Informe de Testing Grupal



Integrantes del grupo C2.026:

* Ignacio Blanquero Blanco ([ignblabla@alum.us.es](mailto:ignblabla@alum.us.es))
* María de la Salud Carrera Talaverón ([marcartal1@alum.us.es](mailto:marcartal1@alum.us.es))
* Joaquín González Ganfornina ([joagongan@alum.us.es](mailto:joagongan@alum.us.es))

Fecha: Sevilla, 8 de Julio 2024

Tabla de contenido

[Resumen del Informe 3](#_Toc170389305)

[Historial de Versiones 4](#_Toc170389306)

[Introducción 5](#_Toc170389307)

[Testing Funcional 6](#_Toc170389308)

[Create 6](#_Toc170389309)

[Delete 6](#_Toc170389310)

[List 7](#_Toc170389311)

[Show 7](#_Toc170389312)

[Update 7](#_Toc170389313)

[Testing de Rendimiento 8](#_Toc170389314)

[Primer análisis 8](#_Toc170389315)

[Segundo análisis 10](#_Toc170389316)

[Contraste 12](#_Toc170389317)

[Conclusiones 13](#_Toc170389318)

[Bibliografía 14](#_Toc170389319)

# Resumen del Informe

En este informe se detalla la metodología empleada para llevar a cabo la realización de pruebas para la entidad Banner. Cada miembro del grupo participó activamente para cumplir con el objetivo de la tarea sin mayores complicaciones. Se describe brevemente cada una de las pruebas, detallando cómo han ayudado a detectar problemas en el código, si procede.

Además, se realiza un análisis estadístico utilizando las herramientas aprendidas en la asignatura, evaluando el rendimiento de las pruebas. Este análisis permitirá valorar la calidad del trabajo y comparar los resultados de las pruebas antes y después de implementar algunas mejoras.

# Historial de Versiones

|  |  |  |  |
| --- | --- | --- | --- |
| **Versión** | **Contenidos** | **Fecha** | **Contribuyente** |
| V0.1 | Estructura principal. | 26/05/2024 | Natalia Olmo Villegas |
| V1.1 | Análisis cobertura testing funcional. | 27/05/2024 | Ignacio Blanquero Blanco, María de la Salud Carrera Talaverón, Natalia Olmo Villegas, Joaquín González Ganfornina, Adrian Cabello Martín |
| V2.0 | Modificaciones para la segunda convocatoria. Descripción de pruebas realizadas. | 27/06/2024 | Mª Salud Carrera Talaverón |
| V2.1 | Modificaciones para la segunda convocatoria. Comparación de los testing de rendimiento y conclusión. | 06/07/2024 | Joaquín González Ganfornina |

# Introducción

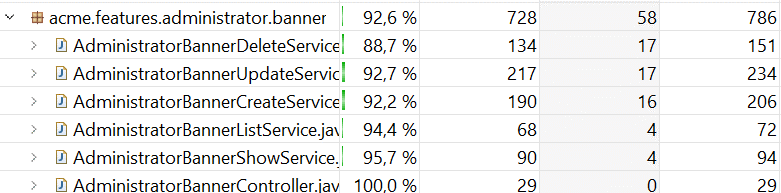
Este informe se divide en dos capítulos: testing funcional y testing de rendimiento o performance. En el capítulo de testing funcional se describen las pruebas realizadas sobre la entidad grupal Banner. Los tests están agrupados por funcionalidad y naturaleza del mismo: test seguro (casos positivos y negativos) o test de hacking. Para cada uno de ellos se explica en qué consiste el test y, si procede, los bugs de código que ha ayudado a encontrar.

Por otro lado, en el capítulo de testing de performance se analiza el rendimiento de la aplicación. Para ello se muestra en una gráfica el tiempo medio que tarda en realizarse cada petición durante la ejecución de los tests y se proporcionan estadísticas sobre estos tiempos. A continuación, se modifica el código para añadir los índices necesarios en Banner, siguiendo las consultas de sus dos repositorios, con el objetivo de mejorar el rendimiento. Se proporciona un análisis sobre esta segunda ejecución y se contrastan ambos resultados.

