# Testing report (Kevin Amador Calzadilla)

Grupo C2.054 | Diseño y Pruebas II | 04/07/2025

Versiones

|  |  |  |
| --- | --- | --- |
| Fecha | Versión | Descripción |
| 26/05/2025 | 1.0 | Creación plantilla |
| 26/05/2025 | 2.0 | Entrega final |
| 02/07/2025 | 3.0 | Entrega extraordinaria |
| 15/10/2025 | 4.0 | Entrega tercera convocatoria |

Miembros:

Kevin Amador Calzadilla (kevamacal@alum.us.es)

Manuel Zurita Fernández (manzurfer@alum.us.es)

Repositorio: <https://github.com/DP2-C1-054/Acme-ANS-C3>

# Índice:

Resumen ejecutivo………………………………………………………………………1

Introducción……………………………………………………………………………..2

Contenido………………………………………………………………………………..5

Pruebas funcionales……………………………………………………………...5

Análisis de rendimiento………………………………………………………...12

Conclusiones…………………………………………………………………………....15

Bibliografía……………………………………………………………………………..16

# Resumen ejecutivo

En este informe, presento las pruebas funcionales y de rendimiento que realicé en mi estudiante. El objetivo era garantizar que todas las características se comportaran como se esperaba y evaluar la rapidez de respuesta del sistema en condiciones normales.

Para las pruebas de rendimiento, seguí la metodología descrita en la guía de la sesión: recopilé los tiempos de ejecución de los archivos de seguimiento y procesé los datos con Excel. Generé gráficos y calculé intervalos de confianza del 95 % para evaluar si los tiempos de respuesta del sistema se mantenían dentro de límites aceptables. Las pruebas se ejecutaron en dos configuraciones diferentes: una utilizando la base de datos sin indexación adicional y otra con los índices pertinentes aplicados y una optimización adicional mediante la aplicación del contenido del curso sobre creación de perfiles de software y hardware. Posteriormente, realicé una comparación estadística entre ambas configuraciones para determinar el impacto de la indexación en el rendimiento.

# Introducción

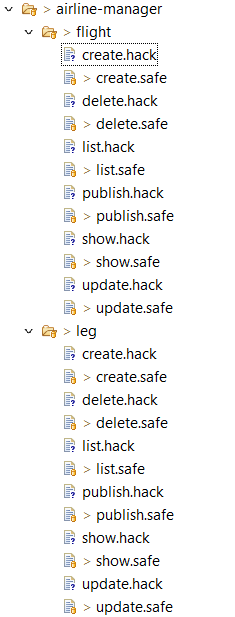
El propósito de este informe es documentar las pruebas que realicé en mi proyecto, abarcando tanto la funcionalidad como el rendimiento. Mi objetivo era asegurar que cada característica funcionara según lo previsto y que el sistema pudiera gestionar las solicitudes eficientemente.

Para lograrlo, primero ejecuté un conjunto de pruebas funcionales agrupadas por característica, verificando que la aplicación se comportara como se esperaba. Después, me centré en las pruebas de rendimiento analizando el tiempo que tardaba el sistema en responder a diversas solicitudes. Esto me ayudó a identificar posibles cuellos de botella y a comparar el rendimiento en diferentes configuraciones.

En general, esta fase de pruebas me proporcionó una comprensión más clara del comportamiento del sistema en condiciones reales.

