Universidad de Sevilla  
Escuela Técnica Superior de Ingeniería Informática



Grado en Ingeniería Informática – Ingeniería del Software  
Diseño y Pruebas II

**Informe de pruebas.**

Curso 2023 – 2024

**Grupo:** C1.014

**Repositorio**: <https://github.com/alvaroChico2408/Acme-SF-D04>

**Fecha**: 26/05/2024

|  |  |
| --- | --- |
| **Integrante del grupo** | **Correo** |
| Chico Castellano, Álvaro | alvchicas@alum.us.es |

**Historial de versiones**

|  |  |  |
| --- | --- | --- |
| **Versión** | **Fecha** | **Descripción de los cambios** |
| 1.0 | 26/05/2024 | Creación del documento |
| 1.1 | 26/05/2024 | Añadido resumen ejecutivo e introducción |
| 1.2 | 26/05/2024 | Añadido información del apartado contenido |
| 1.3 | 26/05/2024 | Añadido conclusión y bibliografía |

**Índice de contenido**

[**3.** **Contenido** 4](#_Toc167725150)

[**4.** **Conclusiones** 14](#_Toc167725151)

[**5.** **Bibliografía** 14](#_Toc167725152)

1. **Resumen Ejecutivo**

Este informe de pruebas se divide en dos capítulos: uno dedicado a las pruebas funcionales y otro a las pruebas de rendimiento. El capítulo de pruebas funcionales detalla los casos de prueba implementados, evaluando su efectividad en la detección de errores. El capítulo de pruebas de rendimiento incluye gráficos y análisis estadísticos del tiempo de respuesta en dos computadoras distintas, con el objetivo de determinar cuál es la más poderosa. Este informe proporciona una visión integral que facilita la mejora de la calidad y el rendimiento del software.

1. **Introducción**

Realizar un informe de pruebas es fundamental en el desarrollo de software, ya que permite evaluar de manera sistemática y detallada tanto la calidad como el rendimiento del sistema. Este documento está organizado en dos capítulos, cada uno enfocado en aspectos cruciales del proceso de pruebas: funcionalidad y rendimiento.

El capítulo de pruebas funcionales se centra en verificar que cada función del software opere conforme a los requisitos especificados. Incluye un listado detallado de los casos de prueba implementados, agrupados por características del software. Para cada caso de prueba se proporciona una descripción concisa y una evaluación de su efectividad para detectar errores. Esta estructura permite a los desarrolladores y gestores del proyecto identificar rápidamente las áreas del sistema que presentan fallos, priorizar las mejoras necesarias y mantener una documentación clara y accesible sobre las pruebas realizadas y sus resultados. Esto facilita futuras revisiones y auditorías, asegurando que cualquier cambio o actualización en el software se base en un conocimiento profundo y detallado de su comportamiento actual.

El capítulo de pruebas de rendimiento es crucial para entender cómo se comporta el software bajo diferentes cargas y en distintos entornos. Este capítulo incluye un contraste de hipótesis con un 95% de confianza para determinar cuál de las dos computadoras es más poderosa en términos de rendimiento. Este análisis comparativo es esencial para optimizar la infraestructura tecnológica utilizada en el proyecto, asegurando que los recursos se usen de manera eficiente y planificando adecuadamente la capacidad futura del sistema.

El informe de pruebas ofrece una serie de beneficios clave. Proporciona una base sólida para la mejora continua del software al identificar sistemáticamente los errores y evaluar el rendimiento del sistema. Equipara a los gestores del proyecto con datos precisos y relevantes, permitiendo tomar decisiones informadas sobre mejoras y optimizaciones. Además, asegura que el software cumpla con los estándares de calidad esperados antes de su lanzamiento o actualización, garantizando que el sistema sea robusto, eficiente y capaz de enfrentar los desafíos operativos presentes y futuros. En resumen, este informe no solo documenta el estado actual del software, sino que también ofrece información valiosa para su mejora continua.

# **Contenido**

* 1. **Sponsorship**

El objetivo con estos casos de prueba es comprobar que la funcionalidad relacionada con los patrocinios se ha implementado correctamente. Para ello se han implementado dos tipos de pruebas: .safe y .hack.

