Universidad de Sevilla  
Escuela Técnica Superior de Ingeniería Informática

**Testing Report**



Grado en Ingeniería Informática – Ingeniería del Software  
Diseño y Pruebas 2.

Curso 2023 – 2024

| **Fecha** | **Versión** |
| --- | --- |
| 23/05/2024 | v1.0.0 |

| **Grupo: C1.011** | | |
| --- | --- | --- |
| **Autores por orden alfabético** | **Rol** | **Descripción del rol** |
| Abouri, Mohamed - Y7156458E | Developer | Persona encargada de desarrollar el código. |
| Cabello Ranea, Rafael -32094496C | Operador | Encargado de las tareas de campo, de las instalaciones y del mantenimiento de los sistemas de la empresa. |
| Calderón Rodríguez, Francisco Javier - 78233219F | Project administrador | Persona encargada de tomar decisiones de diseño y vigilar el correcto desarrollo |
| Delgado Pallares, David - 29519510E | Tester | Persona encargada de realizar pruebas sobre el código. |
| Ramírez Lara, Iván- 77852382G | Developer | Persona encargada de desarrollar el código. |

**Control de Versiones**

| **Fecha** | **Versión** | **Descripción** |
| --- | --- | --- |
| 23/05/2024 | v1.0.0 | Desarrollo de la primera versión. |
|  |  |  |
|  |  |  |
|  |  |  |

**Índice de contenido**

[1. Introducción 2](#_heading=h.gjdgxs)

2[.](#_heading=h.gjdgxs) Testeo funcional 2

3[. Testeo de re](#_heading=h.30j0zll)ndimiento 10

4. Cobertura 15

1. **Introducción**

Este documento presenta dos capítulos esenciales: pruebas funcionales, con un listado de casos de prueba organizados por característica y su efectividad en detectar errores, y pruebas de rendimiento, con gráficos e intervalos de confianza del 95% para el tiempo de respuesta en dos computadoras, además de un análisis para determinar cuál es más potente. Estos capítulos son cruciales para evaluar la calidad y el rendimiento del software.

1. **Testeo funcional**

En esta sección se detallarán todas las pruebas que hemos realizado para testear el funcionamiento de nuestro software. Se explicará qué es lo que se ha hecho, el resultado esperado, el resultado real (que hace referencia al resultado del test corregido después de encontrar bugs que hicieran que el código estuviera mal) y, por último, la columna de bugs encontrados. Si ha habido bugs, se ha arreglado el test, por lo que el resultado real ya no los contiene.

Descripción de la prueba: Se detallará cada prueba realizada, incluyendo los pasos específicos llevados a cabo.

Resultado esperado: Se especificará cuál era el comportamiento o el resultado que se esperaba obtener con la prueba.

Resultado real: Se presentará el resultado obtenido tras la ejecución del test, ya corregido después de encontrar y solucionar cualquier bug que afectará el código.

Bugs encontrados: Se listarán los bugs detectados durante la prueba y las correcciones realizadas. Una vez solucionados los bugs, se actualizará el resultado real para reflejar el funcionamiento correcto del software.

**List-Show**

| **Entidad** | **Descripción** | **Resultado Esperado** | **Resultado Real** | **Bugs detectados** |
| --- | --- | --- | --- | --- |
| **banner safe** | Verificar que un administrador puede listar los banners y acceder a la información de estos. | El sistema deberá mostrar los banners de la aplicación y la información de estos. | El sistema ha mostrado los banners y la información de estos. | Ninguno |
| **banner hack** | Verificar que un rol no administrador no puede listar, banner ni acceder a su información. | El sistema no debe mostrar los banners, usando un rol no admin, dando un error de pánico | El sistema ha dado un error 500 de no autorizado en cada rol no admin. | Ninguno |

