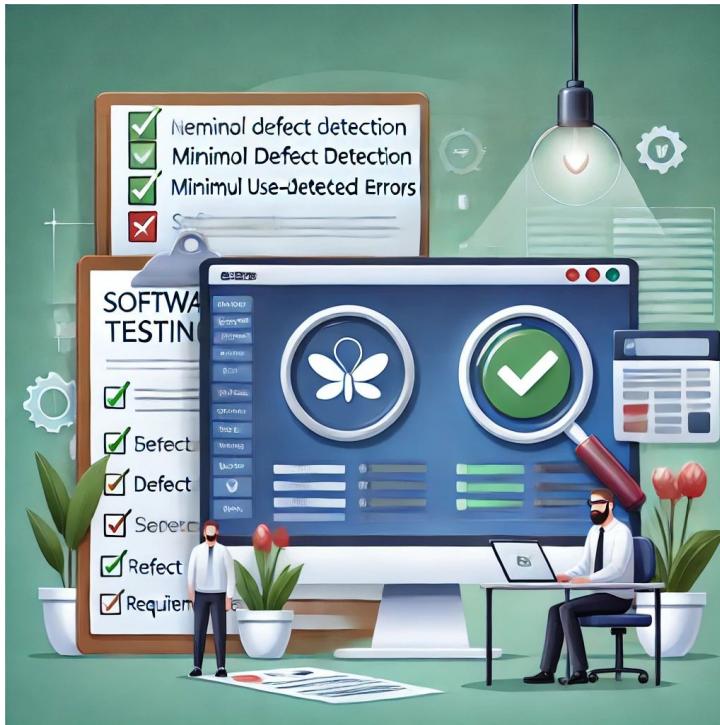
Diapositivas realizadas por Aitor Ventura Edo

IES El Caminás
Curso 2025–2026

“La prueba de programas puede utilizarse para mostrar la presencia de errores, pero nunca para demostrar su ausencia”

Edgar Dijkstra

Prueba de software



Una prueba de software es el proceso de ejecución controlada de un programa o sistema con el objetivo de detectar defectos.

Evalúa si el software desarrollado cumple con los requerimientos especificados y funciona de acuerdo con lo esperado.

Uno de los objetivos fundamentales es que el porcentaje de fallos detectados por el usuario sea lo menor posible.

Prueba de software



Las **pruebas** son fundamentales en la creación de productos industriales y en el desarrollo de software.

¿Comprarías una nevera que no funciona?

Prueba de software



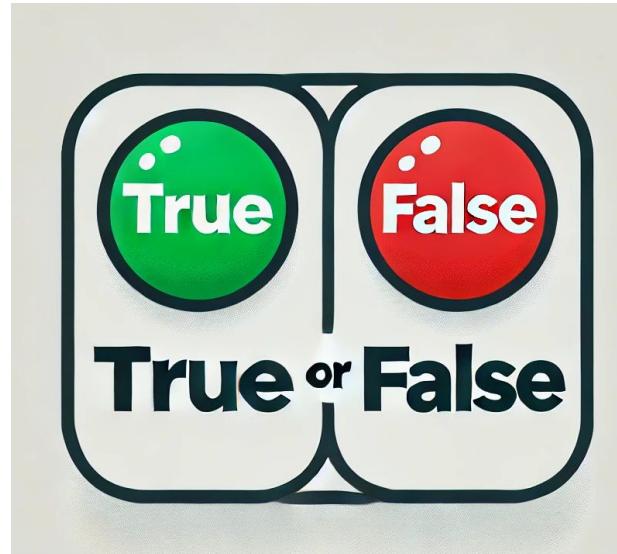
¡Con el software ocurre igual!

Un software **NO** debe entregarse al usuario final con **errores**, ya que esto **afecta la profesionalidad y disminuye la confianza de los clientes**, impactando negativamente en futuras oportunidades.

Prueba de software

¿VERDADERO o FALSO?

Las pruebas de software consisten en probar que las funcionalidades del programa sean las esperadas una vez se ha finalizado la implementación.



Prueba de software



Las **pruebas de software** consisten en **probar que las funcionalidades del programa sean las esperadas** una vez se ha finalizado la implementación.

¿Por qué?

Estas **pruebas pueden y deben** realizarse en **diferentes etapas** del desarrollo.

Además, las **pruebas no solo verifican funcionalidades, sino también aspectos como el rendimiento, la seguridad, la usabilidad y la compatibilidad**. Por lo tanto, las **pruebas son un proceso continuo y no exclusivo del final del desarrollo**.

Impacto de los errores

Ejemplo de impacto de errores

- Error en planificación: Desviaciones de tiempo y coste, como el uso de más recursos de los planeados.
- Error crítico en diseño: Omisión de un campo clave en la base de datos puede repercutir en tareas posteriores, requiriendo retrabajo desde etapas iniciales.

