

# Testing Report

- **Group Number:** C1.047
- **Repository:** <https://github.com/JoaquinBorjaLeon/C1-047-Acme-ANS-D04>
- **Workgroup Members and Corporate Emails:**

Joaquín Borja León: [joaborleo@alum.us.es](mailto:joaborleo@alum.us.es)

Ariel Escobar Capilla: [ariesccap@alum.us.es](mailto:ariesccap@alum.us.es)

Héctor Guerra Prada: [hecguepra@alum.us.es](mailto:hecguepra@alum.us.es)

Juan Carlos León Madroñal: [jualeomad@alum.us.es](mailto:jualeomad@alum.us.es)

José Ángel Rodríguez Durán: [josroddur@alum.us.es](mailto:josroddur@alum.us.es)

- **Date:** 26/05/2025

## Tabla de contenido

### Contenido

1.Executive Summary.....	1
2.Revision Table .....	2
3.Contents.....	2
4.Performance testing.....	4
5.Software Profiling .....	8
6.Hardware Profiling .....	9

## 1.Executive Summary

En este informe presento las pruebas funcionales y de rendimiento que llevé a cabo sobre mi estudiante. El objetivo fue asegurar que todas las funcionalidades se comportaran según lo esperado y evaluar la rapidez con la que responde el sistema en condiciones normales.

Para las pruebas funcionales, organicé los casos de prueba por funcionalidad. Cada caso se centra en una función específica y fue esencial para verificar que la aplicación se comporta correctamente.

Para las pruebas de rendimiento, seguí la metodología indicada en la guía de la sesión: recopilé los tiempos de ejecución a partir de los archivos .trace y procesé los datos utilizando Excel. Generé gráficos y calculé intervalos de confianza del 95 % para evaluar si los tiempos de respuesta del sistema se mantenían dentro de los límites aceptables. Las pruebas se ejecutaron en dos configuraciones distintas: una utilizando la base de datos sin ningún índice adicional, y otra con los índices relevantes aplicados. Posteriormente, realicé una comparación estadística entre ambas configuraciones para determinar el impacto del uso de índices en el rendimiento.

En resumen, este informe refleja las pruebas que he realizado sobre las funcionalidades clave de la aplicación, respaldadas por datos de rendimiento que ofrecen una comprensión sólida del comportamiento del sistema en condiciones reales.

## 2.Revision Table

Revision Number	Date	Description
1.0	[25/05/2025]	Initial version

## 3.Contents

















A continuación se presenta una lista de pruebas funcionales realizadas para las funcionalidades de FlightAssignment:

- **List.safe:** Verifica que los usuarios puedan ver todas las asignaciones de vuelo disponibles para ellos. Asegura la recuperación y visualización correcta de los registros pertenecientes al usuario autenticado.
- **ListPlanned.safe:** Comprueba que solo se muestren las asignaciones planificadas, es decir, aquellas cuyo vuelo no ha aterrizado.
- **Show.safe:** Garantiza que los detalles de una asignación específica se muestren correctamente, siempre que la asignación pertenezca al usuario.
- **Create.safe:** Prueba la creación de una nueva asignación de vuelo utilizando datos válidos. También se evalúan los mecanismos de validación del sistema al introducir campos vacíos o datos inválidos.
- **Update.safe:** Verifica que una asignación pueda modificarse correctamente mientras esté en modo borrador (*draftMode = true*), utilizando tanto entradas válidas como inválidas.
- **Publish.safe:** Evalúa el proceso de publicación de una asignación. Confirma que una asignación en modo borrador puede ser publicada correctamente y que, una vez publicada, ya no puede ser modificada. También se prueban restricciones

relacionadas con intentos de publicar asignaciones invalidas como que solo pueda haber un piloto y copiloto, que el miembro esté disponible, que la leg no este ya aterrizada.

- **Show.hack:** Se prueban intentos de acceso no autorizado a los detalles de asignaciones mediante la manipulación de URLs con IDs inválidos o de asignaciones que pertenecen a otros usuarios.
- **Create.hack:** Simula el uso indebido de herramientas de desarrollo del navegador para alterar manualmente datos del formulario y forzar la creación de asignaciones con IDs ya existentes o datos restringidos.
- **Update.hack:** Comprueba que el sistema impida la modificación de asignaciones que no estén en modo borrador, que no existan o que pertenezcan a otros usuarios. Se manipulan tanto las URLs como los cuerpos de las solicitudes.
- **Publish.hack:** Verifica que no sea posible publicar asignaciones ajenas, ya publicadas o con datos invalidos, mediante la modificación de parámetros o rutas en las solicitudes.
- **List.hack / ListPlanned.hack:** Se simulan intentos de listar asignaciones pertenecientes a otros usuarios o acceder a listados restringidos, manipulando los filtros o los endpoints.