**Create**

| **Entidad** | **Descripción** | **Resultado Esperado** | **Resultado Real** | **Bugs detectados** |
| --- | --- | --- | --- | --- |
| **banner safe** | Verificar que un administrador puede crear banners de forma correcta. | El sistema deberá permitir crear banners válidos. | El sistema ha dejado crear banners con datos válidos. | Ninguno |
| **banner hack** | Verificar que un rol no administrador no puede acceder a la creación de banners. | El sistema debe de lanzar error de pánico si un rol no administrador intenta acceder a la URL o si se hace una petición sin ser administrador. | El sistema ha dado un error 500 de no autorizado en cada rol no administrador. | Ninguno |

**Delete**

| **Entidad** | **Descripción** | **Resultado Esperado** | **Resultado Real** | **Bugs detectados** |
| --- | --- | --- | --- | --- |
| **banner safe** | Verificar que un administrador puede borrar banners de forma correcta. | El sistema deberá permitir eliminar banners que no estén publicados. | El sistema ha dejado eliminar banners no publicados. | Se ha detectado un **error 500** al borrar los banners. El error es causado por una relación con una entidad implementada por el estudiante 5. |
| **banner hack** | Verificar que un rol no administrador no puede acceder a la eliminación de banners. | El sistema debe de lanzar error de pánico si un rol no administrador intenta acceder a la URL o si se hace una petición sin ser administrador. | El sistema ha dado un error 500 de no autorizado en cada rol no administrador y en administrador no autorizados. | Ninguno |

**update**

| **Entidad** | **Descripción** | **Resultado Esperado** | **Resultado Real** | **Bugs detectados** |
| --- | --- | --- | --- | --- |
| **banner safe** | Verificar que un administrador puede actualizar banners válidos de forma correcta. | El sistema deberá permitir actualizar banners válidos. | El sistema ha dejado actualizar banners con datos válidos. | Ninguno |
| **banner hack** | Verificar que un rol no administrador u otro administrador no puede acceder a la actualización de banners. | El sistema debe de lanzar error de pánico si un rol no administrador intenta acceder a la URL, si se hace una petición sin ser administrador o si se hace desde un administrador diferente | El sistema ha dado un error 500 de no autorizado en cada rol no administrador. También ha dado error 500 si un administrador intentaba acceder a un banner de otro administrador | Ninguno |

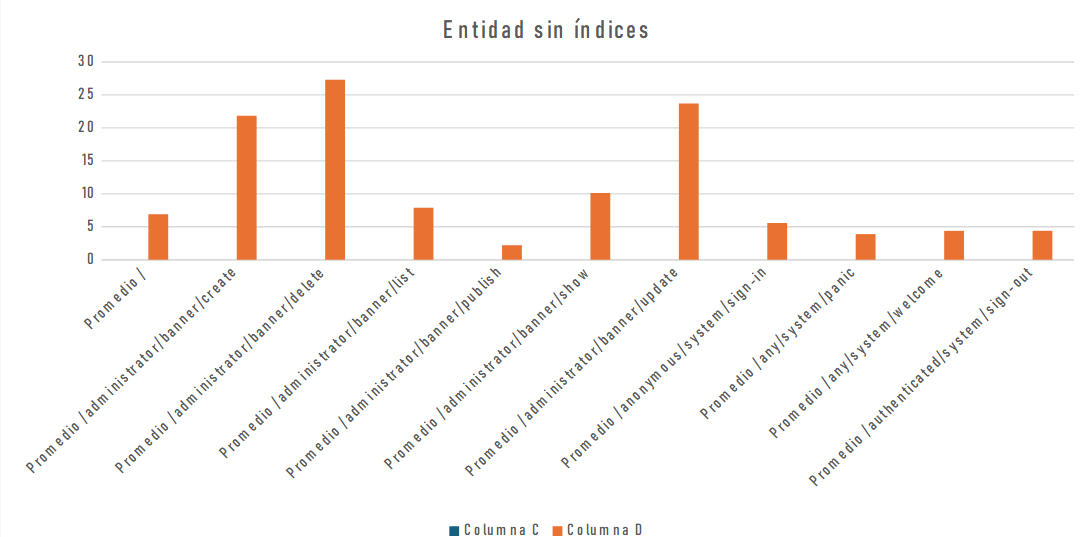
1. **Testeo de rendimiento**

En este capítulo, se evaluará el rendimiento de nuestro proyecto mediante pruebas detalladas que analizan el tiempo de respuesta al atender las solicitudes en nuestras pruebas funcionales. Como en esta entidad no se puede añadir índices debido a que todas las queries usan atributos de navegación, no podemos comparar los resultados obtenidos por hacer realplayer sobre las entidades con índices y sin estos.