Consecuencia de errores no detectados:

- Un error no identificado al inicio puede requerir hasta 50 veces más esfuerzo para solucionarse en fases avanzadas.

Relevancia y Objetivo de las Pruebas

1. Importancia de la fase de pruebas:

- Representa entre el **30% y 50% del coste total** del proyecto.
- Influye en la percepción del cliente sobre la **calidad del software entregado**.

2. Objetivo principal de las pruebas:

- Evaluar la **calidad del software** durante todo su ciclo de vida.
- Validar que el software **cumple con los requerimientos** y se comporta como fue diseñado.

3. Enfoque del ciclo de vida del software:

- Realización de **controles periódicos** para evaluar la **calidad** y detectar defectos tempranamente.
- Todas las aplicaciones **deben ser probadas de manera controlada** antes de su entrega al cliente.

Relevancia y Objetivo de las Pruebas

4. Limitaciones de las pruebas:

- Prueba exhaustiva es impracticable, incluso en programas pequeños.
- Descubrir un defecto es un éxito, no un fracaso, ya que permite mejorar el software.

5. Naturaleza de los defectos:

- Errores no siempre son negligencias del programador; pueden ser consecuencia de:
 - Mala comunicación en el equipo.
 - Malinterpretación de los requisitos.

6. Beneficio de detectar errores:

- El descubrimiento de defectos impulsa la mejora continua y asegura un producto de mayor calidad para el cliente.

Recomendaciones de G.J. Myers

1. Definir resultados esperados:

- Cada caso de prueba debe especificar el resultado esperado.
- Las **discrepancias** entre el resultado esperado y el obtenido son **indicios de defectos**.



2. Separación de roles:

- El **programador** no debería probar su propio código.
- Ideal: **otra persona** debe realizar las **pruebas** para detectar errores más fácilmente.

Recomendaciones de G.J. Myers

3. Inspección minuciosa:

- Analizar cuidadosamente los resultados de las pruebas.
- Evitar ignorar síntomas claros de defectos.



4. Amplitud en los datos de prueba:

- Usar tanto datos válidos como no válidos para cubrir casos esperados e inesperados.

Recomendaciones de G.J. Myers

5. Doble enfoque de las pruebas:

- Verificar si el software no hace lo que debería hacer.
- Comprobar si el software hace lo que no debería hacer (efectos secundarios adversos).



6. Documentar los casos de prueba:

- Evitar casos desecharables.
- Diseñar y documentar las pruebas para repetirlas si es necesario.

Recomendaciones de G.J. Myers

7. Reconocer la presencia de defectos:

- Asumir que **siempre hay defectos**.
- Dedicar suficientes recursos a pruebas y depuración.



8. Concentración de defectos:

- Los **defectos tienden a agruparse**: si hay uno, es probable que haya más en la misma área.

Recomendaciones de G.J. Myers

9. Creatividad en las pruebas:

- Las pruebas **son una tarea creativa**, no rutinaria.
- Requieren **ingenio para detectar el mayor número de defectos** con los recursos disponibles.



Verificación vs Validación



Mediante la realización de pruebas de software, se van a realizar las tareas de verificación y validación del software.

- **Verificación:** Comprueba que el software se está construyendo correctamente según las condiciones impuestas.
- **Validación:** Asegura que el software cumple lo que el usuario desea y satisface los requerimientos.

Metáfora: Construcción de una Casa



Verificación: Es como revisar durante la construcción de una casa que los planos arquitectónicos se están siguiendo correctamente.

- ¿Los muros tienen la altura especificada?
- ¿Las instalaciones eléctricas cumplen con las normas de seguridad?

Validación: Es como preguntar al propietario de la casa si la casa cumple con sus expectativas.

- ¿Las habitaciones tienen el tamaño que necesita?
- ¿La casa satisface su idea de "hogar"?

Estrategia de pruebas basada en el Modelo en Espiral



- Prueba de unidad:
 - Analiza el código implementado en pequeños bloques funcionales.
- Prueba de integración:
 - Verifica el diseño y la arquitectura del sistema al combinar los componentes.
- Prueba de validación:
 - Compara el sistema con los requisitos definidos para garantizar que cumple su propósito.
- Prueba de sistema:
 - Examina el funcionamiento completo del software y su interacción con otros elementos del entorno.

