Grupo: C1.015

Repositorio: https://github.com/jormunrod/Acme-SF-24.1.1

Testing Report

Alejandro Pérez Santiago (alepersan3@alum.us.es)

Sevilla, 27-5-2024

Contenido

1. Resumen ejecutivo

El presente informe detalla las pruebas funcionales y de rendimiento ejecutadas en el proyecto, centradas en las auditorías de código y registros de auditoría. Las pruebas funcionales lograron una cobertura del 93.9% para las auditorías de código y del 91.6% para los registros de auditoría, garantizando la integridad y seguridad del sistema con la detección de muy pocos problemas. En cuanto a las pruebas de rendimiento, se comparó el desempeño del sistema en dos computadoras diferentes, determinando que la Computadora B, equipada con un procesador Ryzen 5, presentó tiempos de respuesta significativamente más rápidos que la Computadora A con procesador Intel i7.

2. Tabla de revisión

Nº Revisión	Fecha	Descripción
1	27/05/2024	Versión inicial

3. Introducción

Este documento presenta un análisis detallado de las pruebas funcionales y de rendimiento realizadas en el proyecto, enfocándose en dos áreas principales: auditorías de código y registros de auditoría. Las pruebas funcionales abarcaron un amplio rango de casos, asegurando que las operaciones de listar, mostrar, crear, actualizar, publicar y eliminar auditorías y registros se ejecuten correctamente, además de garantizar la seguridad contra accesos no autorizados. Las pruebas de rendimiento, por su parte, evaluaron el tiempo de respuesta del sistema en dos configuraciones de hardware diferentes, proporcionando un análisis estadístico riguroso para determinar cuál de las dos computadoras ofrece un mejor desempeño.

4. Contenidos

4.1. Pruebas Funcionales

4.1.1. Listado de Casos de Prueba Implementados

Funcionalidad: Auditorías de código, con los siguientes casos de prueba se cuenta con una cobertura de 93,9%:

- **Caso de Prueba 1 (list.safe)**: Prueba de listado de auditorías de código.
  Descripción: Se verifica que un auditor pueda listar sus auditorías de código, incluyendo el paginado.
  Efectividad: Se detectó 0 bugs

- **Caso de Prueba 2 (show.safe)**: Prueba de vista detallada de una auditoría de código.
  Descripción: Se verifica que un auditor pueda ver los detalles de una auditoría de código.
  Efectividad: Se detectó 0 bugs.

- **Caso de Prueba 3 (create.safe)**: Prueba de creación de una auditoría de código.
  Descripción: Se verifica que un auditor pueda crear una nueva auditoría de código. Se comprueban todas las restricciones individualmente, primero las no permitidas y posteriormente se han creado auditorías de código con valores límites en sus atributos.
  Efectividad: Se detectó la necesidad de un validador más para el atributo “execution”.

- **Caso de Prueba 4 (update.safe)**: Prueba de actualización de una auditoría de código.
  Descripción: Se verifica que un auditor pueda actualizar una auditoría de código no publicada. Como en la prueba anterior, se comprueban todas las restricciones de los atributos individualmente, tanto las permitidas para el manejo de fallos, como los límites permitidos.
  Efectividad: Se detectó 0 bugs.

- **Caso de Prueba 5 (publish.safe)**: Prueba de publicación de una auditoría de código.
  Descripción: Se verifica que un auditor pueda publicar una auditoría de código propia sin publicar. Se comprueba, como anteriormente, el manejo de todas las excepciones para todos los atributos, así como, los valores límites permitidos.
  Efectividad: Se detectó 0 bugs.

- **Caso de Prueba 6 (delete.safe)**: Prueba de eliminación de una auditoría de código.
  Descripción: Se verifica que un auditor pueda eliminar una auditoría de código propia sin publicar.
  Efectividad: Se detectó 0 bugs.

- **Caso de Prueba 7 (list.hack)**: Prueba de vulnerabilidad de listado de auditorías de código.
  Descripción: Se verifica que un usuario con un rol distinto a auditor no pueda listar las auditorías de código de otro auditor registrado.
  Efectividad: Se detectó 0 bugs.