**.safe:** Se comprueba esta funcionalidad de forma normal, como se debería de realizar normalmente. Probando los límites de cada uno de los atributos, y observando si los errores aparecen de la forma correcta. Se comprueba el formulario vacío, rellenando los campos de uno en uno, y comprobando los valores: inf, inf-1, inf+1, sup, sup-1, sup+1 para cada uno de los atributos. Además, se comprueba que todos los errores aparezcan cuando deben aparecer, y no hay anomalías ni problemas.

**.hack:** Se comprueba la funcionalidad con intenciones maliciosas, y forzando límites que el usuario normal no forzaría. Se cambian URLs intentando acceder a acciones prohibidas, con cuentas no autorizadas o con el rol equivocado. El objetivo es comprobar que la aplicación sea segura y no puedan vulnerar las medidas seguridad de esta.

* + 1. **Sponsorship/create**

El objetivo es comprobar que la funcionalidad de create funciona correctamente. Se han realizado los test según la metodología estudiada, y se ha intentado hackear la aplicación tratando de crear un patrocinio sin ser un patrocinador, lo que ha devuelto un error 500.

Como se observa en el coverage (96,3%) todas las líneas de código de la clase aparecen en verde (no contienen bugs) excepto los asserts, que aparecen en amarillo ya que siempre obtienen el mismo resultado (true). Los asserts deben aparecer en amarillo siempre, ya que la única posibilidad de que devuelvan el otro resultado (false) es porque el framework tenga algún error.

* + 1. **Sponsorship/delete**

El objetivo es comprobar que la funcionalidad de delete funciona correctamente. Se han realizado los test según la metodología estudiada, y se ha intentado hackear la aplicación tratando de eliminar un patrocinio publicado siendo el patrocinador al que pertenece, eliminar un patrocinio publicado y no publicado siendo un patrocinador al que no le pertenece y siendo un rol distinto, e intentando borrar un patrocinio inexistente, lo que ha devuelto un error 500 en todos los casos.

Como se observa en el coverage (91,8%) todas las líneas de código de la clase aparecen en verde (no contienen bugs) excepto los asserts (explicado en Sponsorship/create), la línea 39 que aparece en amarillo, y el unbind que aparece en rojo.

El unbind del delete debe aparecer siempre en rojo, ya que la única forma de que se ejecute el unbind, es si el delete falla por algún error de validación. No tiene sentido validar nada en el delete, ya que el objetivo es eliminarlo, por lo que nunca se ejecutará.

Con respecto a la línea 39, sale en amarillo porque el método HasRole() comprueba la id, por lo que, si la siguiente validación se realiza, significa que el método HasRole() ha devuelto un true, por lo que la línea 39 devolverá true también, y en caso de que HasRole() devuelva false, esa línea no se ejecutará, por lo que siempre será true.

* + 1. **Sponsorship/list**

El objetivo es comprobar que la funcionalidad de list funciona correctamente. Se han realizado los test según la metodología estudiada, y se ha intentado hackear la aplicación tratando de listar los patrocinios sin ser patrocinador.

Como se observa en el coverage (93,9%) todas las líneas de código de la clase aparecen en verde (no contienen bugs) excepto los asserts (explicado en Sponsorship/create).

* + 1. **Sponsorship/publish**

El objetivo es comprobar que la funcionalidad de publish funciona correctamente. Se han realizado los test según la metodología estudiada, y se ha intentado hackear la aplicación tratando de publicar un patrocinio publicado siendo el patrocinador al que pertenece, publicar un patrocinio publicado y no publicado siendo un patrocinador al que no le pertenece y siendo un rol distinto, e intentando publicar un patrocinio inexistente, lo que ha devuelto un error 500 en todos los casos.

Como se observa en el coverage (97,4%) todas las líneas de código de la clase aparecen en verde (no contienen bugs) excepto los asserts (explicado en Sponsorship/create) y las líneas 47 (mismo caso que la línea 39 de Sponsorship/delete ) ,139 ,150 y 175 que aparecen en amarillo.

En el caso de las líneas 139 y 150 siempre serán true, ya que no hay ningún error de tipo “invoice” definido.

En el caso de la línea 175 siempre será true ya que en el sample data existen proyectos publicados y todos los Sponsorship requieren de un proyecto para ser creados, por lo que no se podrá publicar un Sponsorship sin proyecto, ya que es necesario para poder crearlo en primera instancia.