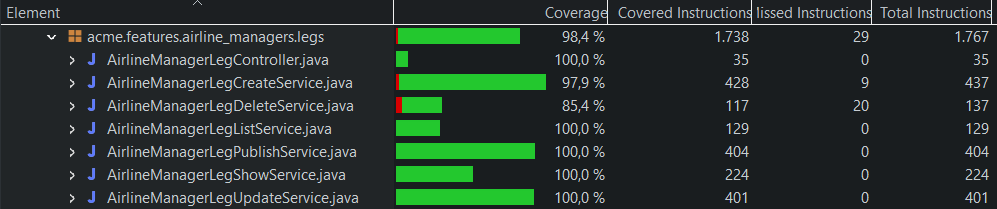
# Contenido

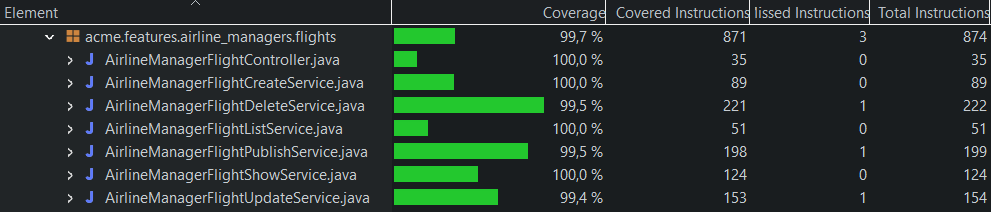
## Pruebas funcionales



Para estas pruebas, evalué las funcionalidades que mi proyecto, Acme-ANS, ofrece a los clientes, específicamente en lo que respecta a sus vuelos y tramo. Las pruebas se documentan mediante archivos de seguimiento ubicados en los directorios /src/test/resources/airline-manager/flights y /src/test/resources/airline-manager/legs. Los casos de pruebas tuvieron aproximadamente un 98% de cobertura de código exceptuando diversos casos como es el de delete, como se puede observar en la siguiente imagen:

Estos son los archivos de seguimiento que demuestran la cobertura del código, incluidos a continuación, y creados siguiendo las recomendaciones del curso. Probé tanto los seguimientos .safe (que abarcan casos de prueba positivos y negativos) como los seguimientos .hack, que simulan intentos de pirateo de la aplicación. Todos los intentos de pirateo se gestionaron correctamente y generaron una respuesta de "Acceso no autorizado".





A continuación se muestra una breve explicación de cada uno de los tests implementados para probar las funcionalidades:

FLIGHT

- list.safe: Verifica la capacidad de un manager de vuelos para listar sus vuelos. La prueba confirma que todos los vuelos asociados con el manager1 se recuperan y muestran correctamente.

- show.safe: Garantiza la correcta visualización de los detalles de los vuelos del manager1, verificando que la información individual de cada vuelo se represente correctamente.

- create.safe: Prueba la creación de vuelos utilizando datos de entrada válidos. Los mecanismos de validación del sistema se evalúan intentando crear un vuelo con campos vacíos, esperando los mensajes de error correspondientes.

- update.safe: Los datos de varios vuelos del manager1 se actualizan utilizando tanto las entradas válidas aceptadas por el sistema como las no válidas, de forma similar a las pruebas de create.safe.

- delete.safe: Confirma que el manager1 puede eliminar sus vuelos y que las entidades asociadas también se eliminan. El sistema se prueba para garantizar que los vuelos publicados no se puedan eliminar. Además, se prueba que no pueda eliminar ningún vuelo con tramos ya publicados.

- publish.safe: Evalúa la capacidad de publicar vuelos. Publica correctamente vuelos válidos y verifica que los vuelos sin tramos o con tramos no publicados no se puedan publicar, mostrando los mensajes de error correspondientes.

- list.hack: Comprueba los intentos de acceso no autorizado mediante la inserción de IDs no válidos o no autorizados en la URL.

- show.hack: Comprueba los intentos de acceso no autorizado mediante la inserción de IDs no válidos o no autorizados en la URL, incluyendo vuelos inexistentes o pertenecientes a otros administradores. Se intenta acceder a vuelos no publicados con la sesión iniciada con otro administrador.

- create.hack: Verifica la protección del sistema contra creaciones no autorizadas modificando los parámetros de la solicitud desde otro tipo de usuarios.

- update.hack: Verifica la protección del sistema contra actualizaciones no autorizadas modificando los parámetros de la solicitud desde otro tipo de usuarios.

- delete.hack: Verifica la protección del sistema contra borrados no autorizados modificando los parámetros de la solicitud desde otro tipo de usuarios.