La cobertura de test alcanzada de la entidad flightAssignment ha sido del 97.8 por ciento

















▼  acme.features.flightcrewmember.flightassignment		97,8 %	1.682	38	1.720
>  FlightAssignmentUpdateService.java		96,5 %	384	14	398
>  FlightAssignmentsCreateService.java		97,0 %	386	12	398
>  FlightAssignmentPublishService.java		98,4 %	562	9	571
>  FlightAssignmentShowService.java		98,1 %	154	3	157
>  FlightAssignmentsController.java		100,0 %	36	0	36
>  FlightAssignmentsLegLandedListService.java		100,0 %	79	0	79
>  FlightAssignmentsLegPlannedListService.java		100,0 %	81	0	81

Todas las líneas han sido testeadas aunque algunas por diseño de código solo se han ejecutado de forma parcial.

Pruebas funcionales realizadas para las funcionalidades de ActivityLog:

- **List.safe:** Verifica que el usuario pueda ver todos los registros de actividad que ha creado. Asegura que se muestren únicamente los registros pertenecientes al usuario autenticado.
- **Show.safe:** Garantiza que se puedan visualizar correctamente los detalles de un registro específico, siempre que pertenezca al usuario.
- **Create.safe:** Evalúa la creación de un nuevo registro de actividad asociado a una asignación de vuelo válida. Se prueban también casos con datos incompletos o inválidos para comprobar la validación del sistema.
- **Update.safe:** Prueba la modificación de un registro existente en modo borrador. Se realizan pruebas tanto con datos válidos como inválidos para asegurar el correcto funcionamiento de las validaciones.
- **Delete.safe:** Confirma que un registro de actividad puede eliminarse únicamente si aún no ha sido publicado. Se asegura que los registros ya publicados no puedan ser eliminados.

- **Show.hack:** Simula intentos de acceder a registros que no pertenecen al usuario o que no existen, manipulando directamente los IDs en la URL.
- **Create.hack:** Intenta asociar nuevos registros de actividad a asignaciones de vuelo que no pertenecen al usuario o que no están disponibles, mediante la alteración del payload de la solicitud.
- **Update.hack:** Comprueba que no sea posible modificar registros publicados, ajenos o inexistentes, manipulando los parámetros de la solicitud.
- **Delete.hack:** Verifica que el sistema impida eliminar registros ya publicados o que no pertenezcan al usuario, modificando el ID en la URL o el cuerpo de la petición.
- **List.hack:** Evalúa si el sistema filtra correctamente los registros visibles, impidiendo que un usuario vea registros de actividad de otros mediante manipulación de filtros o rutas.

▼  acme.features.flightcrewmember.activitylog		97,9 %	914	20	934
>  ActivityLogListService.java		85,3 %	81	14	95
>  ActivityLogCreateService.java		97,3 %	218	6	224
>  ActivityLogController.java		100,0 %	35	0	35
>  ActivityLogDeleteService.java		100,0 %	156	0	156
>  ActivityLogPublishService.java		100,0 %	153	0	153
>  ActivityLogShowService.java		100,0 %	118	0	118
>  ActivityLogUpdateService.java		100,0 %	153	0	153

La cobertura de test alcanzada en este paquete es de 97.9%, todas las líneas han sido testeadas aunque algunas por diseño de código solo se han ejecutado de forma parcial.

## 4. Performance testing

Posteriormente, se han llevado a cabo pruebas de rendimiento con el objetivo de optimizar el acceso a los datos y mejorar los tiempos de respuesta de los distintos servicios involucrados. Para ello, se han introducido índices en los campos clave de las entidades, acelerando de forma significativa las consultas a la base de datos, especialmente en las consultas de filtrado y búsqueda.

Como resultado, se ha conseguido una mejora notable en los tiempos de ejecución, lo que impacta positivamente en la experiencia del usuario y en la eficiencia general del sistema.

