Requisitos No Funcionales y Pruebas: Perspectivas Pre y Post Instalación para Usuarios Finales

Introducción

La calidad de los sistemas de software modernos depende tanto de la funcionalidad que ofrecen como de la manera en que operan bajo diversas condiciones (.). Los requisitos no funcionales (RNF) definen estos atributos de calidad, como seguridad, rendimiento y escalabilidad (.). Su correcta gestión y verificación son cruciales para la satisfacción del usuario y el éxito del producto (.). Este documento explora la aplicación y medición de estos requisitos no funcionales en las fases previas y posteriores a la instalación del sistema para los usuarios finales.

Categorías de Requisitos No Funcionales

Los requisitos no funcionales se clasifican en atributos de calidad (seguridad, rendimiento, usabilidad) y restricciones (tiempo, recursos, entorno) (.).

Aplicación y Medición de Métricas de Requisitos No Funcionales: Antes y Después de la Instalación

A continuación, se detalla cómo se abordan los diferentes tipos de requisitos no funcionales y sus métricas asociadas en las fases previas y posteriores a la instalación.

1. Requerimientos de Fiabilidad

Métrica: Tiempo medio entre fallos (MTBF).

Antes de la instalación:

- **Diseño:** Implementar mecanismos de manejo de errores, redundancia, modularidad.
- **Pruebas:** Pruebas unitarias y de integración para asegurar la robustez de los componentes. Simulaciones de posibles fallos a nivel de diseño.

Después de la instalación:

- **Pruebas:** Pruebas de confiabilidad (operación prolongada sin fallos), pruebas de recuperación y respaldo (capacidad de recuperación ante fallos). Medición del MTBF en entornos de prueba y producción.
- **Monitoreo:** Seguimiento continuo de fallos en producción para calcular el MTBF real.
- **Herramientas:** Selenium, Nagios (para pruebas y monitoreo post-instalación).

2. Requerimientos de Disponibilidad

Métrica: Índice de disponibilidad.

- **Antes de la instalación:**

- **Diseño:** Arquitecturas tolerantes a fallos, estrategias de redundancia de hardware y software.
- **Pruebas:** Simulación de fallos de componentes en entornos de prueba.

- **Después de la instalación:**

- **Pruebas:** Pruebas de conmutación por error (failover) y recuperación. Medición del tiempo de actividad y tiempo de inactividad para calcular el índice de disponibilidad.
- **Monitoreo:** Monitoreo continuo de la disponibilidad del sistema en producción.

- **Herramientas:** Nagios, herramientas de monitoreo de infraestructura.

3. Requerimientos de Mantenibilidad

- **Métrica:** Facilidad de reparación (tiempo y esfuerzo requerido para realizar reparaciones).

- **Antes de la instalación:**

- **Diseño:** Diseño modular, documentación clara, estándares de codificación.
- **Pruebas:** Revisiones de código para identificar posibles puntos de complejidad.

- **Después de la instalación:**

- **Medición:** Registro del tiempo necesario para resolver incidentes y aplicar parches. Evaluación de la facilidad con la que el equipo de soporte puede mantener el sistema.

4. Requerimientos de Eficiencia

- **Métrica:** Capacidad de procesamiento, tasa de utilización de recursos (tiempo de respuesta, transacciones por segundo, consumo de CPU, memoria, red, disco).

- **Antes de la instalación:**

- **Diseño:** Selección de algoritmos eficientes, optimización de consultas a bases de datos.
- **Pruebas:** Pruebas de rendimiento tempranas en entornos de prueba para identificar cuellos de botella. Análisis del consumo de recursos en simulaciones.

- **Después de la instalación:**

- **Pruebas:** Pruebas de carga, estrés y rendimiento para medir tiempos de respuesta, throughput y consumo de recursos bajo diferentes cargas.
- **Monitoreo:** Monitoreo continuo del rendimiento y el consumo de recursos en producción.

- **Herramientas:** Apache JMeter, New Relic (para pruebas y monitoreo).

5. Requerimientos de Usabilidad

Durante estas pruebas, es fundamental recopilar feedback tanto cualitativo (observaciones sobre las dificultades que experimentan los usuarios, sus comentarios y sugerencias) como cuantitativo (métricas como la tasa de éxito de las tareas que se les pide realizar, el tiempo que tardan en completarlas, el número de errores que cometen y su nivel de satisfacción general). Este feedback directo de los usuarios internos es invaluable para identificar áreas específicas de la interfaz de usuario o la funcionalidad del sistema que necesitan mejoras para garantizar una experiencia de usuario positiva y eficiente para los usuarios finales

- **Métrica:** Escala de usabilidad (SUS), facilidad de uso, satisfacción del usuario.
  - **Antes de la instalación:**
    - **Diseño:** Diseño de interfaz intuitivo, pruebas con prototipos y wireframes.
    - **Pruebas:** Pruebas de usabilidad con usuarios internos o representativos en prototipos o versiones tempranas del sistema. Evaluaciones heurísticas.
  - **Después de la instalación:**
    - **Pruebas:** Pruebas de usuario beta para recopilar feedback sobre la usabilidad del sistema instalado. Encuestas de satisfacción del usuario. Análisis de métricas de uso (ej., tiempo para completar tareas, tasa de errores).
- **Herramientas:** Usabilla, Optimizely (para pruebas y análisis de usabilidad).

