Universidad de Sevilla  
Escuela Técnica Superior de Ingeniería Informática



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

Curso 2024 – 2025

**Informe de testing**

**Grupo:** C1.030

**Número de estudiante dentro del grupo:** 2

**Repositorio**: <https://github.com/JuanAntonioMorenoMoguel/Acme-ANS-D04>

**Fecha**: 26/05/2025

|  |  |
| --- | --- |
| **Alumno** | **Correo** |
| García Abadía, Enrique | enrgaraba@alum.us.es |

**Historial de versiones**

|  |  |  |
| --- | --- | --- |
| **Versión** | **Fecha** | **Descripción de los cambios** |
| V1.0 | 24/05/2025 | Creación del informe |
| V1.1 | 25/05/2025 | Elaboración y revisión del informe |
|  |  |  |

**Índice de contenido**

[**1.** **Resumen ejecutivo** 2](#_Toc167735368)

[**2.** **Introducción** 2](#_Toc167735369)

[**3.** **Contenido** 3](#_Toc167735370)

[**3.1.** **Testing funcional** 3](#_Toc167735371)

[**3.1.1.** **Funcionalidad para los managers sobre los proyectos** 3](#_Toc167735372)

[**3.1.1.1.** **List-Mine** 3](#_Toc167735373)

[**3.1.1.2.** **Show** 4](#_Toc167735374)

[**3.1.1.3.** **Create** 4](#_Toc167735375)

[**3.1.1.4.** **Update** 5](#_Toc167735376)

[**3.1.1.5.** **Delete** 5](#_Toc167735377)

[**3.1.1.6.** **Publish** 6](#_Toc167735378)

[**3.1.2.** **Funcionalidad para los managers sobre las historias de usuario** 6](#_Toc167735379)

[**3.1.2.1.** **List-Mine** 6](#_Toc167735380)

[**3.1.2.2.** **List** 7](#_Toc167735381)

[**3.1.2.3.** **Show** 7](#_Toc167735382)

[**3.1.2.4.** **Create** 8](#_Toc167735383)

[**3.1.2.5.** **Update** 8](#_Toc167735384)

[**3.1.2.6.** **Delete** 9](#_Toc167735385)

[**3.1.2.7.** **Publish** 9](#_Toc167735386)

[**3.1.3.** **Funcionalidad para los managers sobre la relación entre proyectos e historias de usuario** 10](#_Toc167735387)

[**3.1.3.1.** **Create** 10](#_Toc167735388)

[**3.1.3.2.** **Delete** 11](#_Toc167735389)

[**3.2.** **Testing de rendimiento** 12](#_Toc167735390)

[**3.2.1.** **Sin índices – Ordenador estudiante 1** 12](#_Toc167735391)

[**3.2.2.** **Con índices – Ordenador estudiante 1** 13](#_Toc167735392)

[**3.2.3.** **Con índices – Ordenador estudiante 4** 15](#_Toc167735393)

[**4.** **Conclusiones** 17](#_Toc167735394)

[**Bibliografía** 18](#_Toc167735395)

# **Resumen ejecutivo**

Este informe presenta los resultados detallados de las pruebas funcionales y de rendimiento realizadas en el proyecto. En el apartado de pruebas funcionales, se describen los casos de prueba implementados. Cada caso de prueba incluye una descripción clara y una evaluación de su efectividad en la detección de errores. En el capítulo de pruebas de rendimiento, se presentan gráficos detallados y un intervalo de confianza del 95% para el tiempo de respuesta del sistema al procesar solicitudes con la aplicación de índices y sin ellos. Además, se incluye un análisis comparativo con un nivel de confianza del 95% para determinar estadísticamente si los índices tienen impacto en el rendimiento de las pruebas. Este informe utiliza un enfoque sistemático y riguroso, asegurando que los resultados sean precisos y útiles para la evaluación del proyecto.

# **Introducción**

El presente documento tiene como objetivo detallar los procedimientos y resultados de las pruebas funcionales y de rendimiento llevadas a cabo en el proyecto. En primer lugar, se aborda el apartado de pruebas funcionales, donde se describen los casos de prueba diseñados para verificar las diversas características del sistema. Cada uno de los casos de prueba incluye una descripción detallada y una evaluación de su efectividad en la detección de errores. Esta sección proporciona una visión clara y estructurada del desempeño del sistema bajo diferentes condiciones de prueba, permitiendo identificar y corregir fallos de manera efectiva.