* + 1. **Sponsorship/show**

El objetivo es comprobar que la funcionalidad de show funciona correctamente. Se han realizado los test según la metodología estudiada, y se ha intentado hackear la aplicación tratando de ver los detalles de un patrocinio publicado y no publicado siendo un patrocinador al que no le pertenece y siendo un rol distinto, e intentando mostrar los detalles un patrocinio inexistente, lo que ha devuelto un error 500 en todos los casos.

Como se observa en el coverage (98,1%) todas las líneas de código de la clase aparecen en verde (no contienen bugs) excepto los asserts (explicado en Sponsorship/create) y las líneas 42 (mismo caso que la línea 39 de Sponsorship/delete ) y 72 (mismo caso que la línea 175 de Sponsorship/publish) que aparecen en amarillo.

* + 1. **Sponsorship/update**

El objetivo es comprobar que la funcionalidad de update funciona correctamente. Se han realizado los test según la metodología estudiada, y se ha intentado hackear la aplicación tratando de actualizar un patrocinio publicado siendo el patrocinador al que pertenece, actualizar un patrocinio publicado y no publicado siendo un patrocinador al que no le pertenece y siendo un rol distinto, e intentando actualizar un patrocinio inexistente, lo que ha devuelto un error 500 en todos los casos.

Como se observa en el coverage (97,1%) todas las líneas de código de la clase aparecen en verde (no contienen bugs) excepto los asserts (explicado en Sponsorship/create) y las líneas 47 (mismo caso que la línea 39 de Sponsorship/delete ) y 157 (mismo caso que la línea 175 de Sponsorship/publish) que aparecen en amarillo.

* 1. **Invoice**
     1. **Invoice/create**

El objetivo es comprobar que la funcionalidad de create funciona correctamente. Se han realizado los test según la metodología estudiada, y se ha intentado hackear la aplicación tratando de crear una factura sin ser un patrocinador, lo que ha devuelto un error 500.

Como se observa en el coverage (95,1%) todas las líneas de código de la clase aparecen en verde (no contienen bugs) excepto los asserts (explicado en Sponsorship/create) que aparecen en amarillo.

* + 1. **Invoice/delete**

El objetivo es comprobar que la funcionalidad de delete funciona correctamente. Se han realizado los test según la metodología estudiada, y se ha intentado hackear la aplicación tratando de eliminar una factura publicada siendo el patrocinador al que pertenece, eliminar una factura publicada y no publicada siendo un patrocinador al que no le pertenece y siendo un rol distinto, e intentando borrar una factura inexistente lo que ha devuelto, un error 500 en todos los casos.

Como se observa en el coverage (93,9%) todas las líneas de código de la clase aparecen en verde (no contienen bugs) excepto los asserts (explicado en Sponsorship/create) y la línea 40 (mismo caso que la línea 39 de Sponsorship/delete) que aparece en amarillo.

* + 1. **Invoice/list**

El objetivo es comprobar que la funcionalidad de list funciona correctamente. Se han realizado los test según la metodología estudiada, y se ha intentado hackear la aplicación tratando de listar de las facturas de un patrocinio sin ser el patrocinador al que pertenece, y listar las facturas de un patrocinio inexistente, devolviendo un error 500 en todos los casos.

Como se observa en el coverage (90,9%) todas las líneas de código de la clase aparecen en verde (no contienen bugs) excepto los asserts (explicado en Sponsorship/create), la línea 36 (mismo caso que la línea 39 de Sponsorship/delete) que aparece en amarillo, y el unbind que aparece en rojo (mismo caso que el unbind de Sponsorship/delete).

* + 1. **Invoice/publish**

El objetivo es comprobar que la funcionalidad de publish funciona correctamente. Se han realizado los test según la metodología estudiada, y se ha intentado hackear la aplicación tratando de publicar una factura publicada siendo el patrocinador al que pertenece, publicar una factura publicada y no publicada siendo un patrocinador al que no le pertenece y siendo un rol distinto, e intentando publicar una factura inexistente lo que ha devuelto, un error 500 en todos los casos.

Como se observa en el coverage (96,7%) todas las líneas de código de la clase aparecen en verde (no contienen bugs) excepto los asserts (explicado en Sponsorship/create)y las líneas 40 (mismo caso que la línea 39 de Sponsorship/delete), 115, 127, 131 que aparecen en amarillo.

