🧪 Tipos de Pruebas Funcionales y No Funcionales

En el ámbito de Quality Assurance, las pruebas se dividen principalmente en **funcionales** y **no funcionales**. Ambas categorías son esenciales para garantizar que el software no solo cumpla con lo que debe hacer, sino que también lo haga de forma eficiente, segura y fiable.

En esta presentación os enseñare alguno de las pruebas que he visto en mi día a día o que he descubierto buscando mas información.

📌 By: **Francis Paneque – QA Specialist** ✉️ francisco.j.paneque@gmail.com

👍 Pruebas Funcionales

Las pruebas funcionales evalúan qué hace el sistema y si cumple los requisitos definidos. Se basan en entradas, salidas y comportamiento esperado.

Smoke Testing

Son pruebas rápidas para verificar que las funciones críticas están operativas tras una nueva versión.

Ejemplo: Después de un despliegue, verificar que los usuarios pueden iniciar sesión, registrarse y acceder al panel principal sin errores.

Beneficio clave: Detecta problemas graves de forma temprana, ahorrando tiempo y recursos en pruebas más extensas de un sistema fundamentalmente defectuoso.

Sanity Testing

Verifica una funcionalidad o corrección específica antes de hacer pruebas más extensas.

Ejemplo: Validar que un error corregido en el módulo de transferencias se ha solucionado correctamente sin profundizar en otras áreas.

Diferencia con Smoke Testing: El Smoke Testing verifica la estabilidad general, mientras que el Sanity Testing se enfoca en validar áreas específicas modificadas.

Pruebas de Regresión

Confirma que los cambios o correcciones no afectan a funcionalidades existentes que ya funcionaban correctamente.

Ejemplo: Después de modificar el sistema de login, revisar que los pagos, transferencias y otras funciones siguen operando normalmente.

Estrategia eficiente: Utilizar automatización para ejecutar un conjunto de casos de prueba críticos después de cada cambio significativo.

👍 Más Pruebas Funcionales

Pruebas de Integración

Comprueban que los módulos o componentes funcionan correctamente juntos, validando la comunicación entre ellos.

Ejemplo: Verificar que el módulo de autenticación se comunica correctamente con la base de datos de usuarios y con el sistema de notificaciones.

Enfoques: Integración ascendente (bottom-up), descendente (top-down) o sándwich (combinación de ambos).

Pruebas de Caja Negra y Caja Blanca

Caja Negra: Se centran en la funcionalidad sin conocer el código interno. Ejemplo: Validar que introducir datos incorrectos muestra el error esperado.

Caja Blanca: Se centran en la lógica interna del código. Ejemplo: Verificar que todas las ramas condicionales de una función de validación se ejecutan correctamente.

Complementariedad: Ambos enfoques son necesarios para una cobertura completa. La caja negra verifica resultados, la caja blanca verifica implementación.

UAT (User Acceptance Testing)

Validación realizada por el cliente o usuario final antes de ir a producción para confirmar que el software cumple sus necesidades reales.

Ejemplo: Los usuarios finales de una aplicación bancaria prueban la nueva función de pagos recurrentes en un entorno controlado.

Importancia crítica: Es la última línea de defensa antes del lanzamiento y la más cercana al uso real del sistema.

Pruebas de Sistema

Evalúan el sistema completo como un todo, incluyendo hardware y software en un entorno que simula el de producción.

Ejemplo: Probar el flujo completo de operaciones bancarias en un entorno de staging, desde el login hasta la generación de informes.

Alcance: Incluye tanto aspectos funcionales como no funcionales, verificando que el sistema cumple con todos los requisitos especificados.

Pruebas de Compatibilidad

Confirman que la aplicación funciona correctamente en distintos dispositivos, navegadores o sistemas operativos.

Ejemplo: Validar que una aplicación web bancaria funciona correctamente en Chrome, Firefox, Safari y Edge, así como en diferentes tamaños de pantalla.

Matrices de prueba: Se suelen utilizar matrices que combinan navegadores, sistemas operativos y dispositivos para garantizar una cobertura completa.

⚙️ Pruebas No Funcionales

Las pruebas no funcionales evalúan cómo funciona el sistema: rendimiento, seguridad, experiencia de usuario, entre otros aspectos críticos para la calidad del software.

Pruebas de Performance

Miden tiempo de respuesta, estabilidad y uso de recursos bajo diferentes condiciones.

Ejemplo: Verificar que 500 usuarios simultáneos pueden hacer login en menos de 2 segundos y que el consumo de CPU no supera el 70%.

Métricas clave: Tiempo de respuesta, throughput (transacciones por segundo), utilización de recursos (CPU, memoria, red) y escalabilidad.

Pruebas de Carga (Load Testing)

Evalúan el rendimiento bajo una carga esperada, simulando el uso real del sistema en condiciones normales o pico.