En segundo lugar, el capítulo de pruebas de rendimiento se centra en el análisis del tiempo de respuesta del sistema al procesar solicitudes durante las pruebas funcionales. Se incluyen gráficos detallados que ilustran los tiempos de respuesta con la implementación de índices y sin ellos, junto con un intervalo de confianza del 95% para estos tiempos. Además, se lleva a cabo un contraste de hipótesis con un nivel de confianza del 95% para determinar si la implementación de índices tiene impacto en el rendimiento. El enfoque utilizado en este informe es metódico y sistemático, utilizando herramientas de planificación y Eclipse para garantizar la precisión y efectividad en la detección de errores durante las pruebas funcionales.

# **Contenido**

# **Testing funcional**

En esta sección vamos a explicar los casos de pruebas (positivos, negativos y de hacking) que hemos implementado para cada funcionalidad del sistema. Además, vamos a ofrecer una explicación de como de efectivo es para detectar errores, ayudados por la cobertura que nos aporta la herramienta Eclipse la cual nos indica el porcentaje de instrucciones que ha sido ejecutadas.

# **Funcionalidad para los customer sobre los booking**

Vamos a comenzar por las pruebas establecidas para las funcionalidades de los customer sobre los bookings.

# **List**

Esta funcionalidad consiste en que un customer pueda ver la lista de sus bookings.

Para realizar las pruebas positivas y negativas de esta funcionalidad nos hemos logueado como un customer y hemos listado sus bookings.

Para realizar las pruebas de hacking, con el rol anónimo hemos introducido al url de mostrar nuestros booking (…/customer/booking/list) comprobando que nos saltaba un error y los mismo, pero ahora con otro rol (manager en mi caso).

Tras realizar estas pruebas hemos obtenido en el servicio un 100% de cobertura lo cual nos asegura que la funcionalidad no esconde ningún posible fallo ya que ha ejecutado todas las instrucciones.

# **Show**

Esta funcionalidad consiste en que un customer pueda ver las características/atributos de uno de sus bookings.

Para realizar las pruebas positivas y negativas de esta funcionalidad nos hemos logueado como customer, hemos mostrado el listado de sus bookings y hemos ido pulsando en algún booking comprobando que aparecía la información correcta.

Para realizar las pruebas de hacking, hemos intentado acceder a la información de un proyecto con un rol distinto al del customer (anónimo y manager) y luego hemos intentado acceder como un customer pero con el usuario incorrecto, es decir, al booking de un customer distinto al que nos hemos logueado. En todos los casos han saltado los errores correspondientes.

Tras realizar estas pruebas hemos obtenido un 100% de cobertura asegurándonos de esta manera que no van a existir errores ya que se han ejecutado todas las líneas de código.

# **Create**

Esta funcionalidad consiste en que un customer puede crear un booking estableciendo las características de su booking.

Para realizar las pruebas positivas y negativas, hemos empezado primero por las negativas, mandando el formulario en blanco y luego más tarde tras rellenar todos los campos con valores que pasan los test hemos ido probando todos los fallos distintos posibles que podían darse para cada atributo. Tras probar todos los casos negativos posibles, nos hemos limitado a crear bookings probando con los rangos de valores posible como se ha explicado en la metodología de clase (probando los extremos de las restricciones, los diferentes valores que puede tomar…), siempre siendo valores permitidos.

Para realizar las pruebas de hacking, hemos probado a introducir la url para crear proyecto con un rol distinto al de customer (anónimo y manager), comprobando que no dejaba acceder.

Tras realizar estas pruebas hemos obtenido un 100% de cobertura asegurándonos de esta manera que no van a existir errores ya que se han ejecutado todas las líneas de código.

.

# **Update**

Esta funcionalidad consiste en que un customer pueda actualizar uno de sus bookings siempre y cuando este booking no haya sido publicado.

Para realizar las pruebas positivas y negativas, hemos llevado una metodología muy parecida a la llevada en el create solo que en el caso de las positivas en vez de ir creándolos hemos ido actualizando el booking.

