Universidad de Sevilla  
Escuela Técnica Superior de Ingeniería Informática

TESTING REPORT INDIVIDUAL - D04



**C1.027**

**Repositorio**: <https://github.com/josrojrom1/DP2-G27-Acme>

**Autor:**

Ridruejo Pineda, Guadalupe

guaridpin@alum.us.es

**Fecha:** 27/05/2024

**Tabla de contenidos**

[**Resumen 3**](#_heading=h.gjdgxs)

[**Tabla de revisión 3**](#_heading=h.30j0zll)

[**Introducción 4**](#_heading=h.1fob9te)

[**Contenidos 4**](#_heading=h.3znysh7)

[**Capítulo I: Testing funcional 4**](#_heading=h.3k3mzb7frzkc)

[**Capítulo II: Testing de rendimiento 9**](#_heading=h.vr2c6jw2oh94)

[**Bibliografía 11**](#_heading=h.3dy6vkm)

# 

# Resumen

Este documento evalúa el proceso de testing seguido en los requisitos individuales de entregas anteriores, con el objetivo de llegar a una conclusión de los resultados en materia de mejora del código o del mismo proceso de testing. El documento consta de dos capítulos que tratan de los métodos estadísticos realizados para calcular intervalos de confianza y el contraste de hipótesis, alcanzando una valoración final.

# Tabla de revisión

| **Versión** | **Fecha** | **Descripción** |
| --- | --- | --- |
| V1 | 27/05/2024 | Creación y desarrollo del documento |

# 

# Introducción

A continuación, veremos en dos capítulos los métodos estadísticos realizados para analizar los resultados tras el testing formal de los requisitos de todas las entregas del proyecto.

# Contenidos

# Capítulo I: Testing funcional

Este capítulo mostrará el listado con los casos de prueba implementados, agrupados por funcionalidades. Para cada caso de prueba, proporcionará una descripción sucinta y una indicación clara sobre la eficacia de la detección de errores.

**Rol Sponsor sobre entidad Sponsorship (patrocinio)**

**sponsor/sponsorship/create:** para realizar las pruebas de esta funcionalidad se comenzaron haciendo pruebas negativas en el formulario de creación de un patrocinio vacío. Para ello, se envió en primer lugar el formulario vacío y después, uno por uno, se fueron rellenando el resto de campos con todas las mínimas variantes posibles de valores incorrectos, comprobando que se obtenían por pantalla los mensajes de error asociados en las validaciones implementadas. Finalmente se enviaron varios formularios de creación con valores correctos, realizando la prueba positiva correspondiente con diferentes resultados positivos. Además, se realizó la prueba de *hacking* correspondiente para verificar que solo usuarios asignados al rol de *sponsor* pueden crear patrocinios en el sistema.

Se ha logrado una cobertura de un 96.0% de código para la clase *SponsorSponsorshipCreateService*, a excepción de algunos *warnings* relacionados con las líneas de código donde se realizan comprobaciones de que el objeto no es nulo, que no pueden ser suprimidas y, por tanto, se ignoran.

**sponsor/sponsorship/delete:** para realizar esta prueba, se ha ingresado a un formulario de un patrocinio no publicado y se ha eliminado, completando la prueba positiva.

Para la prueba de *hacking*,se verificó que únicamente puedan eliminarse patrocinios que existan en el sistema, tanto si tienen facturas asociadas como si no, y que estén en modo borrador, además de comprobar que únicamente una persona con rol *sponsor* y propiedad del patrocinio pudiera eliminarla. No obstante, la herramienta *inspeccionar*  que proporciona *Firefox* generó una prueba incorrecta al tratar de eliminar un patrocinio desde un rol no autorizado, devolviendo un código de tipo 4xx en lugar de 5xx; esta es la razón por la que la línea principal del método *authorise* de este servicio aparezca en amarillo en lugar de verde. Sin embargo, la restricción para asegurar eliminar un patrocinio desde el rol adecuado está perfectamente implementada.