Típos de pruebas

1. Pruebas de Unidad

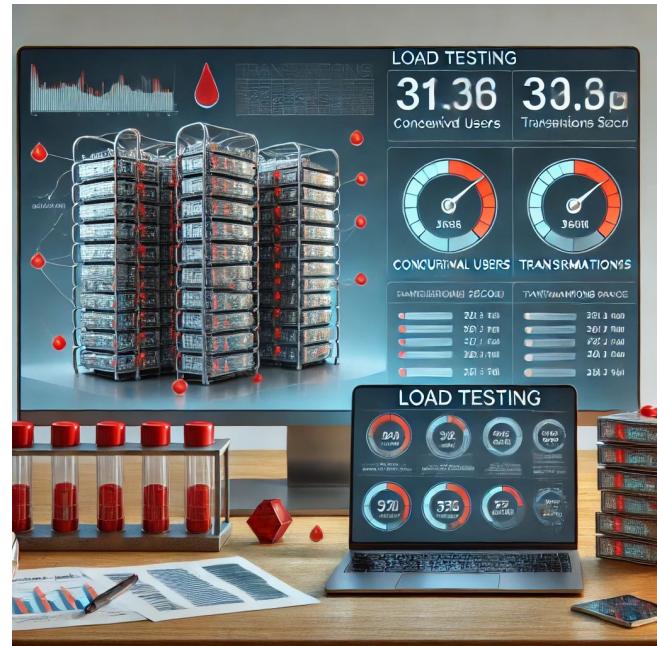
- Propósito: Validar el **correcto funcionamiento** de un módulo específico de código.
- Enfoque: Asegurarse de que **cada componente** funciona de forma **independiente**.



Típos de pruebas

2. Pruebas de Carga

- Propósito: Evaluar el comportamiento de la **aplicación** bajo una cantidad de **peticiones** esperada.
- Aspectos evaluados:
 - **Usuarios concurrentes.**
 - **Transacciones por unidad de tiempo.**
 - Identificar **cuellos de botella** (servidor, base de datos, etc.).



Típos de pruebas

3. Pruebas de Estrés

- **Propósito:** Probar la resistencia de la aplicación en condiciones extremas, aumentando gradualmente la carga hasta que falle.
- **Utilidad:** Determinar la robustez en momentos de carga muy alta y si puede soportar picos inesperados.



Típos de pruebas

4. Pruebas de Estabilidad

- Propósito: Verificar si la aplicación puede manejar una carga esperada de forma continua sin fallos.
- Aspectos evaluados:
 - Detección de fugas de memoria.
 - Mantenimiento del rendimiento en sesiones prolongadas.



Típos de pruebas

5. Pruebas de Picos

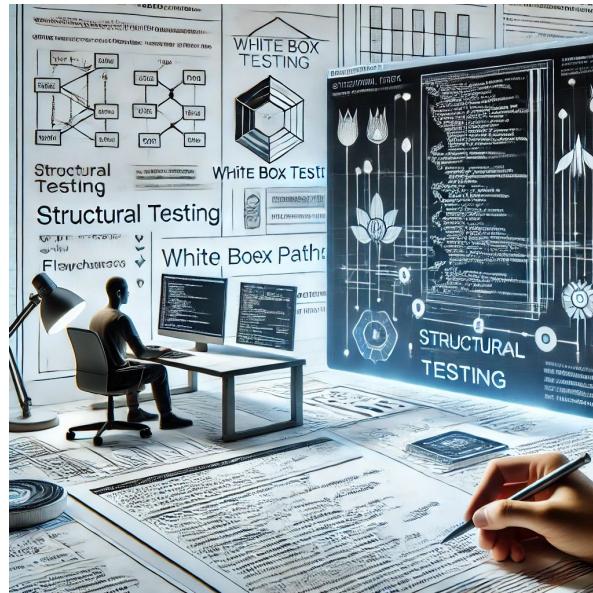
- **Propósito:** Observar el comportamiento del sistema ante cambios drásticos en el número de usuarios.
- **Enfoque:** Analizar variaciones súbitas de carga (aumento o disminución).



Típos de pruebas

6. Prueba Estructural (Caja Blanca)

- Propósito: Evaluar la estructura interna del programa y los caminos de ejecución.
- Enfoque: Analizar el código y sus rutas lógicas.



Típos de pruebas

7. Prueba Funcional (Caja Negra)

- Propósito: Evaluar las funciones basándose en las entradas y salidas, sin analizar el código interno.
- Enfoque: Comprobar que las funcionalidades cumplen con los requerimientos especificados.



Típos de pruebas

8. Pruebas Aleatorias

- Propósito: Generar casos de prueba a partir de **modelos estadísticos que simulen entradas posibles al sistema.**
- Enfoque: Descubrir **errores en escenarios no predecibles.**



Típos de pruebas

9. Pruebas de Regresión

- Propósito: Detectar errores inducidos por cambios recientes en el software.
- Enfoque:
 - Verificar que las modificaciones no generen problemas en funcionalidades previamente correctas.
 - Comparar el comportamiento actual con el esperado.