Para realizar las pruebas de hacking, hemos intentado actualizar un proyecto con un rol diferente al de customer (manager y anónimo), después nos hemos logueado con un customer pero con un usuario distinto al del booking que hemos intentado actualizar (un booking publicado y otro no publicado). Por último, nos hemos logueado con un customer y con el usuario adecuado pero hemos intentado actualizar un proyecto que ya está publicado. En todos los casos el sistema ha funcionado como esperábamos, rechazando tales peticiones.

Tras realizar estas pruebas hemos obtenido un 99.5% de cobertura donde la única línea en amarillo es la siguiente:



Esta línea se encuentra en el unbind de la clase. Estas líneas nos ayudan en la pagina a gestionar cuando se muestra y cuando no el botón de visualizar la lista de pasajeros de un booking. Se desconoce el motivo de que esta línea este en amarillo ya que se ha probado a hacer update en un booking el cual tenga pasajeron, viéndose así el botn en la pagina de visualizar pasajeros.

# **Delete**

Esta funcionalidad consiste en que un customer puede borrar sus bookings siempre y cuando estos no hayan sido publicados.

Para realizar las pruebas positivas y negativas, hemos primero probado a borrar bookigs no publicados que además no tengas pasajeros asociados (que no tenga booking records relacionados). Es decir, el único caso positivo, ya que los casos negativos no están permitidos porque no existe autorización para borrar un booking que este publicado o que tenga pasajeros asociados (no damos la opción del botón de borrar si esta publicado y si tiene pasajeros salta una mensaje de error que controlamos mediante un validator).

Para realizar las pruebas de hacking han seguido la misma línea que el Update, intentar borrar bookings con el rol inadecuado, intentar borrarlo con el mismo rol pero usuario inadecuado (tanto publicados como no publicados) e intentar borrar un booking publicado con el rol y usuario adecuado. En todos los casos el acceso ha sido denegado.

En este caso, tras realizar las pruebas hemos obtenido un 99.0% de cobertura donde la única línea que han salido en amarillo es la misma que se ha comentado en el caso anterior, siendo el mismo motivo de su aparición el explicado anteriormente.



# **Publish**

Esta funcionalidad consiste en que un customer puede publicar uno de sus booking siempre y cuando tenga al menos un pasajero publicado asociado, que todos sus pasajeros estén publicadas y que el lastNibble este escrito y sin errores.

Para llevar a cabo las pruebas positivas y negativas, primero hemos intentado publicar booking en donde pasajeros no estaban publicadas, otro donde no tenía pasajeros asociados, otro donde el booking no tenía un lastNibble valido asociado, otro cambiando ciertos atributos y otro probando casos en los que no eran validos (es decir, pruebas negativas). Tras comprobar que en todos estos casos nos han dado los errores correspondientes, hemos procedido a publicar un booking que sí que podíamos publicar.

Para llevar realizar las pruebas de hacking, hemos intentado realizarlo con roles que no eran de customer, luego con el rol de customer pero con el usuario inadecuado (para bookings publicados y no publicados) y, por último, con el rol y usuario correcto hemos intentado publicar bookings que ya estaban publicados. En todos los casos se nos ha denegado el acceso, por tanto, la aplicación ha funcionado como esperábamos.

Tras realizar estas pruebas hemos obtenido un 100% de cobertura asegurándonos de esta manera que no van a existir errores ya que se han ejecutado todas las líneas de código.

# **Funcionalidad para los customer sobre los pasajeros**

Vamos a continuar por las pruebas establecidas para las funcionalidades de los customer sobre los pasajeros.

# **List**

Esta funcionalidad consiste en que un customer pueda ver todos sus pasajeros y que se puedan visualizar los pasajeros dentro de un booking. Para realizar los casos de prueba positivos y negativos, nos hemos logueado como un customer y hemos accedido al listado de sus pasajeros y allí hemos comprobado que se listan todos sus pasajeros. Además, hemos accedido a un booking y hemos listado los psajeros de ese booking comprobando que todo era correcto.