En cuanto al porcentaje de cobertura, se ha obtenido un valor de 59.4%, principalmente debido a las líneas marcadas en rojo del método *unbind* del servicio de borrado (*SponsorSponsorshipDeleteService*) correspondiente, pues estas en ningún caso se ejecutan. Se ha mantenido el método por coherencia con la estructura del proyecto.

**sponsor/sponsorship/list:** para realizar esta prueba, se ha accedido al listado de patrocinios de un rol *sponsor* comprobando que se lista adecuadamente, completando la prueba positiva de la funcionalidad.

Para la prueba de *hacking*, se han intentado listar los patrocinios desde un usuario con otro rol (en este caso, *manager*) obteniendo el correspondiente error 500. Se ha logrado un porcentaje de cobertura de 92.6% del servicio *SponsorSponsorshipListService*, quedando warnings relacionados con la comprobación de si un objeto es nulo a través de *asserts*, los cuales se ignoran.

**sponsor/sponsorship/publish:** para realizar esta prueba, se ha accedido a un patrocinio de un *sponsor* en modo borrador, es decir, que aún no estuviese publicado. Se cumplió con el mismo procedimiento anteriormente descrito en la funcionalidad de creado, para verificar que el sistema asegura que solo puedan publicarse patrocinios que cumplan con las validaciones indicadas (especialmente la que restringe la publicación para aquellos patrocinios que no puedan ser publicados hasta que su coste esté completado por todas sus facturas publicadas), cumpliendo con las pruebas negativas. Finalmente, se publicó un patrocinio con datos correctos para cumplir con la prueba positiva.

Para la prueba de *hacking*, se intentó publicar un patrocinio inexistente, otro que ya estuviese publicado y realizar la misma acción desde otro *sponsor* que no fuese dueño del patrocinio y también desde otro rol (surgió el mismo conflicto para esta prueba anteriormente descrito), obteniendo el correspondiente error 500.

Se ha obtenido un porcentaje de cobertura de 96.7% del código relacionado con el servicio *SponsorSponsorshipPublishService*, quedando *warnings* señalados debido a las comprobaciones de objetos nulos a través de *asserts,* los cuales se ignoran.

**sponsor/sponsorship/show:** para realizar esta prueba, se ha accedido al listado de patrocinios de un rol *sponsor* y se ha ingresado a una de estos, comprobando que se muestra adecuadamente, completando la prueba positiva de la funcionalidad.

Para la prueba de *hacking*, se han intentado mostrar patrocinios inexistentes, desde otro rol y desde otro usuario *sponsor* que no estuviese relacionado con él .

Se ha conseguido un porcentaje de cobertura de 98.1% del código relacionado con mostrar un patrocinio. Solo han quedado algunos *warnings* marcando comprobaciones de que un objeto no es nulo a través de *asserts*, los cuales se ignoran, y algunas sentencias *if* que realizan cálculos de propiedades de las facturas que no podían ver cumplimentados todos sus caminos de ejecución a partir de los datos de prueba.

**sponsor/sponsorship/update:** para realizar las pruebas de esta funcionalidad se ha comenzado haciendo pruebas negativas en el formulario de de un patrocinio previamente creado, con la misma metodología aplicada para crear y publicar para generar los casos de prueba negativos. Finalmente, se ha enviado el formulario de creación con valores correctos, realizando la prueba positiva correspondiente.

Para la prueba de *hacking* se ha realizado el mismo procedimiento que para publicar. Se ha conseguido un 96.7% de cobertura del código relacionado con *SponsorSponsorshipUpdateService*. Solo han quedado algunos warnings relacionados con comprobaciones de que un objeto no es nulo a través de *asserts*, los cuales se ignoran y algunos caminos de ejecución relacionados con los hijos que no pudieron ser ejecutados con los datos de prueba.

**Rol Sponsor sobre entidad Invoice (factura)**

**sponsor/invoice/create:** para realizar las pruebas de esta funcionalidad se ha comenzado haciendo pruebas negativas en el formulario de creación de una factura de código vacía. Se comenzó enviado el formulario vacío y después uno por uno los campos con valores incorrectos, generando todas las combinaciones mínimas posibles, comprobando que se obtienen por pantalla los correspondientes mensajes de error de las validaciones implementadas. Finalmente se enviaron varios formularios de creación con valores correctos, asociándose a distintos patrocinios, realizando la prueba positiva correspondiente.

