**✅ Pruebas de Software – Resumen Completo**

**¿Qué son las pruebas de software?**

Las pruebas de software son actividades destinadas a verificar que el software cumpla con los requerimientos establecidos y funcione correctamente.

**Importancia de las pruebas:**

* Identificar errores y defectos antes del lanzamiento.
* Asegurar la calidad del producto final.
* Reducir el riesgo de fallos en producción.
* Validar el cumplimiento de los requerimientos del cliente.

**Objetivos de las pruebas:**

* Validar el software frente a los requerimientos.
* Verificar su funcionalidad.
* Mejorar la confiabilidad y la seguridad.

**🔄 Ciclo de Vida de las Pruebas de Software**

**1. Planificación:**  
Define objetivos, alcance, recursos y cronograma.  
📌 *Ejemplo:* Se elige realizar pruebas funcionales y de rendimiento antes del despliegue.

**2. Diseño:**  
Creación de casos de prueba detallados.  
📌 *Ejemplo:* Se escriben los pasos para verificar el login del sistema.

**3. Ejecución:**  
Los testers ejecutan los casos definidos.  
📌 *Ejemplo:* Se ejecuta una prueba de login con usuario y contraseña incorrectos.

**4. Reporte:**  
Registro de defectos detectados.  
📌 *Ejemplo:* Se reporta que el botón “Guardar” no responde.

**5. Cierre:**  
Evaluación del proceso de prueba y lecciones aprendidas.  
📌 *Ejemplo:* Se documentan los errores más comunes encontrados en el sprint.

**🧪 Tipos de Pruebas de Desarrollo de Software (con ejemplos)**

**🔹 Pruebas Unitarias**

Verifican el correcto funcionamiento de componentes individuales.  
📌 *Ejemplo:* Comprobar si una función que calcula el IVA retorna el valor correcto.

**🔹 Pruebas de Integración**

Validan la interacción entre módulos.  
📌 *Ejemplo:* Probar si el módulo de facturación se comunica correctamente con el módulo de stock.

**🔹 Pruebas Funcionales**

Revisan si el software cumple con los requerimientos funcionales.  
📌 *Ejemplo:* Verificar que un usuario pueda registrarse con email y contraseña.

**🔹 Pruebas de Aceptación**

Aseguran que el software cumple las expectativas del cliente.  
📌 *Ejemplo:* Cliente prueba el sistema y valida que puede cargar un pedido desde su panel.

**🔹 Pruebas de Rendimiento**

Evalúan cómo se comporta el sistema bajo carga.  
📌 *Ejemplo:* Simular 500 usuarios realizando búsquedas simultáneas.

**🧠 Diseño de Casos de Prueba**

Un **caso de prueba** es un conjunto de pasos y condiciones que permiten validar una funcionalidad.  
**Elementos:**

* Identificador
* Descripción
* Precondiciones
* Pasos
* Resultado esperado
* Resultado actual

**▶️ Ejecución de Pruebas**

Pasos típicos:

1. Preparar el entorno.
2. Ejecutar los casos.
3. Registrar resultados.
4. Reportar defectos si se encuentran.

**🔳 Caja Negra vs Caja Blanca**

| **Tipo** | **Descripción** | **Ejemplos** |
| --- | --- | --- |
| **Caja Negra** | No requiere conocimiento del código interno. Evalúa funcionalidades. | Pruebas funcionales, de aceptación |
| **Caja Blanca** | Requiere conocer la lógica del código. Evalúa la estructura interna. | Pruebas unitarias, de integración |

**🧰 Tipos Especializados de Pruebas (con ejemplos y herramientas)**

| **Tipo de prueba** | **Ejemplo práctico** | **Herramientas comunes** |
| --- | --- | --- |
| **Regresión** | Probar funciones antiguas tras una actualización | Selenium, TestComplete |
| **Seguridad** | Simular ataques de inyección SQL | OWASP ZAP, Burp Suite |
| **Usabilidad** | Evaluar si el usuario encuentra fácilmente una opción | UserTesting, Hotjar |
| **Compatibilidad** | Probar en Chrome, Firefox, Safari | BrowserStack, Sauce Labs |
| **Localización** | Validar que los textos estén traducidos correctamente | SDL Trados, MemoQ |
| **Instalación** | Verificar que la app se instale sin errores en Windows | InstallShield, NSIS |
| **Recuperación** | Simular un fallo del servidor y medir la respuesta | Chaos Monkey, Gremlin |
| **Estrés** | Simular miles de usuarios simultáneos | JMeter, Gatling |
| **Carga** | Verificar respuesta con 1000 usuarios | LoadRunner, Gatling |
| **Volumen** | Procesar millones de registros | JMeter |
| **Configuración** | Probar diferentes combinaciones de hardware/software | Docker, VMware |
| **Documentación** | Revisar si los manuales de usuario son claros | MadCap Flare, RoboHelp |
| **Interfaz de usuario** | Verificar que botones y menús funcionen | Selenium, Ranorex |
| **Accesibilidad** | Validar compatibilidad con lectores de pantalla | Axe, NVDA |
| **API** | Verificar respuesta de una API REST | Postman, SoapUI |
| **Base de datos** | Validar la integridad de los datos y transacciones | SQL Server Management Studio |
| **Red** | Verificar funcionamiento con red lenta o pérdida de paquetes | Wireshark, Fiddler |
| **Compatibilidad de SO** | Probar en Windows, Linux, macOS | VirtualBox, Docker |
| **Compatibilidad de dispositivos** | Probar en iOS y Android | Appium, BrowserStack |