Para realizar los casos de prueba de hacking, nos hemos logueado con un rol inadecuado intentado obtener un listado de pasajeros. A parte de esto se ha probado a acceder al listado de pasajeros de un booking cuando se estaba logueado con otro rol y con un usuario que no era propietario de ese booking, respondiendo la aplicación como esperábamos.

Tras realizar estas pruebas hemos obtenido un 100% de cobertura asegurándonos de esta manera que no van a existir errores ya que se han ejecutado todas las líneas de código.

# **Show**

Esta funcionalidad consiste en que un customer pueda ver las características/atributos de una de sus pasajeros.

Para realizar las pruebas positivas y negativas de esta funcionalidad nos hemos logueado como customer, hemos mostrado el listado de pasajeros y hemos ido pulsando en algunos pasajeros comprobando que aparecía la información correcta. También hemos accedido a un booking, hemos accedido a sus características y hemos accedido al listado de sus pasajeros y hemos centrado en uno.

Para realizar las pruebas de hacking, hemos intentado acceder a la información de un pasajero con un rol distinto al del customer (anónimo y manager) y luego hemos intentado acceder como un customer pero con el usuario incorrecto, es decir, a el pasajero de un customer distinto al que nos hemos logueado. En todos los casos han saltado los errores correspondientes.

Tras realizar estas pruebas hemos obtenido un 100% de cobertura asegurándonos de esta manera que no van a existir errores ya que se han ejecutado todas las líneas de código.

# **Create**

Esta funcionalidad consiste en que un customer puede crear un pasajero. Para realizar las pruebas positivas y negativas, hemos empezado primero por las negativas, mandando el formulario en blanco y luego más tarde tras rellenar todos los campos con valores que pasan los test hemos ido probando todos los fallos distintos posibles que podían darse para cada atributo. Tras probar todos los casos negativos posibles, nos hemos limitado a crear pasajeros probando con los rangos de valores posible como se ha explicado en la metodología de clase (probando los extremos de las restricciones, los diferentes valores que puede tomar…), siempre siendo valores permitidos.

Para realizar las pruebas de hacking, hemos probado a introducir la url para crear un pasajero con un rol distinto al de customer (anónimo y manager), comprobando que no dejaba acceder.

Tras realizar estas pruebas hemos obtenido un 100% de cobertura asegurándonos de esta manera que no van a existir errores ya que se han ejecutado todas las líneas de código.

# **Update**

Esta funcionalidad consiste en que un customer pueda actualizar un pasajero siempre y cuando este pasajero no haya sido publicado.

Para realizar las pruebas positivas y negativas, hemos llevado una metodología muy parecida a la llevada en el Create solo que en el caso de las positivas en vez de ir creándolos hemos ido actualizando los pasajeros.

Para realizar las pruebas de hacking, hemos intentado actualizar un pasajero con un rol diferente al de customer (manager y anónimo), después nos hemos logueado con un customer pero con un usuario distinto al del pasajero que hemos intentado actualizar (un pasajero publicado y otro no publicado). Por último, nos hemos logueado con un customer y con el usuario adecuado pero hemos intentado actualizar un pasajero que ya estaba publicado. En todos los casos han funcionado el sistema como esperábamos, rechazando tales peticiones.

Tras realizar estas pruebas hemos obtenido un 100% de cobertura asegurándonos de esta manera que no van a existir errores ya que se han ejecutado todas las líneas de código.

# **Delete**

Esta funcionalidad consiste en que un customer puede borrar un pasajero siempre y cuando este no haya sido publicado.

Para realizar las pruebas positivas y negativas, hemos primero probado borrar pasajeros no publicados sin bookings relacionados. Es decir, el único caso positivo, ya que los casos negativos no están permitidos porque no existe autorización para borrar una historia de usuario que este publicada o que tenga un booking asociado (no damos la opción del botón de borrar y si tiene un booking asociado salta una excepción controlada por nosotros mediante un validator).

Para realizar las pruebas de hacking han seguido la misma línea que el Update, intentar borrar pasajeros con el rol inadecuado, intentar borrarlo con el mismo rol pero usuario inadecuado (tanto publicados como no publicados) e intentar borrar un pasajero publicado con el rol y usuario adecuado. En todos los casos el acceso ha sido denegado.