6. Requerimientos de Portabilidad

- **Métrica:** Portabilidad = 1 - (ET/ER).
  - **Antes de la instalación:**
    - **Diseño:** Adherencia a estándares, abstracción de dependencias del sistema operativo.
    - **Pruebas:** Pruebas en diferentes entornos (sistemas operativos, navegadores) si es posible antes de la instalación final.
  - **Después de la instalación:**
    - **Pruebas:** Verificación del funcionamiento correcto en los entornos objetivo. Medición del esfuerzo requerido para migrar a nuevos entornos si es necesario.
    - **7. Requerimientos de Rendimiento y Escalabilidad**
    - Para asegurar que el sistema pueda manejar las demandas de los usuarios finales y mantener un rendimiento óptimo, es fundamental realizar pruebas de rendimiento y escalabilidad.
    - **Pruebas de Carga (Load Testing)**

- Las pruebas de carga tienen como objetivo primordial evaluar el comportamiento del sistema bajo la carga de usuarios concurrentes esperada, midiendo métricas cruciales como los tiempos de respuesta, el throughput (la cantidad de transacciones que el sistema puede procesar en un período de tiempo) y la utilización de los recursos del servidor. Estas pruebas son esenciales para asegurar que el sistema pueda manejar el volumen de tráfico previsto sin que el rendimiento se degrade de manera significativa, afectando la experiencia del usuario.
- Para simular la carga de usuarios esperada de manera realista, se deben utilizar herramientas especializadas que permitan generar un número variable de usuarios virtuales. Estos usuarios virtuales deben ejecutar escenarios de uso típicos del sistema, emulando las acciones que los usuarios reales llevarían a cabo. La definición de perfiles de usuario diversos y la emulación de sus patrones de interacción con el sistema son clave para obtener resultados de prueba significativos.
- Durante la ejecución de las pruebas de carga, es fundamental monitorear una serie de métricas clave.[10] El tiempo de respuesta promedio y máximo indica la rapidez con la que el sistema responde a las solicitudes de los usuarios. El throughput, medido en transacciones por segundo (TPS) o peticiones por segundo (RPS), refleja la capacidad del sistema para procesar trabajo. La tasa de errores señala la frecuencia con la que el sistema falla al procesar las solicitudes bajo carga. Finalmente, la utilización de la CPU y la memoria del servidor ayuda a identificar posibles cuellos de botella en los recursos del sistema que podrían estar limitando el rendimiento.
- **Pruebas de Estrés (Stress Testing)**
- Las pruebas de estrés van un paso más allá de las pruebas de carga, buscando evaluar la robustez del sistema llevándolo más allá de sus límites de operación normales.[1] El objetivo crucial de estas pruebas es identificar los puntos de fallo del sistema, determinar su capacidad máxima y observar cómo se comporta y se recupera ante condiciones extremas e inesperadas. Estas pruebas son vitales para asegurar que el sistema no falle catastróficamente ante picos inesperados de carga o en situaciones donde los recursos del servidor son limitados.
- Es importante considerar diversos escenarios de estrés durante estas pruebas.Esto puede incluir la simulación de un número de usuarios concurrentes mucho mayor del esperado, la inyección de grandes cantidades de datos o datos con formatos erróneos, o incluso la reducción drástica de los recursos del servidor, como la CPU, la memoria o el espacio en disco. Estos escenarios ayudan a descubrir vulnerabilidades ocultas y a asegurar que el sistema pueda recuperarse de manera adecuada tras experimentar condiciones de estrés.
- Las métricas clave a observar durante las pruebas de estrés incluyen la tasa de errores, que indicará la frecuencia con la que el sistema falla bajo carga extrema, el tiempo de respuesta bajo estas condiciones, la utilización de los recursos del sistema y, de manera fundamental, el tiempo que tarda el sistema en recuperarse y volver a un estado operativo normal una vez que la condición de estrés ha cesado

8. Requerimientos de Seguridad

Análisis de Vulnerabilidades en el Código

Es fundamental realizar un análisis exhaustivo de las vulnerabilidades presentes en el código ya desarrollado. Para ello, se recomienda utilizar herramientas de análisis estático de código (SAST, por sus siglas en inglés). Estas herramientas examinan el código fuente en busca de patrones que puedan indicar posibles fallos de seguridad, como vulnerabilidades de inyección SQL, cross-site scripting (XSS) y otros problemas comunes de seguridad en las aplicaciones web. Realizar este análisis antes del despliegue permite identificar y corregir estos problemas en una etapa temprana, lo que es mucho más eficiente y menos costoso que hacerlo una vez que el sistema está en producción.