- publish.hack: Verifica la protección del sistema contra publicaciones no autorizadas modificando los parámetros de la solicitud desde otro tipo de usuarios.

LEGS

- list.safe: Verifica que un manager de vuelos pueda recuperar y listar todos los registros de sus tramos asociados. Esta prueba confirma que los tramos vinculados al manager1 se obtienen y muestran correctamente.

- show.safe: Garantiza la correcta visualización de los detalles de los tramos del manager1, verificando que la información individual de cada tramo se represente correctamente.

- create.safe: Prueba la creación de tramos utilizando datos de entrada válidos aceptados por el sistema. También incluye intentos de crear un tramo sin entrada (esperando mensajes de error de validación apropiados). Otros escenarios prueban la respuesta del sistema a entradas no válidas intentar usar un número de vuelo duplicado, etc. Se espera que cada uno de estos casos devuelva respuestas de error relevantes e informativas.

- update.safe: Los datos de varios tramos del manager1 se actualizan utilizando tanto entradas válidas aceptadas por el sistema como entradas no válidas, de forma similar a las pruebas de create.safe.

- delete.safe: confirma que el manager1 puede eliminar correctamente los registros de los tramos seleccionados.

- publish.safe: Prueba la publicación de tramos no publicados previamente utilizando tanto entradas válidas aceptadas por el sistema como entradas no válidas, de forma similar a las pruebas de create.safe.

- list.hack: Se realizan pruebas de piratería GET accediendo a las URL asociadas con los managers de otros vuelos y de un usuario no registrado.

- show.hack: Se realizan pruebas de piratería GET accediendo a las URL asociadas con los managers de otros vuelos y de un usuario no registrado. También se intenta acceder a los tramos registrados por otros managers de manera que se prueba que efectivamente el sistema detecta un error de autenticación al realizar una de estas acciones no permitidas.

- create.hack: Simula la manipulación del lado del cliente mediante herramientas de desarrollo del navegador (p. ej., F12) para modificar los ID de relación (como aeronaves o aeropuertos) durante la creación de un tramo. Estas pruebas confirman que el sistema detecta y bloquea los intentos ilegítimos, mostrando los mensajes de error correspondientes.

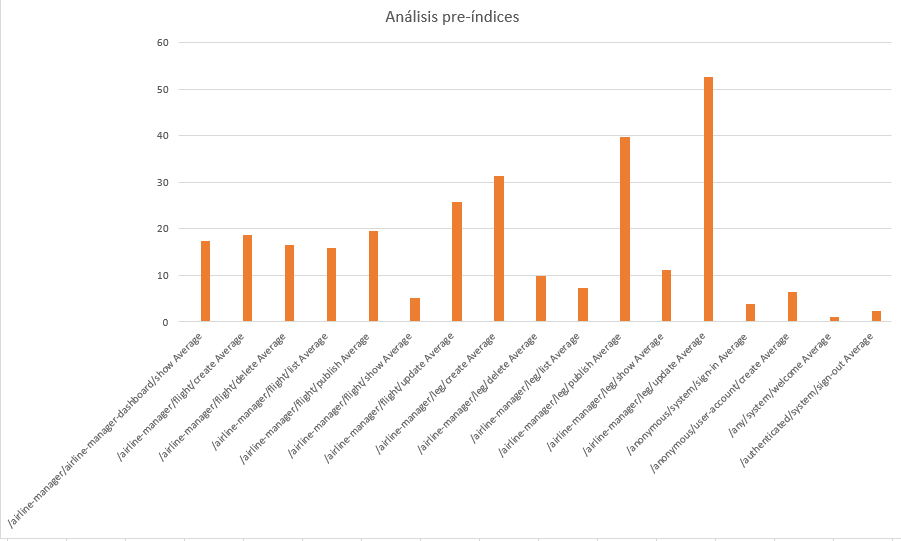
- update.hack: Al igual que create.hack, esta prueba implica solicitudes POST con ID de relación modificados mediante herramientas de desarrollo. Se deben evitar los intentos de actualizar tramos no autorizados o inexistentes y responder con las respuestas de validación adecuadas.