Tras realizar estas pruebas hemos obtenido un 100% de cobertura asegurándonos de esta manera que no van a existir errores ya que se han ejecutado todas las líneas de código.

# **Publish**

Esta funcionalidad consiste en que un customer puede publicar un pasajero. Para realizar las pruebas positivas y negativas, hemos primero probado a publicar pasajeros modificando campos para que dieran errores y luego hemos pasado a publicar pasajeros no publicadas con bookings relacionados y después a pasajeros no publicados sin bookings relacionados.

Para llevar realizar las pruebas de hacking, hemos intentado realizarlo con roles que no eran de customer, luego con el rol de customer pero con el usuario inadecuado (para pasajeros publicados y no publicados) y, por último, con el rol y usuario correcto hemos intentado publicar pasajeros que ya estaban publicados. En todos los casos se nos ha denegado el acceso, por tanto, la aplicación ha funcionado como esperábamos.

Tras realizar estas pruebas hemos obtenido un 100% de cobertura asegurándonos de esta manera que no van a existir errores ya que se han ejecutado todas las líneas de código.

# **Funcionalidad para los customer sobre booking record (relación entre booking y passenger)**

Concluimos con las pruebas establecidas para las funcionalidades de los customer para añadir o eliminar pasajeros a un booking

# **Create**

Esta funcionalidad permite a los customer añadir pasajeros a un booking. Para ello solo podrán añadirse pasajeros del customer de ese booking (tiene que ser el mismo del de los pasajeros) y que esos pasajeros no estén ya añadidos.

Para los casos positivos, hemos añadido pasajeros (publicados y no publicados) a booking con pasajeros (publicados y no publicados) y a bookings sin pasajeros asociados anteriormente. Para los casos negativos hemos intentado añadir un pasajero nulo dando el error que esperábamos.

Para las pruebas de hacking, hemos intentado añadir pasajeros a proyecto estando logueados con un rol inadecuado, con el rol adecuado pero un usuario inadecuado (booking y pasajero publicado y no publicado) y con el rol y usuario adecuado pero estando el booking publicado. En todos los casos hemos tenido los errores que teníamos que obtener.

Tras estas pruebas la cobertura ha sido de un 95.1% esto se debe a la siguiente instrucción:

Texto

El contenido generado por IA puede ser incorrecto.

En este caso, esta instrucción nos ayuda a validar que no se cree ningún booking record ya existente, es decir, crear una asociación entre un booking y un pasajero que ya exista. Estos casos han sido probados en la colección de tests y después de intentar repetirlo numerosas veces no se ha conseguido cambiar de amarillo a verde. Aun así, se puede asegurar la integridad de la aplicación y su correcto funcionamiento en esta funcionalidad.

# **Delete**

Esta funcionalidad permite a los customer eliminar pasajeros de un booking. Para ello solo podrán eliminarse pasajeros de los bookings que no estén publicados.

Para los casos positivos, hemos eliminado pasajeros (publicados o no publicados) de bookings con pasajaros (publicados o no publicados). Para los casos negativos de esta funcionalidad no se ha tenido que probar nada ya que se deshabilita el botón de eliminar en los booking record de bookings publicados.

Para las pruebas de hacking, hemos intentado eliminar pasajeros de proyecto estando logueados con un rol inadecuado, con el rol adecuado pero un usuario inadecuado (booking publicado y no publicado) y con el rol y usuario adecuado pero estando el booking publicado. En todos los casos hemos tenido los errores que teníamos que obtener.

Tras estas pruebas la cobertura ha sido de un 98.6% esto se debe a que nunca se llega a acceder al método unbind y por tanto hay una línea en rojo que no podemos controlar:

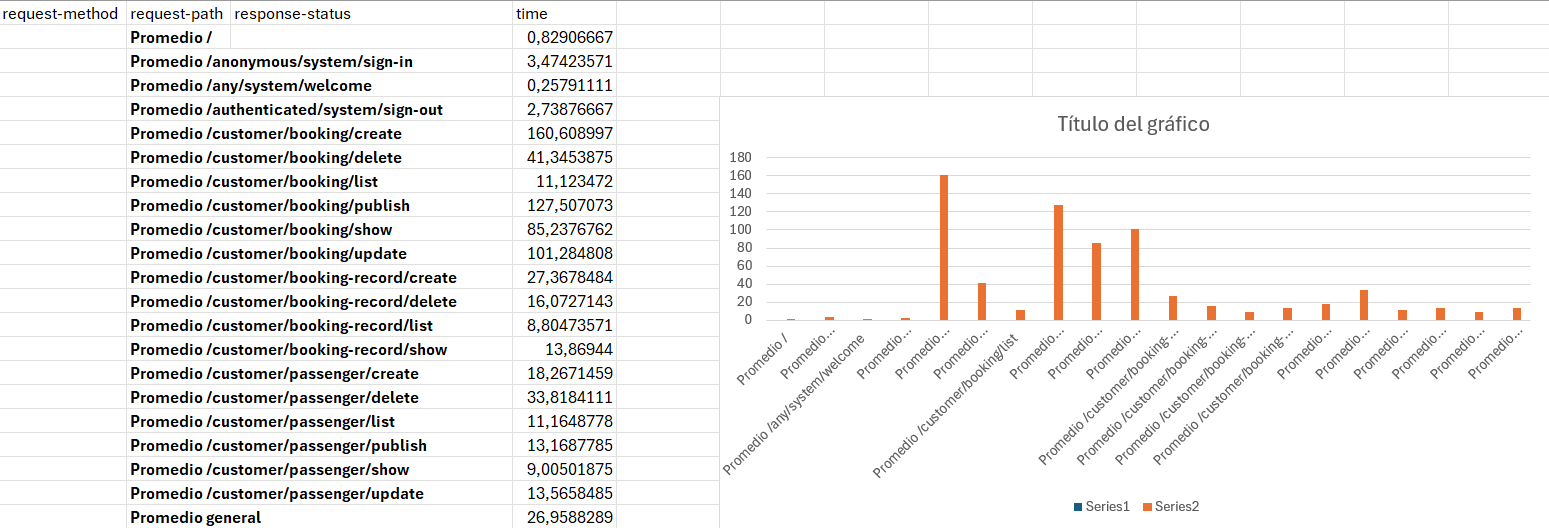
Texto

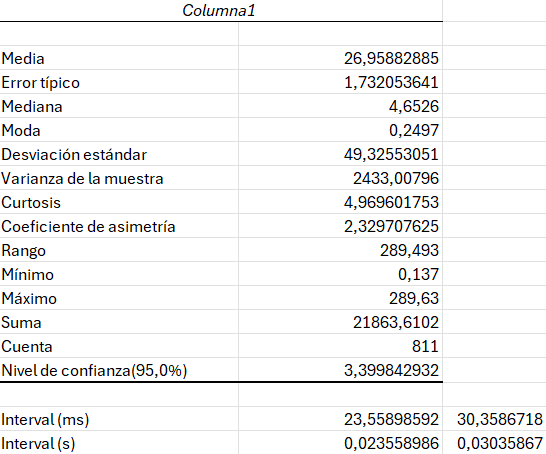
El contenido generado por IA puede ser incorrecto.

# **Testing de rendimiento**

Nos centramos ahora en el análisis del tiempo de respuesta del sistema al procesar solicitudes durante las pruebas funcionales. Se llevarán a cabo comparaciones entre nuestro portátil sin y con los índices para ver si estos han mejorado el rendimiento.