- **Caso de Prueba 8 (show.hack)**: Prueba de vulnerabilidad de los detalles de una auditoría de código.
  Descripción: Se verifica que un usuario con un rol distinto a auditor no pueda ver los detalles de una auditoría de código concreta.
  Efectividad: Se detectó 0 bugs.

- **Caso de Prueba 9 (create.hack):** Prueba de vulnerabilidad de la creación de una auditoría de código.
  Descripción: Se verifica que un usuario con un rol distinto a auditor no pueda acceder al formulario de creación de auditorías de código.
  Efectividad: Se detectó 0 bugs.

- **Caso de Prueba 10 (update.hack)**: Prueba de vulnerabilidad de la actualización de una auditoría de código.
  Descripción: Se verifica que el sistema no permita a un usuario con un rol distinto a auditor acceder al formulario de actualización de una auditoría de código. Además, se comprueba que un usuario con rol auditor no pueda actualizar una auditoría de código de otro auditor y que no pueda actualizar una auditoría de código propia ya publicado.
  Efectividad: Se detecto 0 bugs.

- **Caso de Prueba 11 (publish.hack)**: Prueba de vulnerabilidad de la publicación de una auditoría de código.
  Descripción: Se verifica que el sistema no permita a un usuario con rol distinto a auditor acceder al formulario de publicación de una auditoría de código. Además, se comprueba que un usuario con rol auditor no pueda publicar una auditoría de código que no le pertenece y que no pueda volver a publicar una auditoría de código propia ya publicado.
  Efectividad: Se detecto 0 bugs.

- **Caso de Prueba 12 (delete.hack)**: Prueba de vulnerabilidad de la eliminación de una auditoría de código. **Descripción**: Se verifica que el sistema no permita a un usuario con rol distinto a auditor acceder a la eliminación de una auditoría de código. Además, se comprueba que un usuario con rol auditor no pueda eliminar una auditoría de código que no le pertenece, ni eliminar una auditoría de código propia ya publicado.
  Efectividad: Se detectó 0 bugs.

Funcionalidad: Registros de auditoría, con los siguientes casos de prueba se cuenta con una cobertura de 91,6%:

- **Caso de Prueba 1 (list.safe)**: Prueba de listado de registros de auditoría.
  Descripción: Se verifica que un auditor pueda listar sus registros de auditoría, incluyendo el paginado.
  Efectividad: Se detectó 0 bugs

- **Caso de Prueba 2 (show.safe)**: Prueba de vista detallada de un registro de auditoría.
  Descripción: Se verifica que un auditor pueda ver los detalles de un registro de auditoría.
  Efectividad: Se detectó 0 bugs.

- **Caso de Prueba 3 (create.safe)**: Prueba de creación de un registro de auditoría.
  Descripción: Se verifica que un auditor pueda crear un nuevo registro de auditoría. Se comprueban todas las restricciones individualmente, primero las no permitidas y posteriormente se han creado registros de auditoría con valores límites en sus atributos.
  Efectividad: Se detectó la necesidad de un validador más para el atributo “execution”.

- **Caso de Prueba 4 (update.safe)**: Prueba de actualización de un registro de auditoría.
  Descripción: Se verifica que un auditor pueda actualizar un registro de auditoría no publicado. Como en la prueba anterior, se comprueban todas las restricciones de los atributos individualmente, tanto las permitidas para el manejo de fallos, como los límites permitidos.
  Efectividad: Se detectó 0 bugs.

- **Caso de Prueba 5 (publish.safe)**: Prueba de publicación de un registro de auditoría.
  Descripción: Se verifica que un auditor pueda publicar un registro de auditoría propio sin publicar. Se comprueba, como anteriormente, el manejo de todas las excepciones para todos los atributos, así como, los valores límites permitidos.
  Efectividad: Se detectó la carga errónea del formulario cuando se intentaba publicar un proyecto con algún campo incorrecto.

- **Caso de Prueba 6 (delete.safe)**: Prueba de eliminación de un registro de auditoría.
  Descripción: Se verifica que un auditor pueda eliminar un registro de auditoría propio sin publicar.
  Efectividad: Se detectó 0 bugs

- **Caso de Prueba 7 (list.hack)**: Prueba de vulnerabilidad de listado de registros de auditoría.
  Descripción: Se verifica que un usuario con un rol distinto a auditor no pueda listar los

registros de auditoría de otro auditor registrado.
Efectividad: Se detectó 0 bugs.