Además, se realizó la prueba de *hacking* correspondiente para verificar que solo usuarios asignados al rol de *sponsor* y propietarios de una entidad patrocinio que aún no estuviese publicada pudieran crear patrocinios en el sistema.

Se ha alcanzado un porcentaje de cobertura del código de 96.2%. Solo han quedado warnings relacionados con comprobaciones de un objeto nulo a través de *asserts*, los cuales se ignoran.

**sponsor/invoice/delete:** para realizar esta prueba, se ha ingresado a una factura aún no publicada y se ha eliminado, completando la prueba positiva de dicha funcionalidad.

Para la prueba de *hacking*, se intentó eliminar una factura de otro usuario *sponsor*, así como intentar eliminar una factura publicada, o que pertenciese a un patrocinio publicado o inexistente, así como tratar de realizar la misma acción desde otro rol no autorizado, obteniendo los correspondientes errores 500.

Se ha alcanzado un porcentaje de cobertura de 61.9%. Esto se debe a que nunca se prueba el método *unbind* del servicio relacionado al tratarse de un servicio de borrado.

**sponsor/invoice/list:** para realizar esta prueba, se ha accedido al listado de facturas de un rol *sponsor* comprobando que se lista adecuadamente, completando la prueba positiva de dicha funcionalidad.

Para la prueba de *hacking*, se han intentado listar las facturas de otro usuario *sponsor* así como listar facturas asociadas a un patrocinio inexistente, obteniendo el correspondiente error 500.

Se ha obtenido un porcentaje de cobertura de código de 93.9%. Solo han quedado warnings relacionados con comprobaciones de objetos nulos a través de *asserts*, los cuales se ignoran.

**sponsor/invoice/publish:** Para realizar esta prueba, se ha accedido a una factura de un *sponsor* en modo borrador y se ha seguido el mismo procedimiento que para la funcionalidad de crear, generando sus casos de prueba negativos. Para cumplir con los casos de prueba positivos, publicó una factura con datos de entrada válidos. Para la prueba de *hacking*, se ha intentado publicar una factura inexistente, que no estuviese en modo borrador o que perteneciera a otro usuario, además de comprobar que tampoco pueden publicarse desde otro rol, obteniendo el correspondiente error 500.

Se ha obtenido un porcentaje de cobertura de 95,9%. Solo han quedado warnings marcando comprobaciones de un objeto nulo a través de *asserts*, los cuales se ignoran.

**sponsor/invoice/show:** para realizar esta prueba, se ha accedido al listado de facturas de un patrocinio de un rol *sponsor* y se ha mostrado una de estas, comprobando que se muestra adecuadamente, completando la prueba positiva de la funcionalidad.

Para la prueba de *hacking*, se han intentado mostrar facturas de otro usuario, desde otro rol e incluso inexistentes, obteniendo el correspondiente error 500.