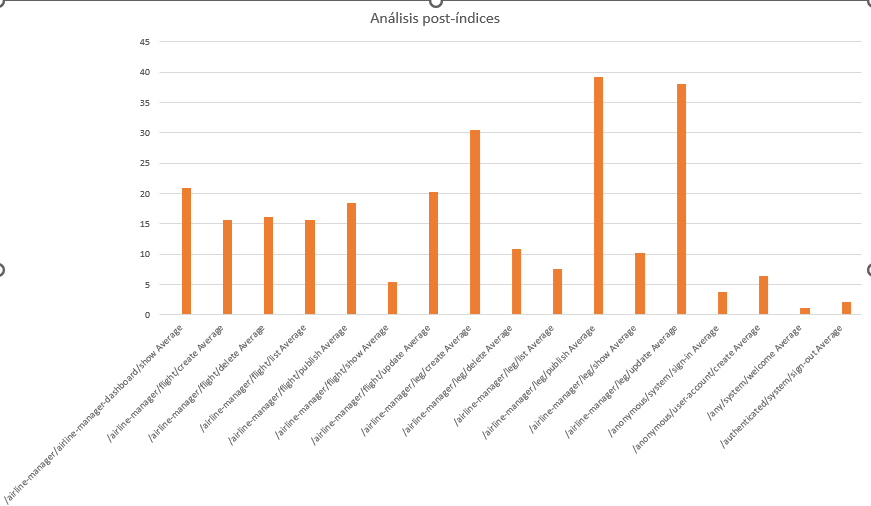
- delete.hack: Intenta eliminar registros de tramos sin autorización mediante la manipulación de los parámetros de la solicitud o los ID de relación. Las pruebas se realizan con la sesión iniciada como usuario no autorizado o sin haber iniciado sesión. El sistema debe evitar estas eliminaciones.

- publish.hack: Al igual que create.hack, esta prueba implica solicitudes POST con ID de relación modificados mediante herramientas de desarrollo. Se deben evitar los intentos de publicar tramos no autorizados o inexistentes y responder con las respuestas de validación adecuadas.

## Análisis de rendimiento

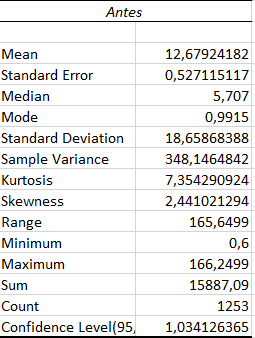
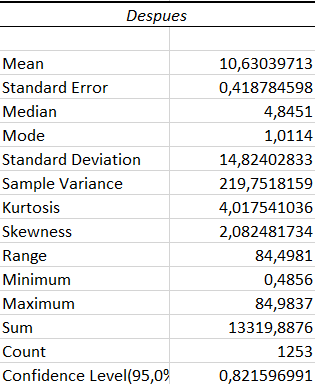
Las pruebas de rendimiento se realizaron en el mismo ordenador. Para ello, todos los rastros mostrados anteriormente se ejecutaron dos veces: primero sin índices de base de datos y, una segunda vez, con índices, con el objetivo de mejorar el rendimiento reduciendo el tiempo de resolución de las consultas. A continuación, se muestran las tablas de tiempos de ambos análisis:

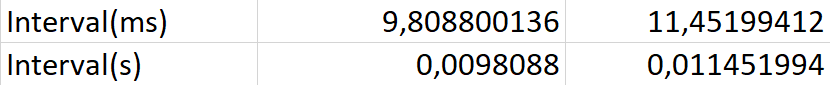




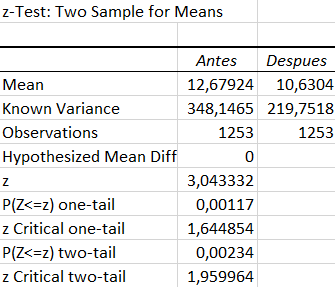
Se ha observado una mejora notable en el rendimiento gracias a la incorporación de algunos índices bien posicionados, sobre todo, se puede observar en publish serivce, dado que hubo una mejoría bastante notable.