Finalmente, el informe incluye una conclusión sobre los resultados obtenidos y las medidas tomadas respecto a ellos.

# Testing Funcional

En el proceso de pruebas de la entidad *“Banner.java”*, se ha logrado alcanzar un 92.6% de cobertura. Este indicador es muy positivo, pues destaca la exhaustividad de las pruebas realizadas. Dicha cobertura se traduce en que, de un total de 786 instrucciones, 728 han sido cubiertas y 58 no lo han sido.

Aunque una cobertura del 92.6% es un excelente indicador, es importante recordar que el objetivo principal no es alcanzar el 100% de cobertura, sino asegurar que las áreas más críticas y susceptibles a errores del código estén adecuadamente cubiertas. Este objetivo ha sido cumplido con éxito, ya que ninguna de las áreas críticas corresponde a las 58 líneas que no fueron cubiertas.

Cabe destacar que todos los tests incluyen líneas de código resaltadas en amarillo correspondientes a la instrucción *“assert object != null”*. Esta situación es completamente normal y esperada, ya que la cobertura de esta línea indicaría que el sistema ha encontrado un fallo irreversible durante los casos de prueba. Estas líneas están presentes como medida preventiva; sin embargo, es afortunado que no hayan sido ejecutadas durante las pruebas.

Es importante señalar también que, en este proyecto, sólo se han considerado posibles intentos de hacking que implican la inserción de URLs en la barra de búsqueda.

A continuación, se describen las pruebas grabadas para la entidad Banner. Todas corresponden a las funcionalidades que pueden realizar los usuarios administradores sobre los banner.s

### Create

* **create.safe:** varios administradores crean banners nuevos. Para cada atributo, se han probado todos los casos posibles y no posibles, con el fin de comprobar que salta el error adecuado. Se han tenido en cuenta las restricciones personalizadas, como que el período de visualización deba ser de al menos una semana de duración o que la fecha de inicio debe comenzar después de la fecha de instanciación, entre otras. El único caso que no se ha probado ha sido el de poner una URL de más de 255 caracteres en los campos “picture” o “targetWebDocument”, pues esto provoca un error 500.
* **create-right-role-right-user-wrong-action.hack:** un administrador intenta crear un proyecto introduciendo datos de Script y SQL injection en el campo “slogan”, sin conseguir realizar el hacking. Además, introduce un enlace de más de 255 caracteres en los campos correspondientes y recibe un error 500 como respuesta.
* **create-wrong-role.hack:** un usuario anónimo introduce la URL de creación de un proyecto y recibe un error 500 como respuesta.

### Delete

* **delete.safe:** varios administradores borran los banners uno a uno hasta que no queda ninguno. Como esta funcionalidad no tiene ningún requisito más allá de que el usuario sea un administrador, no tiene mayor complejidad.

### List

* **list.safe:** todos los administradores listan los proyectos. Utilizan los índices de navegación para ver distintas páginas de la lista y la barra de búsqueda para filtrar las entradas.
* **list-wrong-role.hack:** un usuario anónimo introduce la URL de listado de banners y recibe un error 500 como respuesta ya que no tiene el rol “administrator” que necesita para poder acceder a la URL.

### Show

* **show.safe:** varios administradores muestran varios banners.
* **show-right-role-right-user-wrong-action.hack:** un administrador intenta mostrar un banner inexistente.
* **show-wrong-role.hack:** un usuario anónimo intenta mostrar un banner como si fuera un administrador, para lo que no está autorizado.