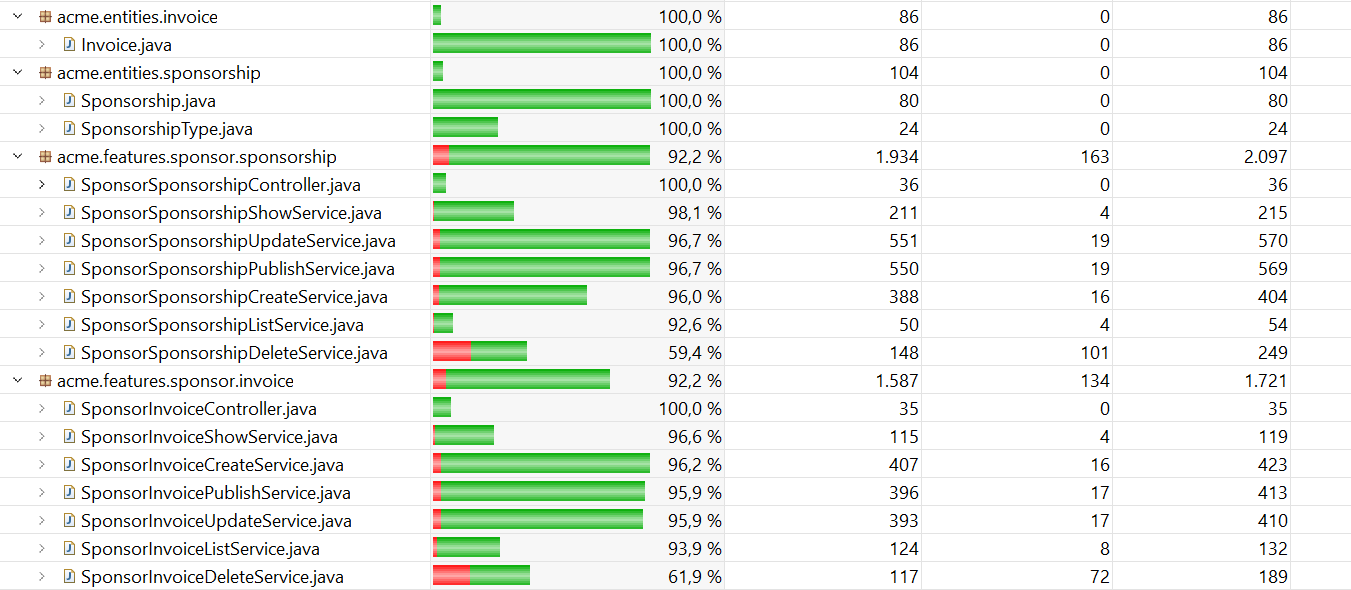
Se ha alcanzado un porcentaje de cobertura de código del 96,6%, quedando solo algunos warnings relacionados con comprobaciones de objetos nulos a través de *asserts*, los cuales se ignoran.

**sponsor/invoice/update:** para realizar las pruebas se siguió el mismo procedimiento de pruebas negativas que para la creación o publicación. Finalmente se ha enviado el formulario de creación con valores correctos, realizando la prueba positiva correspondiente.

Para la prueba de *hacking*se ha intentado actualizar una factura inexistente, que no estuviese en modo borrador, que perteneciera a otro *sponsor* y desde otro rol, obteniendo el correspondiente error 500. Se ha alcanzado una cobertura de código de 95.9%, quedando solo *warnings* relacionados con comprobaciones de objetos nulos a través de *asserts*, los cuales se ignoran.

Para comprobar los detalles sobre el tiempo de ejecución y los resultados de las pruebas realizadas, así como los datos originales de los cuales se ha extraído la información de este capítulo, puede consultar el documento adjunto llamado ***tester-performance-clean.xlsx*** enviado junto a la entrega ***D04: testing performance***.

Por otro lado, las pruebas realizadas en modo ***cobertura*** han mostrado un porcentaje de cobertura adecuado a las pruebas realizadas, el cual quedaría resumido en la siguiente captura.



Lo único que podría ayudar a mejorar el porcentaje sin añadir más código sería suprimir o reducir las líneas del método *unbind* en ambos servicios de borrado (borrar patrocinio y borrar factura), acción la cual no se ha llevado a cabo porque la diferencia de porcentaje de cobertura sería insignificante. También, podría haberse asegurado una mejor cobertura de código si para los métodos que recorrían a las clases hijas desde la clase padre en métodos como crear, actualizar o publicar, hubiesen contado con facturas en modo borrador en los datos de ejemplo. No obstante, para otras pruebas estos datos fueron creados durante la grabación para asegurar su cobertura.