En el caso de las líneas 115 y 131 siempre serán true, ya que no hay ningún error de tipo “invoices” definido. Y en la línea 127 ocurre igual con “TotalAmount”

* + 1. **Invoice/show**

El objetivo es comprobar que la funcionalidad de show funciona correctamente. Se han realizado los test según la metodología estudiada, y se ha intentado hackear la aplicación tratando de ver los detalles de una factura publicada y no publicada siendo un patrocinador al que no le pertenece y siendo un rol distinto, e intentando mostrar los detalles de una factura inexistente, lo que ha devuelto un error 500 en todos los casos.

Como se observa en el coverage (97,1%) todas las líneas de código de la clase aparecen en verde (no contienen bugs) excepto los asserts (explicado en Sponsorship/create) y la línea 37 (mismo caso que la línea 39 de Sponsorship/delete ) que aparece en amarillo.

* + 1. **Invoice/update**

El objetivo es comprobar que la funcionalidad de update funciona correctamente. Se han realizado los test según la metodología estudiada, y se ha intentado hackear la aplicación tratando de actualizar una factura publicada siendo el patrocinador al que pertenece, actualizar una factura publicada y no publicada siendo un patrocinador al que no le pertenece y siendo un rol distinto, e intentando actualizar una factura inexistente, lo que ha devuelto un error 500 en todos los casos.

Como se observa en el coverage (96,7%) todas las líneas de código de la clase aparecen en verde (no contienen bugs) excepto los asserts (explicado en Sponsorship/create) y la línea 41 (mismo caso que la línea 39 de Sponsorship/delete ) que aparecen en amarillo.

* 1. Interfaz de usuario gráfica, Aplicación

     Descripción generada automáticamente**Comparación**

Tras aplicar los procedimientos indicados en las transparencias, estos han sido los resultados obtenidos:

* Los primeros datos se obtuvieron realizando los test en mí ordenador personal antes de insertar los índices.
* Los segundos datos se obtuvieron realizando los test en mi ordenador personal después de insertar los índices.
* Los últimos datos se obtuvieron realizando los test en el ordenador de mi compañero Jaime después de insertar los índices.

Tras compararlos dos a dos obtenemos los siguientes resultados:

Aplicación

Descripción generada automáticamente con confianza baja

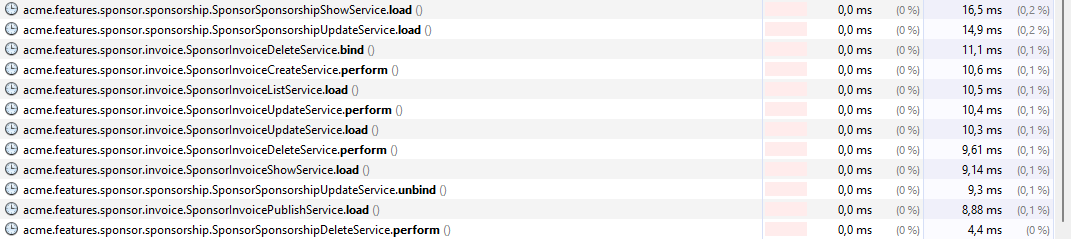
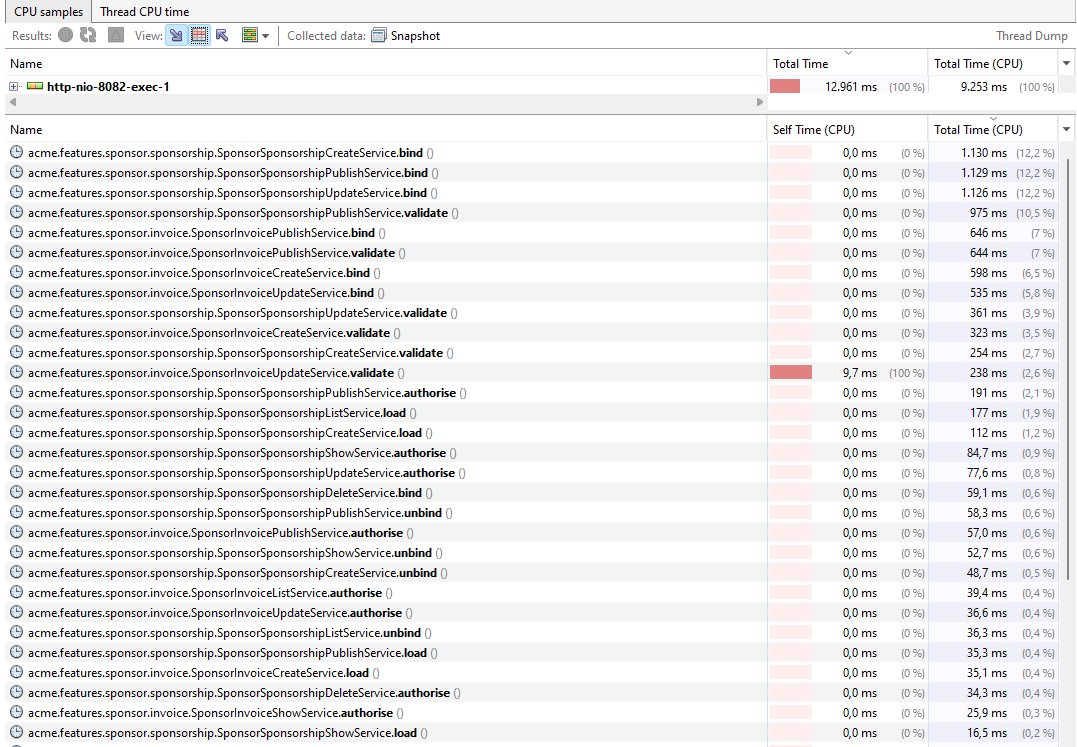
La primera comparativa, se realizó ejecutando los test en mi ordenador personal antes y después de insertar los índices. Como se observa en la fila **Valor crítico de z (dos colas),** el valor está en el intervalo (Alpha, 1] lo que nos indica que no ha habido una gran mejoría.