# **Sin índices**

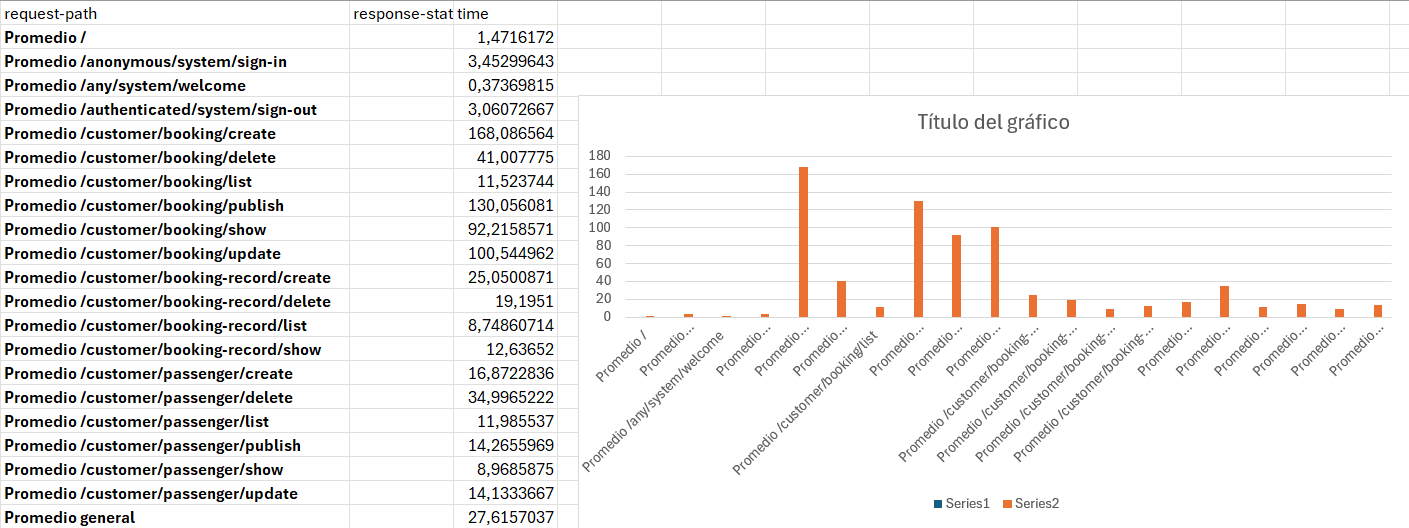
Tras aplicar los procedimientos explicados en las diapositivas hemos obtenido los siguientes tiempos para las diversas funcionalidades. Además, mostramos un gráfico donde podemos ver cuáles son las funcionalidades que tardan más en ejecutarse.

Y hemos obtenido de los datos los siguientes resultados después del análisis estadístico:

A continuación, en el siguiente apartado, veremos si los resultados con los índices ya incluidos han mejorado o no las prestaciones.

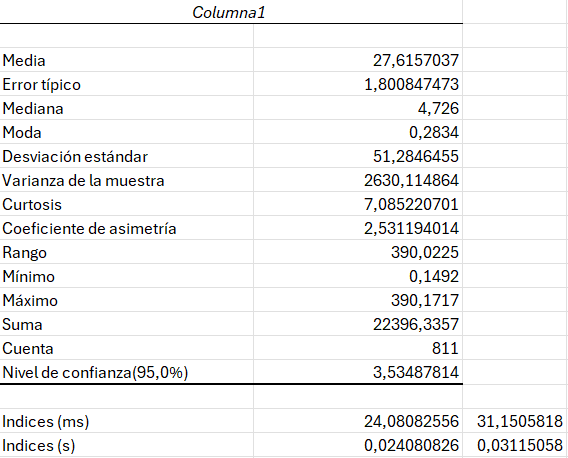
# **Con índices**

Estos son los tiempos obtenidos para las distintas funcionalidades además de un gráfico para resaltar cuales son las funcionalidades que tardan más en ejecutarse.