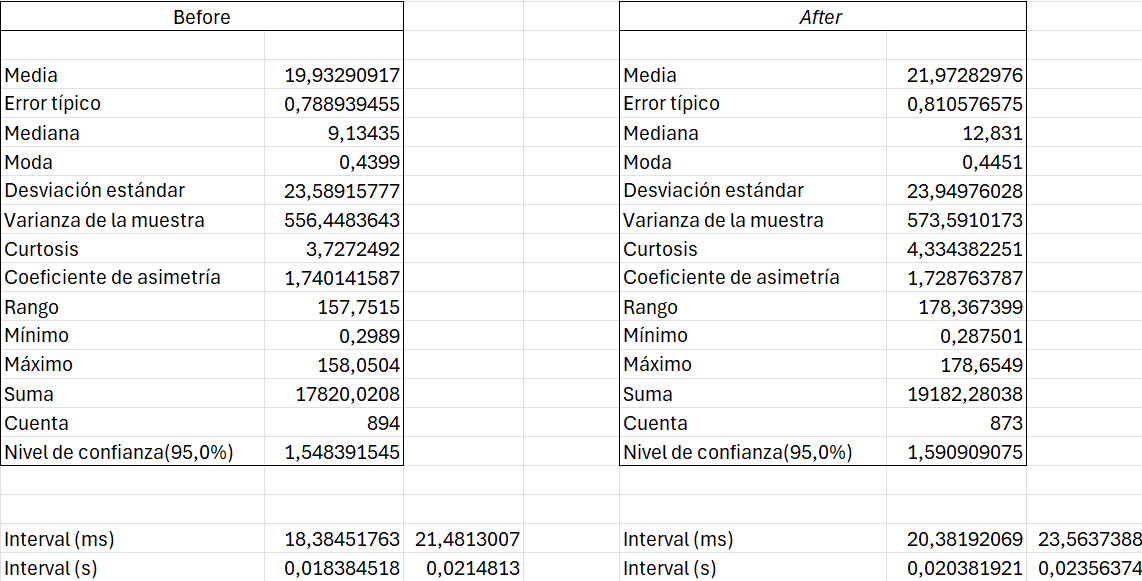
*Nota: para la relación patrocinio - proyecto, se intentó comprobar si desde la vista del sistema se podía asignar a un patrocinio un proyecto que no estuviese publicado o que no existiese. Como no se consiguió de ninguna forma debido a la implementación del* SelectChoices *que muestra los proyectos publicados para escoger, se tomó la decisión de no incluir esta validación en las pruebas para evitar reducir la cobertura de código con validaciones que nunca llegarían a cumplirse. Para la relación facturas - patrocinio, se pensó algo similar. Para evitar romper la relación una vez creada, el patrocinio asociado no se envía en ningún método bind*.

# Capítulo II: Testing de rendimiento

En este capítulo se proporcionarán gráficos adecuados y un intervalo de confianza del 95 % para el tiempo que tarda el proyecto en atender las solicitudes de las pruebas funcionales, además de un contraste de hipótesis de confianza del 95 % con respecto a los tests antes y después de añadir índices.