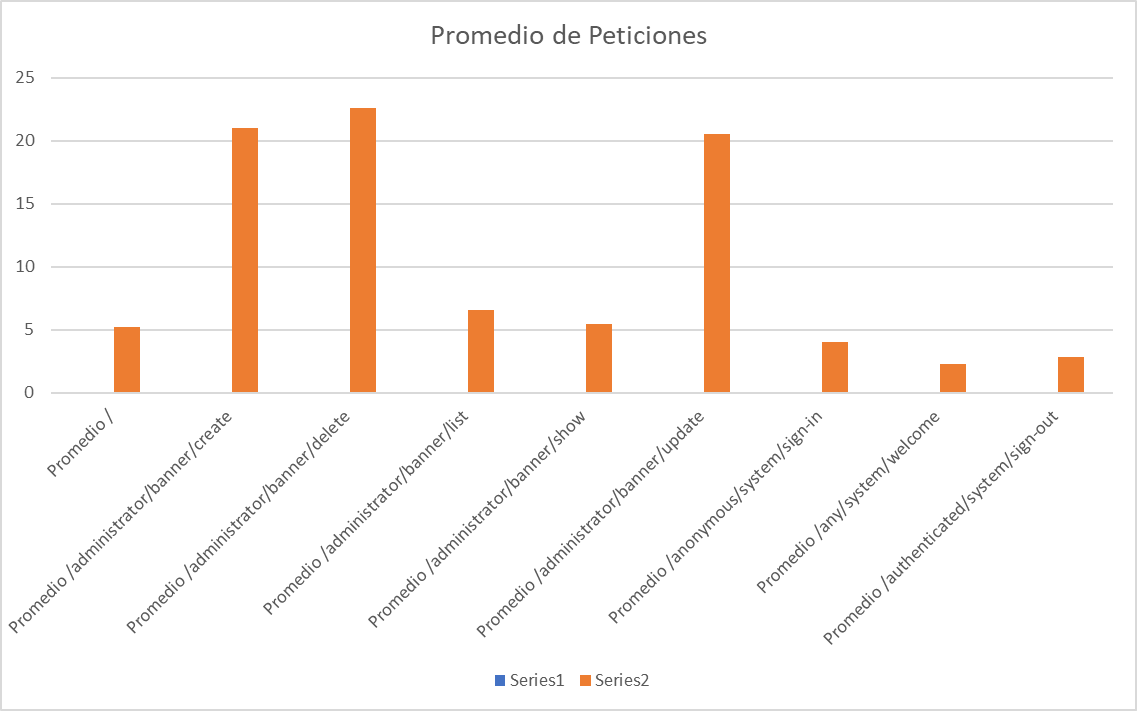
### Update

* **update.safe:** varios administradores actualizan algunos banners existentes. Para cada atributo, se han probado todos los casos posibles y no posibles, con el fin de comprobar que salta el error adecuado. Se han tenido en cuenta las restricciones personalizadas, como que el período de visualización deba ser de al menos una semana de duración o que la fecha de inicio debe comenzar después de la fecha de instanciación, entre otras. El único caso que no se ha probado ha sido el de poner una URL de más de 255 caracteres en los campos “picture” o “targetWebDocument”, pues esto provoca un error 500.
* **update-right-role-right-user-wrong-action.hack:** un administrador intenta actualizar un banner introduciendo datos de Script y SQL injection en el campo “slogan”, sin conseguir realizar el hacking. Además, introduce un enlace de más de 255 caracteres en los campos correspondientes y recibe un error 500 como respuesta.