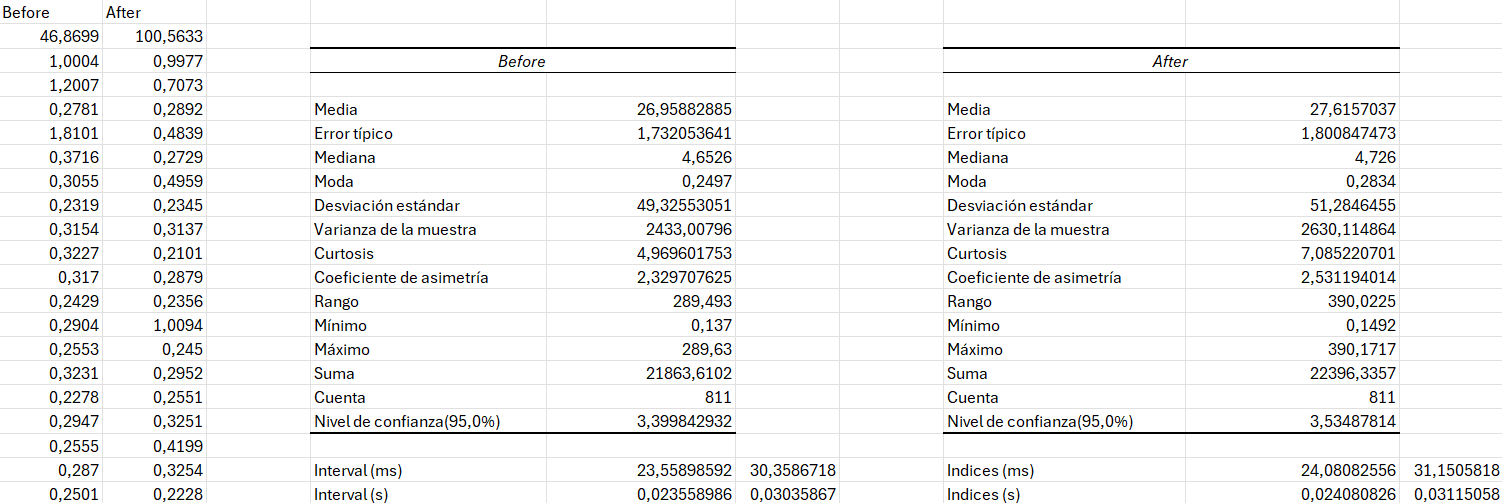


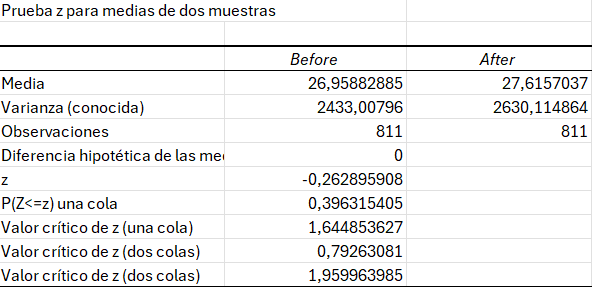
Como podemos observar el promedio general ha sido prácticamente igual (incluso un poco mayor) que sin utilizar los índices lo cual ya puede darnos un indicio de que no han mejorado las prestaciones.

Y hemos obtenido de los datos los siguientes resultados después del análisis estadístico:



**Comparación entre utilizar y no utilizar índices**

Nos disponemos a comparar los siguientes resultados:

Tras compararlos hemos obtenido lo siguiente resultados:

Como se observa en *Valor crítico de z (dos colas)***,** el valor está en el intervalo (Alpha, 1], lo que indica que las diferencias en los tiempos medios entre las dos ejecuciones no son significativas para el análisis del rendimiento así que podemos concluir el análisis.

# **Conclusiones**

Este informe presenta los resultados de las pruebas funcionales y de rendimiento realizadas en el proyecto. Las pruebas funcionales detallan diversos casos implementados para verificar las funcionalidades del sistema, evaluando su efectividad en la detección de errores. Las pruebas de rendimiento se enfocan en el análisis del tiempo de respuesta del sistema al procesar solicitudes en dos ordenadores diferentes, utilizando gráficos detallados e intervalos de confianza del 95% para asegurar la precisión de los resultados.

La sección de pruebas funcionales describe casos de prueba positivos, negativos y de hacking para cada funcionalidad del sistema, utilizando herramientas como Eclipse para asegurar una alta cobertura de código. En la mayor parte de los archivos se ha logrado alcanzar una cobertura de 100% siendo la mínima cobertura encontrada de 95,1%, un valor bastante próximo al 100%. Este análisis nos indica que nuestro código es robusto frente a ataques y que se ha evaluado la funcionalidad correcta de toda la aplicación.

Las pruebas de rendimiento compararon el tiempo de respuesta del sistema con y sin índices. Los resultados mostraron que el uso de índices no fue de utilidad para el análisis ya que los cambios del rendimiento no suficientemente significativos para su análisis.

En resumen, el informe confirma la efectividad del sistema en términos de funcionalidad y rendimiento. Estos resultados proporcionan una base sólida para la evaluación y mejora continua del proyecto.

# **Bibliografía**

Intencionalmente en blanco.