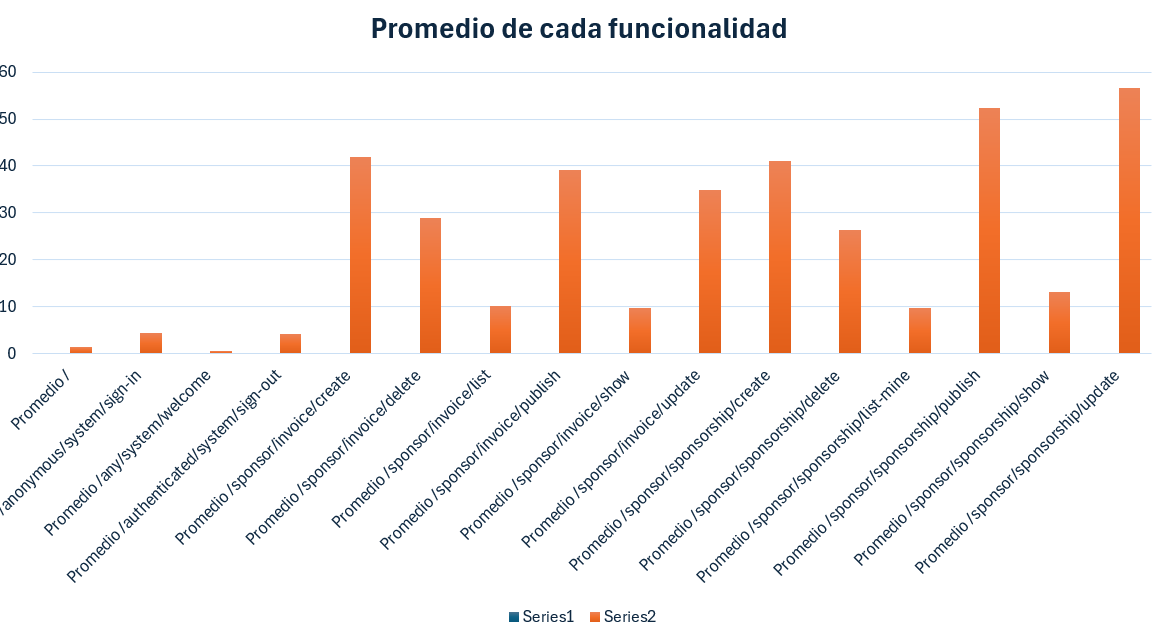
**Intervalo de confianza**

En este apartado se han recolectado 895 y 875 datos de los distintos tests y se ha calculado el intervalo de confianza con un nivel del 95% en la misma máquina, primero sin índices en las tablas de la base de datos y luego con ellos.



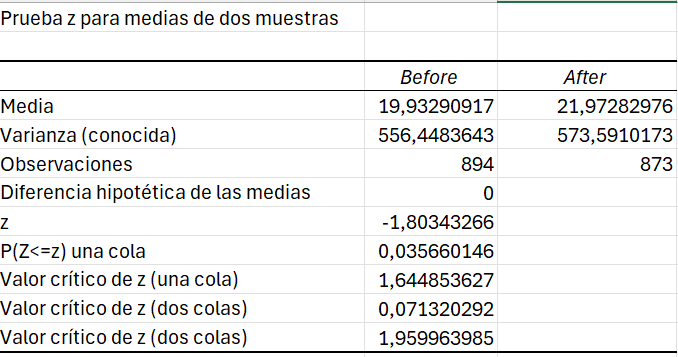
Como se muestra en los intervalos en segundos al final de cada tabla, se ha producido un aumento en los tiempos, a pesar de que en la segunda tabla se presentan los resultados de ejecutar los tests usando índices, lo cual, en teoría, debería mejorar el rendimiento.

Los tests ejecutados antes de añadir los índices mostraron que la función de actualizar un patrocinio (MIR) tuvo un impacto significativo, como se puede ver en el siguiente gráfico del tiempo (en ms) de cada funcionalidad probada.



**Contraste de hipótesis**

Para comparar adecuadamente los intervalos de confianza calculados en ambas pruebas tras ejecutar los tests, se ha realizado la prueba Z-Test sobre las columnas de tiempos generados por ambas pruebas (antes y después de añadir índices), obteniendo un valor crítico de z (dos colas) = 0,0713202916174789.



Este valor crítico de z indica que nos encontramos en el intervalo (a - 1.00], donde a = alpha = 1 - Intervalo de Confianza = 1 - 0.95 = 0.05. Al ser z = 0,0713202916174789, tenemos que: a < z <= 1.00.

Esta información revela que las pruebas ejecutadas tras añadir índices no resultaron en una mejora significativa respecto a las pruebas anteriores. Los tiempos de muestreo son diferentes, pero en conjunto son los mismos. Por lo tanto, a pesar de añadir índices a las entidades durante la fase de testing, los tests terminaron con los mismos resultados. De esta manera, podemos garantizar que el proceso de testing realizado refleja los resultados obtenidos, ya que la segunda prueba con índices confirma los mismos resultados, aunque con tiempos diferentes.

Los cálculos detallados del análisis estadístico de este capítulo pueden consultarse en los documentos adjuntos a esta entrega:

* tester-performance-clean.xlsx
* tester-performance2.xlsx
* ztest.xlsx

# Bibliografía

-Documento 08 Annexes, de los contenidos de la plataforma virtual de esta asignatura.

-S02 Performance testing, transparencias del módulo de testing L04 de la asignatura