Sin embargo, en la segunda comparativa, realizada ejecutando los test después de insertar los índices en mi ordenador personal ( comprado recientemente) y el ordenador de Jaime (con una antigüedad de unos 6 años), como se observa en la fila **Valor crítico de z (dos colas),** el valor es igual a 0, es decir, está en el intervalo [0, Alpha), lo que indica que mi ordenador realiza la tarea de forma más rápida y eficiente, lo que nos proporciona una mayor eficacia y velocidad.

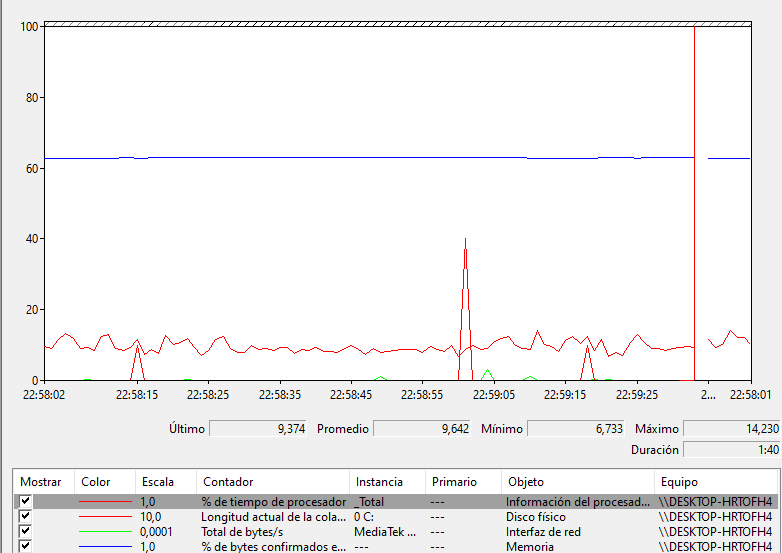
En resumen, la inserción de los índices no ha supuesto una mejora significativa en el rendimiento, sin embargo, mi ordenador ofrece una mayor velocidad con respecto al de mi compañero, aumentando el rendimiento.

* 1. **Hardware y Software**

Tras aplicar los procedimientos explicados en las transparencias, los resultados de monitorizar el software de mi equipo durante la ejecución de los test son:



Y el hardware:



# **Conclusiones**

Tras analizar los test realizados, llegamos a la conclusión de que el coverage es bastante bueno, y debido a las justificaciones y explicaciones de aquellas líneas que no cubre, podemos estar medianamente seguros de que no hay bugs ocultos dentro del código.

Por otro lado, tras hacer la performance, llegamos a la conclusión de que mi ordenador personal es considerablemente más potente que el de mi compañero, lo que provoca un mejor rendimiento.

# **Bibliografía**

Intencionalmente en blanco.