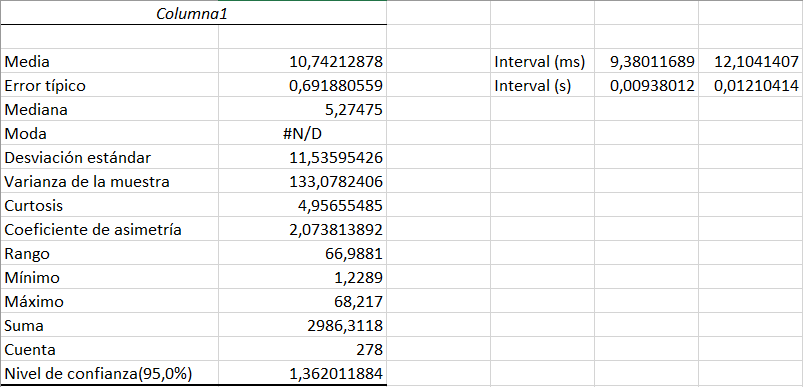
# Testing de Rendimiento

El testing del rendimiento de la aplicación se ha realizado analizando los tiempos de ejecución de las pruebas grabadas descritas en el capítulo anterior. Para ello, se ha realizado una primera ejecución sobre el código implementado en el tercer entregable (con los arreglos pertinentes tras descubrir bugs y errores y tras corregir el proyecto para la segunda convocatoria). Posteriormente, se ha modificado la entidad Banner para añadirle índices, lo que debería mejorar el rendimiento, y se ha realizado un segundo análisis. Los archivos con los resultados de los análisis pueden consultarse en la carpeta /reports/Group/tester-performance del proyecto.

## Primer análisis



En este gráfico se observa que las peticiones más costosas corresponden a las de creación, borrado y actualización de banners. Esto puede deberse a que son los servicios más costosos, por lo que sus peticiones tardan más tiempo en realizarse.

Esta imagen muestra una serie de variables estadísticas sobre los tiempos de todas las peticiones realizadas. Vemos que el intervalo de confianza es, aproximadamente, de [9.38, 12.104) milisegundos.

## Segundo análisis

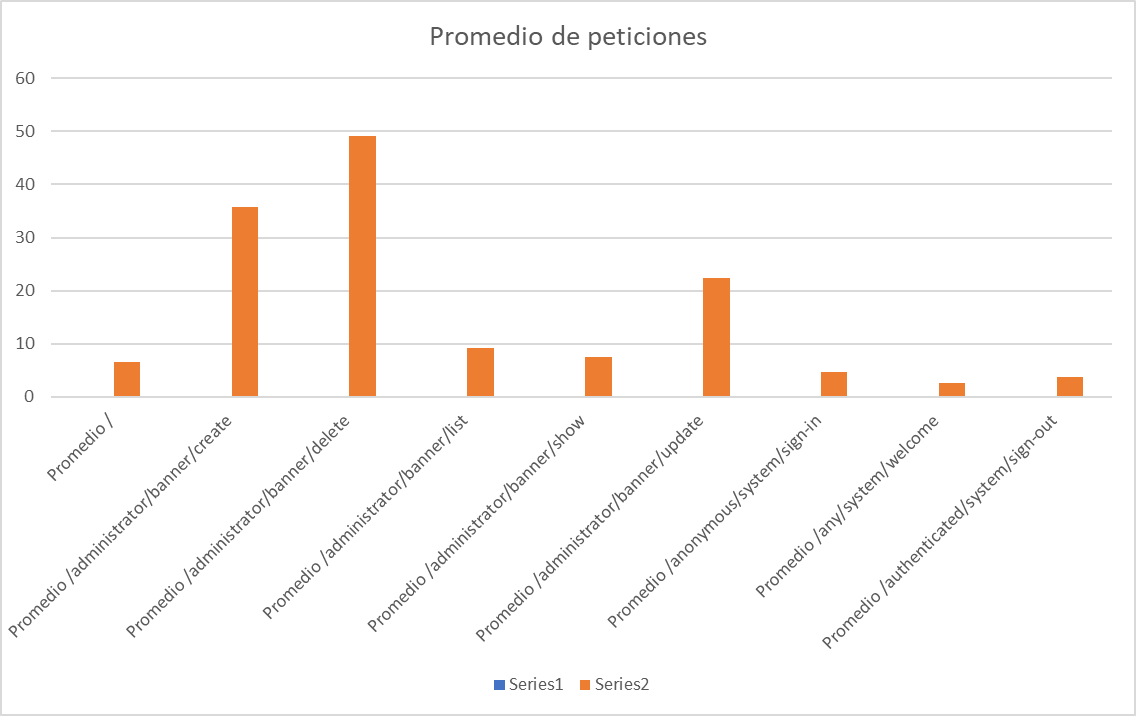
Antes de realizar el segundo análisis, se han añadido los siguientes índices en la clase Banner.java, tomando de referencia las consultas hechas en los repositorios AdministratorBannerRepository.java y BannerRepository.java:

@Table(indexes = {

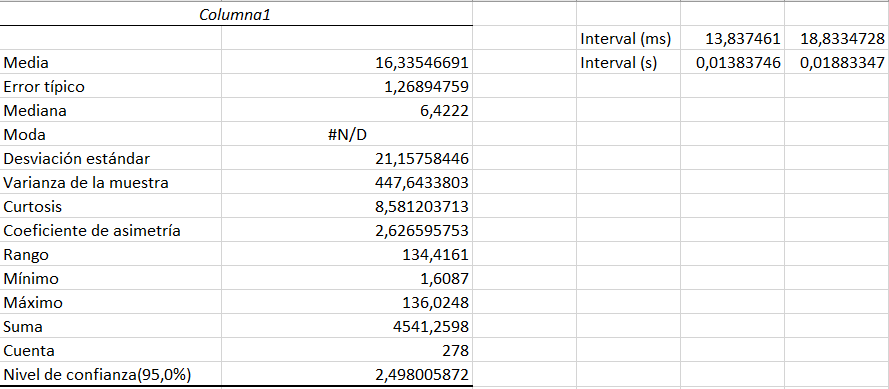
@Index(columnList = "finishDate,startDate")

})

Ahora, los resultados obtenidos han sido los siguientes:



Vemos que, las peticiones de borrado son las más costosas, seguido por las peticiones de creación y de actualización. Además, el resultado es considerablemente peor, ya que el tiempo de estas peticiones es mayor con respecto a la gráfica anterior.

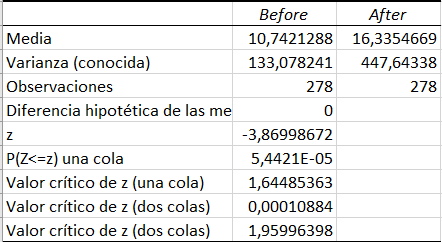


En las medidas estadísticas, observamos que ahora el intervalo de confianza está aproximadamente en el rango [13.83, 18.83) milisegundos.

## Contraste



En esta imagen, se pueden ver una frente a otra las estadísticas de ambas ejecuciones. Para comprobar si el rendimiento ha mejorado o empeorado, debemos realizar la “Prueba z para medias de dos muestras”:



La prueba se ha realizado con un valor de alfa igual a 0.05. La variable que nos interesa analizar es el valor crítico de z (dos colas), el primero de los dos que aparecen en la imagen, puesto que es el que va a determinar si nuestro análisis es concluyente o no. Vemos que dicho valor crítico se encuentra en el intervalo [0.00, alfa); por tanto, se pueden comparar las medias. Concluimos que la introducción de índices ha hecho que el rendimiento sea peor, porque el valor de la media del segundo análisis es mayor, por lo que se ha optado por eliminarlos.

Debido a que este es un buen contraste entre ambos análisis, como ya se ha explicado (valor crítico de z), se ha decidido que no es necesario realizar una tercera ejecución en otro equipo. Los resultados obtenidos son concluyentes.

# Conclusiones

Después de analizar los tests realizados, se destaca que la cobertura de las entidades grupales ha sido bastante alta y aceptable. Además, se ha establecido un índice del 95% para el conjunto de casos de prueba sin los índices recomendados, así como para el conjunto que se han utilizado los índices recomendados.

Esto se realizó utilizando la herramienta Excel y siguiendo las indicaciones dadas en las trasparencias de la asignatura, con el fin de hacer un análisis estadístico que nos permitiera decidir cuál de las dos opciones ha tenido un mejor resultado.

Tras analizar los resultados del conjunto de casos con los índices recomendados y el conjunto sin índices, se ha comprobado que el conjunto sin índices tenía un rendimiento mayor, por lo que se ha optado por eliminarlos.

# Bibliografía

Transparencias de la asignatura DP2 del curso 2023/2024:

* D04: S01 – Formal testing
* D04: S02 – Performance testing