Esta mejora se debe principalmente a la aplicación de las técnicas explicadas en el material del curso sobre creación de perfiles de software y hardware, lo que ayudó a identificar áreas de optimización, logrando así un rendimiento mucho mayor.

Se puede observar en las anteriores imágenes que el intervalo de confianza disminuye y por tanto se sigue confirmando lo comentado anteriormente, es decir, que los cambios si tuvieron efectos significativos en la aplicación.



En la prueba z de dos muestras para medias, se obtuvo un valor estadístico z = 3,043332, con un valor p de dos colas igual a 0,00234. Este p-value representa la probabilidad de observar una diferencia entre medias obtenida (12,67 vs. 10,63) si en realidad no existiera ninguna diferencia.

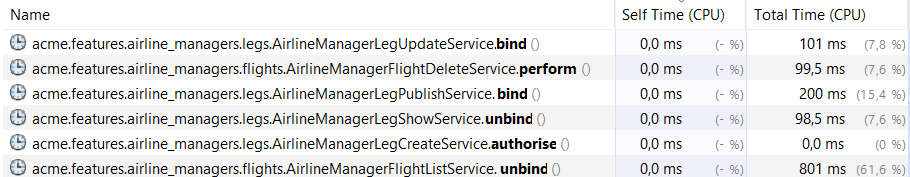
Dado que el valor p es mucho menor que el nivel de significancia α = 0,05, se rechaza la hipótesis nula, lo cual indica que la diferencia en rendimiento es estadísticamente significativa.

En conclusión, los cambios aplicados al sistema tuvieron un impacto real y medible en la mejora del rendimiento, y la diferencia observada no se debe al azar.

Tras ello, se llevaron a cabo pruebas de rendimiento del tipo profiling, en primer lugar del

software:

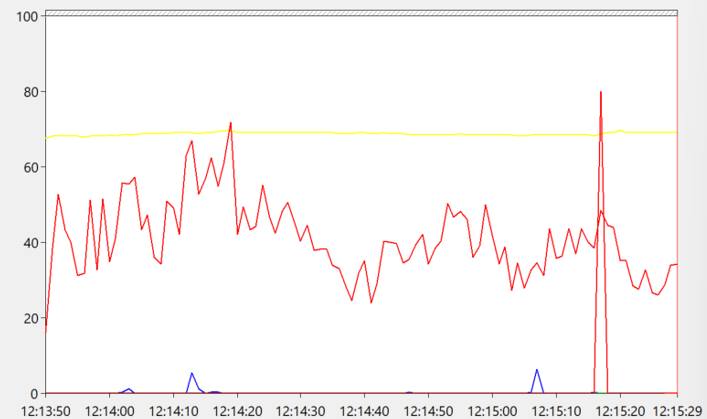


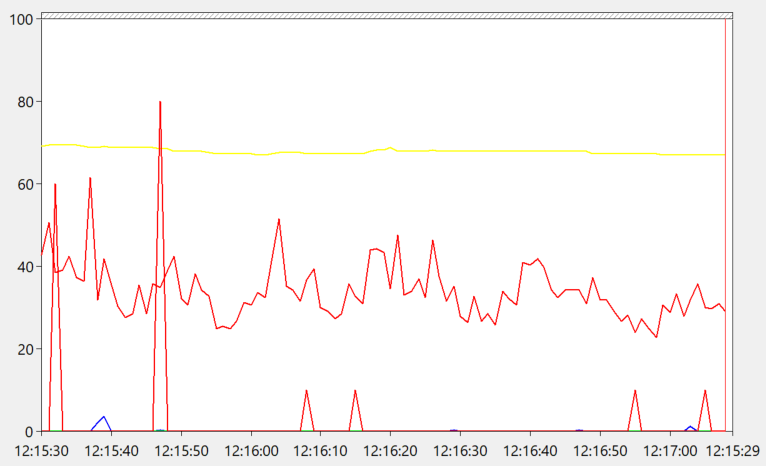


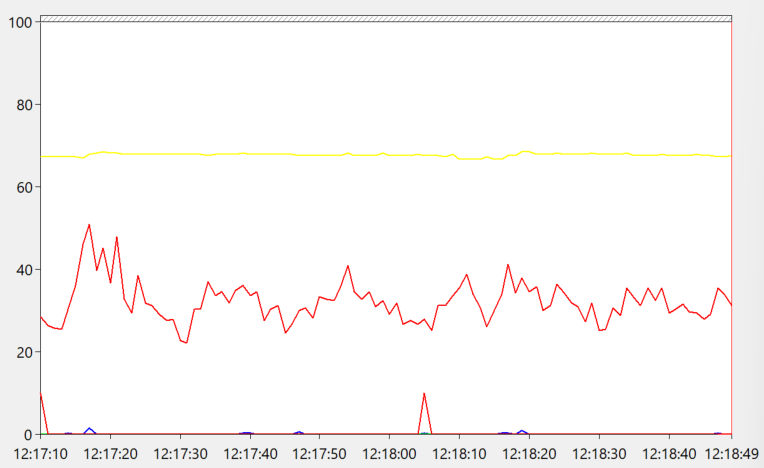
Se puede observar como tras aplicar los índices también hubo una gran mejora de rendimiento .

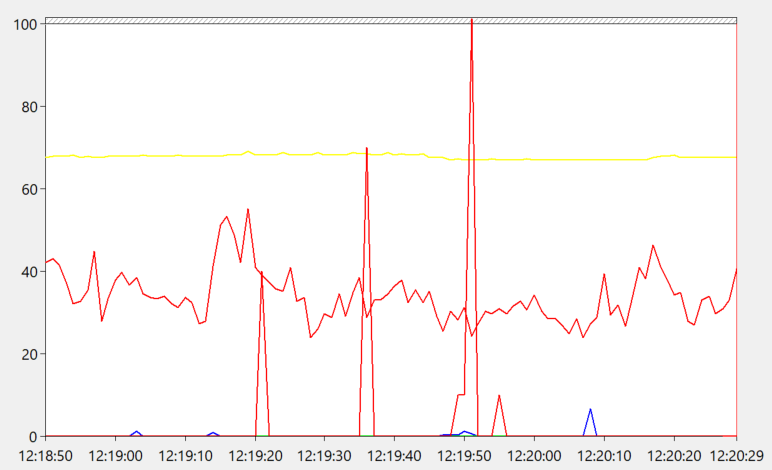
Por último, se llevaron a cabo las pruebas de hardware profiling:

SIN INDICES

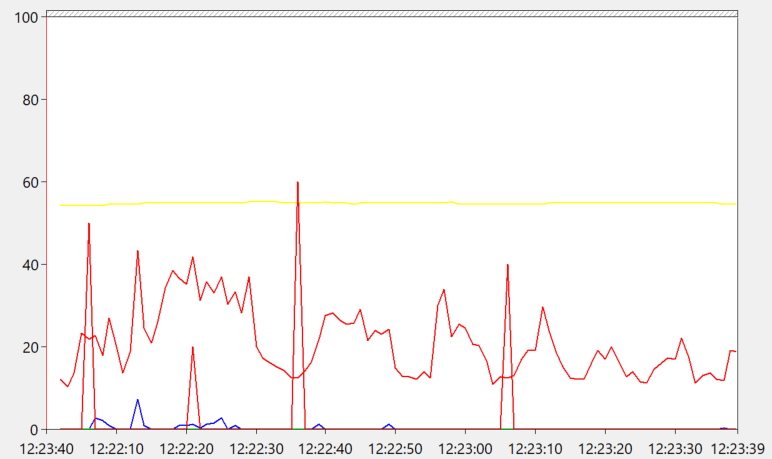


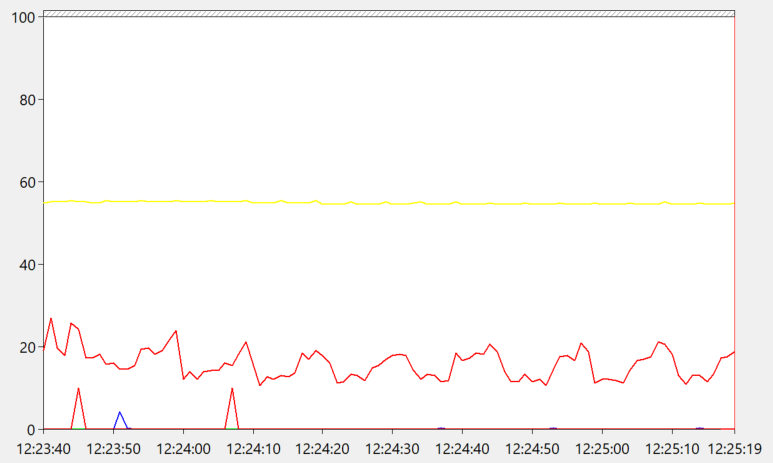


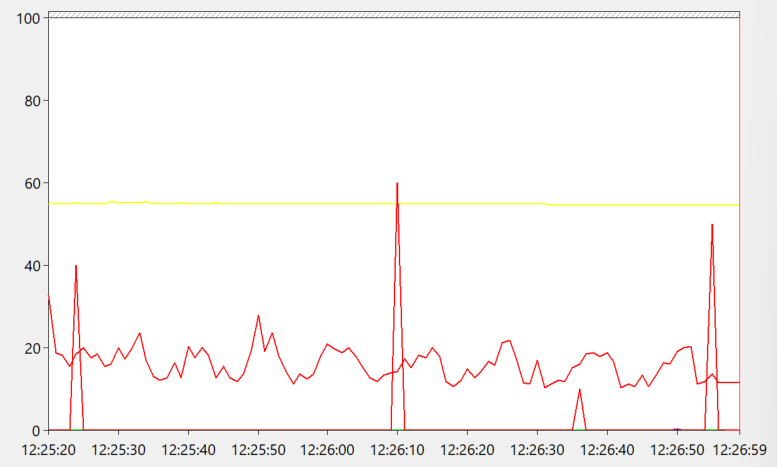




CON INDICES







Las gráficas demuestran que, al incorporar índices, se reduce significativamente la carga sobre los recursos del sistema (especialmente CPU y disco), y se logra una mayor estabilidad durante la ejecución de consultas. En contraste, la ausencia de índices genera cuellos de botella y obliga al sistema a trabajar con mayor esfuerzo, lo que puede derivar en tiempos de respuesta más altos y una menor eficiencia general.

# Conclusiones

A lo largo de este proceso de pruebas, se ha verificado de manera exhaustiva tanto el correcto funcionamiento de las funcionalidades principales del sistema como su rendimiento bajo diferentes configuraciones. Las pruebas funcionales han demostrado que el sistema responde adecuadamente ante escenarios tanto válidos como maliciosos, cumpliendo con los requisitos de seguridad y restricciones de acceso, y alcanzando una cobertura de código cercana al 98 %.

Por otro lado, el análisis de rendimiento ha puesto de manifiesto la importancia de una correcta optimización mediante el uso de índices en la base de datos. La incorporación de estos índices ha supuesto una mejora significativa en los tiempos de respuesta del sistema, como se ha demostrado con el análisis estadístico y las pruebas de profiling. El valor p obtenido y la reducción del intervalo de confianza avalan que los cambios aplicados no solo son medibles, sino también significativos desde el punto de vista estadístico y práctico.

En definitiva, este proceso de testing ha permitido validar la robustez, eficiencia y fiabilidad del sistema, asegurando que está preparado para operar en condiciones reales de uso con un alto grado de confianza.

# Bibliografía

Intencionalmente en blanco