- **Caso de Prueba 8 (show.hack)**: Prueba de vulnerabilidad de los detalles de un registro de auditoría.
  Descripción: Se verifica que un usuario con un rol distinto a auditor no pueda ver los detalles de un registro de auditoría concreto.
  Efectividad: Se detectó 0 bugs.

- **Caso de Prueba 9 (create.hack):** Prueba de vulnerabilidad de la creación de un registro de auditoría.
  Descripción: Se verifica que un usuario con un rol distinto a auditor no pueda acceder al formulario de creación de registros de auditoría.
  Efectividad: Se detectó 0 bugs.

- **Caso de Prueba 10 (update.hack)**: Prueba de vulnerabilidad de la actualización de un registro de auditoría.
  Descripción: Se verifica que el sistema no permita a un usuario con un rol distinto a auditor acceder al formulario de actualización de un registro de auditoría. Además, se comprueba que un usuario con rol auditor no pueda actualizar un registro de auditoría de otro auditor y que no pueda actualizar un registro de auditoría propio ya publicado.
  Efectividad: Se detecto 0 bugs.

- **Caso de Prueba 11 (publish.hack)**: Prueba de vulnerabilidad de la publicación de un registro de auditoría.
  Descripción: Se verifica que el sistema no permita a un usuario con rol distinto a auditor acceder al formulario de publicación de un registro de auditoría. Además, se comprueba que un usuario con rol auditor no pueda publicar un registro de auditoría que no le pertenece y que no pueda volver a publicar un registro de auditoría propio ya publicado.
  Efectividad: Se detecto 0 bugs.

- **Caso de Prueba 12 (delete.hack)**: Prueba de vulnerabilidad de la eliminación de un registro de auditoría.
  Descripción: Se verifica que el sistema no permita a un usuario con rol distinto a auditor acceder a la eliminación de un registro de auditoría. Además, se comprueba que un usuario con rol auditor no pueda eliminar un registro de auditoría que no le pertenece, ni eliminar un registro de auditoría propio ya publicado.
  Efectividad: Se detectó 0 bugs.

4.2. Pruebas de Rendimiento

En este capítulo, se presentan los resultados de las pruebas de rendimiento realizadas en el proyecto. Se ha evaluado el tiempo de respuesta del sistema al atender las solicitudes generadas durante las pruebas funcionales en dos computadoras diferentes. Además, se incluyen análisis estadísticos con intervalos de confianza y un contraste de hipótesis para determinar cuál de las dos computadoras es más potente.

4.2.1. Metodología

Para las pruebas de rendimiento, se llevaron a cabo las siguientes acciones:

- Entorno de Pruebas: Las pruebas se realizaron en dos computadoras con especificaciones diferentes. La Computadora A es un ordenador portátil con un procesador Intel i7 y 16GB de RAM. La Computadora B es un ordenador portátil con un procesador Ryzen 5 y 16GB de RAM.