- **Métrica:** Nivel de cumplimiento de estándares de seguridad, cantidad de brechas reportadas, nivel de autenticación, nivel de autorización.
  - **Antes de la instalación:**
    - **Diseño:** Implementación de principios de seguridad desde el diseño (seguridad por diseño).
    - **Pruebas:** Análisis estático de código (SAST) para identificar vulnerabilidades. Pruebas de penetración básicas en entornos de prueba.
  - **Después de la instalación:**
    - **Ejemplo práctico: El sistema deberá ser capaz de evitar ataques de inyección de SQL sistemáticos.**
    - **Pruebas de Seguridad: Pruebas de penetración (reconocimiento, análisis de vulnerabilidades, explotación de vulnerabilidades), pruebas de análisis estático**
    - **Monitoreo:** Monitoreo continuo de logs y alertas de seguridad en producción.
- **Herramientas:** OWASP ZAP, Burp Suite (para pruebas de seguridad).

Métricas Clave de Rendimiento a Monitorear

La siguiente tabla resume las métricas clave de rendimiento que se recomienda monitorear durante cada tipo de prueba no funcional:

Tipo de Prueba	Métricas Clave Recomendadas	Herramientas Sugeridas

Pruebas de Carga	Tiempo de respuesta promedio, Tiempo de respuesta máximo, Throughput (TPS/RPS), Tasa de errores (%), Utilización de CPU (%), Utilización de Memoria (%)	Apache JMeter , k6 , Locust , Gatling.
Pruebas de Estrés	Tasa de errores (%), Tiempo de respuesta bajo carga extrema, Utilización de CPU (%), Utilización de Memoria (%), Tiempo de recuperación del sistema (segundos)	Apache JMeter
Pruebas de Resistencia	Utilización de Memoria (%) (tendencia a lo largo del tiempo), Utilización de CPU (%) (tendencia a lo largo del tiempo), Tiempo de respuesta promedio (estabilidad), Tasa de errores (%) (estabilidad)	Apache JMeter , k6

Pruebas Indispensables de Seguridad (.)

Análisis SAST y pruebas de penetración básicas se realizan antes de la instalación, mientras que pruebas más exhaustivas y monitoreo continuo se llevan a cabo después.

Pruebas Esenciales de Usabilidad (Internas)

Las pruebas de usabilidad con usuarios internos se realizan antes de la instalación para identificar problemas tempranamente. Las pruebas beta con usuarios finales se realizan después.

Métricas Clave de Rendimiento a Monitorear

La tabla original resume las métricas clave durante las pruebas de rendimiento *antes* de la instalación. Después de la instalación, estas métricas se siguen monitoreando en producción.

Herramientas Sugeridas para la Ejecución de Pruebas

Las herramientas mencionadas se utilizan tanto en las pruebas previas a la instalación como en el monitoreo post-instalación.

Conclusiones y Próximos Pasos (.)

La evaluación continua de los requisitos no funcionales antes y después de la instalación es esencial para garantizar la calidad del software.

En esta tarea, los estudiantes, deben identificar sus requisitos no funcionales clave y diseñar y ejecutar pruebas para evaluar su cumplimiento. Se espera que los estudiantes investiguen apliquen los conceptos y técnicas aprendidos , incluyendo la selección de métricas adecuadas, la configuración de entornos de prueba y la utilización de herramientas relevantes.

Entregables:

1. **Informe de Requisitos No Funcionales:**
    - Descripción del sistema de software seleccionado.
    - Identificación de al menos 5 requisitos no funcionales clave para el sistema, cubriendo diferentes categorías (ej., fiabilidad, rendimiento, seguridad, usabilidad).
    - Justificación de la importancia de cada requisito no funcional para el sistema y sus usuarios.
    - Para cada requisito, especificar las métricas que se utilizarán para evaluar su cumplimiento.
2. **Plan de Pruebas No Funcionales:**

- Para cada requisito no funcional, diseñar al menos 2 casos de prueba que permitan evaluar su cumplimiento.
- Descripción detallada de cada caso de prueba, incluyendo:
  - Objetivo de la prueba.
  - Entorno de prueba requerido (hardware, software, datos).
  - Procedimiento de prueba (pasos a seguir).
  - Criterios de éxito/fracaso (resultados esperados).
- Identificación de las herramientas que se utilizarán para ejecutar las pruebas (si aplica).

3. **Informe de Resultados de Pruebas:**
   - Descripción del entorno de pruebas utilizado.
   - Resultados obtenidos al ejecutar cada caso de prueba.
   - Análisis de los resultados:
     - Indicar si el sistema cumple o no con los requisitos no funcionales evaluados.
     - Identificar posibles problemas o áreas de mejora en el sistema.
     - Proponer recomendaciones para mejorar el cumplimiento de los requisitos no funcionales.
4. **Presentación**
   - Presentación oral de los resultados de la tarea, destacando los hallazgos más importantes y las recomendaciones propuestas.

Bibliografía

Pruebas no funcionales: proceso, herramientas, tipos y mucho más.