Ejemplo: Simular las operaciones de pago de 1000 clientes durante las horas punta para verificar que el sistema mantiene los tiempos de respuesta aceptables.

Herramientas: JMeter, Gatling, LoadRunner y k6 son algunas de las más utilizadas para generar carga controlada.

Pruebas de Estrés (Stress Testing)

Evalúan el comportamiento bajo cargas extremas o por encima de lo esperado para identificar puntos de ruptura.

Ejemplo: Forzar 10.000 transacciones simultáneas para determinar el límite del sistema y verificar que falla de manera elegante (graceful degradation).

Objetivo: No solo identificar límites, sino también verificar recuperación tras sobrecarga y prevenir pérdida de datos.

⚙️ Más Pruebas No Funcionales

Pruebas de Seguridad

Detectan vulnerabilidades, accesos no autorizados y fallos de cifrado para proteger datos e infraestructura.

Ejemplo: Intentar acceder a cuentas sin autenticación, realizar inyecciones SQL o ejecutar ataques de cross-site scripting (XSS).

Técnicas: Análisis estático de código, escaneo de vulnerabilidades, pruebas de penetración y revisión de configuraciones.

Pruebas de Usabilidad

Miden la facilidad de uso y experiencia del usuario, enfocándose en la interacción humano-computadora.

Ejemplo: Evaluar si un cliente puede completar una transferencia en menos de 3 pasos o si encuentra intuitivamente las funciones principales.

Métodos: Observación de usuarios reales, análisis de patrones de navegación, cuestionarios de satisfacción y evaluación heurística.

Pruebas de Recuperación (Recovery Testing)

Validan la capacidad del sistema para recuperarse tras fallos, garantizando continuidad del servicio.

Ejemplo: Simular una caída de servidor y verificar que la base de datos no se corrompe y que las transacciones incompletas se restauran correctamente.

Escenarios: Fallos de hardware, cortes de energía, saturación de red o errores de software que provocan terminación abrupta.

Pruebas de Escalabilidad

Evalúan si el sistema puede crecer en usuarios, datos o transacciones sin degradar su rendimiento.

Ejemplo: Medir si el sistema mantiene tiempos de respuesta aceptables al duplicar gradualmente el número de usuarios o el volumen de datos.

Estrategias: Escalado vertical (más potencia) vs horizontal (más instancias), validando elasticidad y eficiencia en ambos enfoques.

Pruebas de Mantenibilidad

Evalúan la facilidad para actualizar, modificar y corregir el sistema a lo largo del tiempo.

Ejemplo: Validar que aplicar un cambio en la interfaz de usuario no requiere modificaciones extensas en el backend debido a un acoplamiento excesivo.

Aspectos clave: Modularidad, legibilidad del código, documentación, cobertura de pruebas y deuda técnica.

📊 Comparativa de Tipos de Pruebas Funcionales y No Funcionales

Smoke Testing	Funcional	Verificar rápidamente que las funciones críticas están operativas.	Probar login y acceso al dashboard tras un despliegue.
Sanity Testing	Funcional	Validar un fix o funcionalidad específica antes de pruebas completas.	Confirmar que un bug en transferencias fue corregido.
Regresión	Funcional	Garantizar que cambios no rompen funcionalidades existentes.	Revisar pagos tras modificar el login.
Integración	Funcional	Validar que módulos funcionan correctamente juntos.	Login + base de datos + dashboard.
Sistema	Funcional	Probar el sistema completo como un todo.	Flujo completo de operaciones bancarias.
Caja Negra	Funcional	Evaluar funcionalidad sin ver el código interno.	Probar importes negativos en transferencias.
Caja Blanca	Funcional	Revisar la lógica interna y cobertura de código.	Validar funciones de validación de importes.
UAT	Funcional	Validación final por cliente/usuario.	Cliente prueba pagos recurrentes antes del despliegue.
Compatibilidad	Funcional	Asegurar que la app funciona en varios entornos.	Probar en Chrome, Firefox y Safari.
Performance	No Funcional	Medir velocidad, respuesta y estabilidad.	500 usuarios simultáneos logueados.
Carga (Load)	No Funcional	Evaluar bajo carga esperada.	1000 pagos en hora punta.
Estrés (Stress)	No Funcional	Medir rendimiento bajo carga extrema.	10.000 transacciones simultáneas.
Seguridad	No Funcional	Detectar vulnerabilidades.	Intentar acceder a cuentas sin login.
Usabilidad	No Funcional	Evaluar experiencia de usuario.	Transferencia en menos de 3 pasos.
Recuperación	No Funcional	Verificar recuperación tras fallos.	Simular caída de servidor.
Escalabilidad	No Funcional	Medir capacidad de crecimiento.	Duplicar usuarios y medir respuesta.
Mantenibilidad	No Funcional	Evaluar facilidad de mantenimiento y actualización.	Cambiar interfaz sin romper arquitectura.