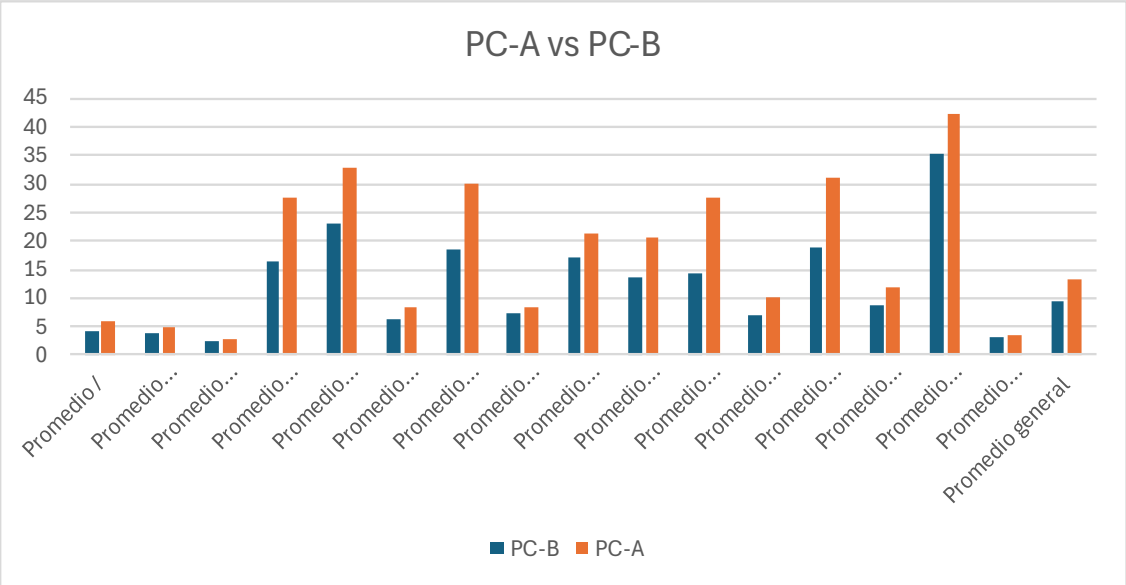
- Tiempos de Respuesta: Se midieron los tiempos de respuesta (wall time) en milisegundos para cada solicitud enviada durante las pruebas funcionales.

- Número de Solicitudes: Se realizaron un total de 488 solicitudes en cada computadora para garantizar la representatividad de los datos.

4.2.2. Resultados de las Pruebas de Rendimiento

4.2.2.1. Gráfica de Tiempo de Respuesta

La siguiente gráfica muestra la comparativa de los tiempos de respuesta registrados en ambas computadoras:



4.2.2.2. Análisis de Intervalo de Confianza

Se ha calculado el intervalo de confianza del 95% para los tiempos de respuesta en ambas computadoras:

Computadora A:

- Tiempo de respuesta promedio: 13,30 ms

- Intervalo de confianza del 95%: [11,95 - 14,66] ms

Computadora B:

- Tiempo de respuesta promedio: 8,71 ms

- Intervalo de confianza del 95%: [7,93- 9,49] ms

4.2.2.3. Contraste de Hipótesis

Para determinar cuál de las dos computadoras es más potente, se realizó un Z-test con un nivel de confianza del 95%.

Hipótesis:

- H0 (Hipótesis nula): No hay diferencia significativa en el rendimiento entre la Computadora A y la Computadora B.

- H1 (Hipótesis alternativa): La Computadora A es significativamente más rápida que la Computadora B.

Resultados del Z-test:

- Se compararon las medias de los tiempos de respuesta de ambas computadoras.

- El valor p obtenido fue $7{,}77 \times 10^{-9}$, lo cual es mucho menor al nivel de significancia 0’05.

Dado que el valor p es mucho menor a 0.05, rechazamos la hipótesis nula H0 y aceptamos la hipótesis alternativa H1. Por lo tanto, concluimos que la Computadora B es significativamente más rápida que la Computadora A en términos de tiempo de respuesta.

Conclusión:

Las pruebas de rendimiento muestran que la Computadora b tiene un mejor desempeño en comparación con la Computadora a, con tiempos de respuesta más rápidos y una menor variabilidad. En base a estos hallazgos, se recomienda utilizar la Computadora b para tareas críticas que requieran tiempos de respuesta mínimos.

5. Conclusiones

Las pruebas funcionales de auditorías de código y registros de auditoría demostraron una alta fiabilidad del sistema, con una cobertura superior al 90% y la detección de pocos problemas significativos, destacando únicamente la necesidad de un validador adicional para el atributo "execution" y la carga errónea del formulario en casos específicos. Las pruebas de rendimiento revelaron que la Computadora B, con un procesador Ryzen 5, supera significativamente a la Computadora A en términos de tiempos de respuesta, como lo demuestra un valor p extremadamente bajo en el Z-test realizado. En conclusión, se recomienda utilizar la Computadora B para tareas críticas que requieran tiempos de respuesta mínimos, mientras se continúa optimizando el sistema con base en los resultados obtenidos de las pruebas funcionales.

6. Bibliografía

Intencionalmente en blanco.