**🧩 Conclusión**

* Las pruebas de software aseguran calidad, funcionalidad y rendimiento.
* Existen numerosos tipos de pruebas para cubrir todos los aspectos del desarrollo.
* Las herramientas de automatización aumentan la eficiencia y cobertura.
* La elección adecuada de pruebas y métricas mejora el proceso y el producto final.
* **Pruebas de Software en Aplicaciones Web**
* **1. Introducción** Las pruebas de software son fundamentales en cualquier desarrollo informático, y en el caso de las aplicaciones web, adquieren particular relevancia debido a su naturaleza distribuida y al entorno en el que se ejecutan.
* **2. Métodos de Desarrollo para la Web**
* Desarrollo iterativo
* Desarrollo incremental
* Programación extrema (XP)
* Desarrollo guiado por pruebas (TDD)
* Integración continua
* **3. Tipos Generales de Pruebas de Software**
* **Prueba Unitaria**: Ejecutar y validar cada módulo individual. Ejemplo: probar una función que calcula el IVA.
* **Pruebas de Integración**: Verifican la interacción entre módulos. Ejemplo: conexión entre el módulo de pagos y el de facturación.
* **Pruebas del Sistema**: Evalúa todo el sistema. Ingreso, procesamiento, salida. Ejemplo: registrar un usuario y simular una compra.
* **Pruebas de Regresión**: Identifican errores introducidos al combinar programas previamente probados. Se recomienda utilizar la técnica "bottom-up". Ejemplo: se reejecutan pruebas tras cambios recientes para asegurar que nada previo falló.
* **Pruebas de Humo**: Detectan errores tempranos en versiones preliminares. Consisten en ejecutar integraciones completas periódicamente (diarias o semanales). Ejemplo: revisión general tras cada despliegue.
* **Pruebas de Aceptación**: Determinan si el sistema cumple los requisitos del cliente antes del paso a producción. Ejemplo: el cliente evalúa si puede realizar sus tareas con la aplicación final.
* **4. Pruebas Específicas para Aplicaciones Web**
* **Pruebas de Rendimiento / Carga / Estrés / Volumen**:
* Evalúan la capacidad del sistema bajo distintos niveles de carga.
* Ejemplo: simular 1000 usuarios concurrentes o un gran volumen de datos procesados.
* **Pruebas de Múltiples Sitios**:
* Verifican consistencia y comunicación entre instalaciones geográficamente separadas.
* Ejemplo: validar sincronización entre sedes de una empresa.
* **Pruebas de Recuperación y Tolerancia a Fallos**:
* Verifican restauración ante fallos de hardware, red o software.
* Ejemplo: simular una caída del sistema y comprobar la recuperación de datos.
* **Pruebas de Integridad de Datos y Base de Datos**:
* Validan acceso y procesos que no corrompan datos. Se prueban métodos con datos válidos e inválidos.
* Ejemplo: ejecutar operaciones CRUD asegurando integridad.
* **Pruebas de Compatibilidad y Conversión**:
* Verifican que conversiones de datos y compatibilidad entre sistemas sean exitosas.
* Ejemplo: importar registros de un sistema antiguo.
* **Pruebas del Ciclo del Negocio**:
* Simulan operaciones a lo largo del tiempo para evaluar comportamiento del sistema.
* Ejemplo: emular transacciones durante un año fiscal.
* **Pruebas de Seguridad y Control de Acceso**:
* Evalúan restricciones por usuario, roles y accesos remotos.
* Ejemplo: evitar que un usuario sin permisos acceda a información sensible.
* **Pruebas de Configuración**:
* Validan el funcionamiento del sistema en diferentes entornos de hardware/software.
* Ejemplo: abrir documentos mientras se ejecuta la aplicación en diferentes sistemas.
* **Pruebas de GUI (Interfaz de Usuario)**:
* Evalúan menús, botones, navegación y consistencia visual. Se prueba estado, foco y medidas.
* **Pruebas de Estilo**:
* Verifican cumplimiento de estándares del cliente: colores, tipografías, formatos.
* **Pruebas de Documentación y Procedimientos**:
* Evalúan si la documentación del usuario es clara, útil y coherente con el sistema.
* **Pruebas de Instalación**:
* Verifican correcta instalación o actualización en distintas configuraciones.
* **Pruebas de Usabilidad**:
* Evalúan facilidad de uso, recuperación de errores y simplicidad de los procedimientos.
* **Pruebas Funcionales**:
* Verifican que el sistema responde correctamente a entradas válidas e inválidas.
* **Pruebas Alfa**:
* Se realizan internamente en ambientes controlados para detectar errores antes del despliegue.
* **Pruebas de Campo**:
* Se ejecutan en el entorno real de operación para validar comportamiento en producción.
* **Pruebas Beta**:
* Realizadas por usuarios finales en condiciones reales antes del lanzamiento final.
* **5. Validación de HTML/CSS y Herramientas del Navegador**
* Validación del código transmitido por el servidor para garantizar correcta visualización.
* Uso de herramientas del navegador (DevTools) para inspección de red, rendimiento, DOM, etc.
* Es esencial garantizar que el HTML/CSS sean válidos para evitar errores de renderizado en el cliente.
* **6. Caja Negra vs Caja Blanca**
* **Caja Negra**: Se prueba el sistema desde afuera (entradas y salidas), sin conocimiento del código.
* **Caja Blanca**: Se prueba conociendo el código interno, evaluando estructuras y lógica de programación