Como no podemos realizar el análisis de estas pruebas, solo presentaremos gráficos que ilustran los resultados obtenidos. Además, no calcularemos los intervalos de confianza del 95% para los tiempos de respuesta medidos y no realizaremos un contraste de hipótesis, en su lugar mostraremos los tiempos que tarda en realizarse las pruebas sin índices.

**Muestra sobre resultados desde Coverage.**

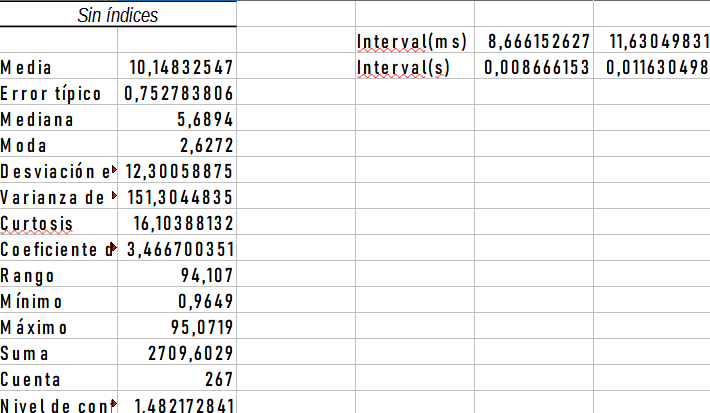
**Gráficas de entidades sin índices:**

****

**Tiempo de hacer los test:**

****

**Análisis de los tiempos:**

****

**Conclusión**

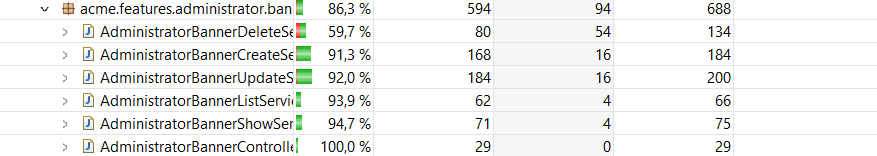
Al analizar los resultados, observamos una gran diferencia en los tiempos de ejecución de los métodos delete, update y create en comparación con los demás métodos. Esta discrepancia se debe al gran volumen de datos probado en estas operaciones. Además, el tiempo adicional necesario se debe al proceso de validación de datos durante las solicitudes POST. Antes de insertar los datos en la base de datos, es necesario verificar que todos los datos sean válidos, lo cual no ocurre con otros métodos.

**Cobertura**

En esta sección, analizaremos la cobertura de código lograda con nuestros tests. La cobertura de código indica qué porcentaje del código fuente ha sido ejecutado durante las pruebas, ayudando a identificar áreas verificadas y posibles errores no detectados.

Discutiremos los métodos de medición, presentaremos los resultados obtenidos y evaluaremos la efectividad de nuestros tests. También identificaremos lagunas en la cobertura y propondremos estrategias para mejorarla, asegurando así mayor confiabilidad y robustez del software.

**Cobertura Banner.**



Como se puede apreciar, la cobertura sobre las features de la entidad Banner es bastante buena, ya que todas las features implementadas tienen una cobertura de más del 90% excepto el delete.

Esto ocurre porque el delete cuenta con el método unbind, pero este no es nunca usado porque no es necesario para esta funcionalidad, si comentamos este código o lo borramos la cobertura ya superaria el 90% de porcentaje, es por esto que no estamos preocupados por que este no llegue al 90% de cobertura.