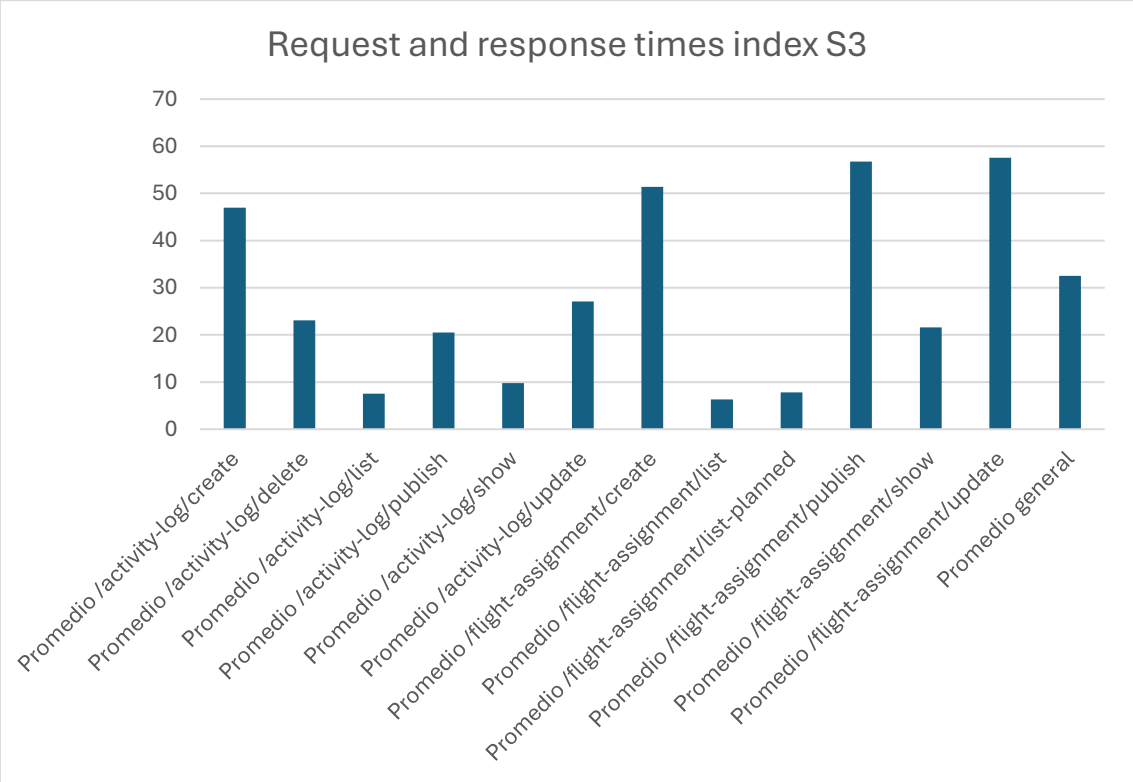
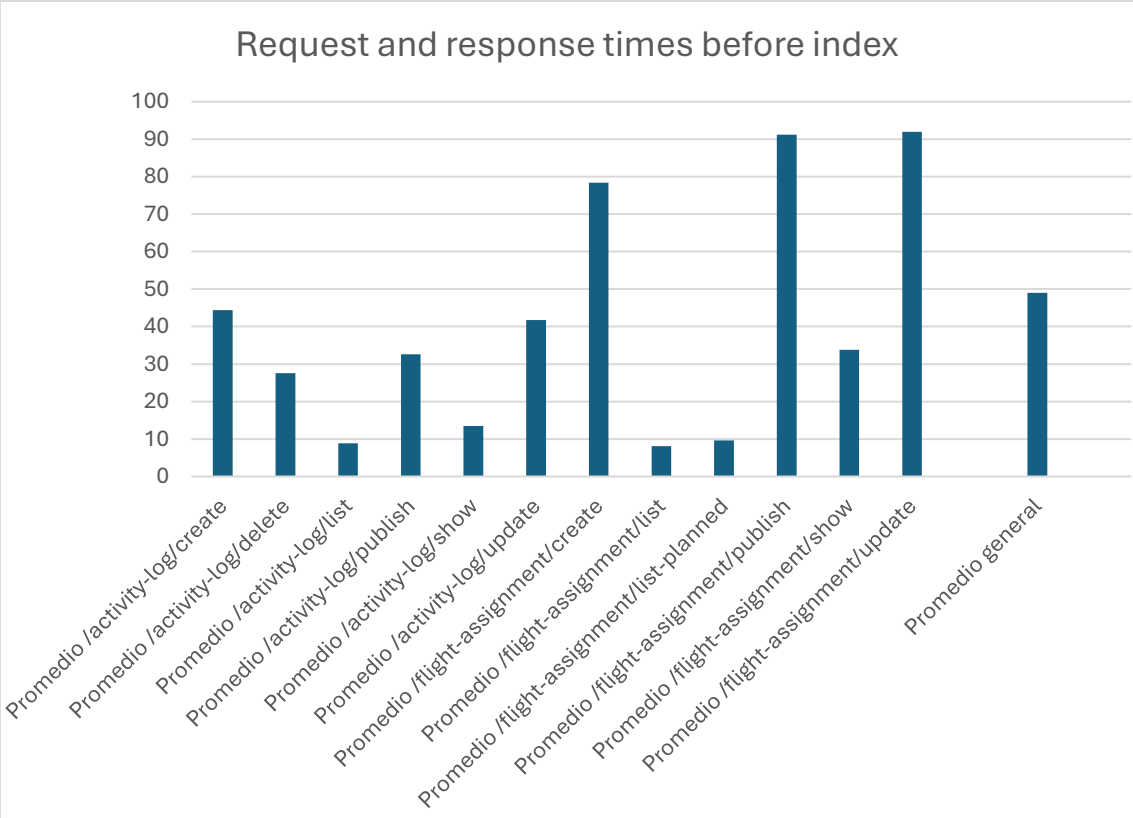
### Análisis de la mejora

Para complementar las pruebas de rendimiento, se ha realizado un análisis estadístico utilizando Excel y la prueba Z (Z-test) con el objetivo de validar si la mejora en los tiempos de respuesta tras la introducción de los índices es estadísticamente significativa.

El resultado del análisis ha arrojado un valor  $P_z < 0.00$ , lo cual permite rechazar la hipótesis nula y concluir que los cambios aplicados han tenido un efecto significativo y positivo en el rendimiento del sistema.

Este resultado se refleja de forma clara en las gráficas de tiempos, donde se observa que servicios como FlightAssignment -> publish y update, que son los métodos que más

consultas realizan a los repositorios y poseen queries más complejas, superaban los 90 milisegundos antes de la optimización. Tras la introducción de los índices, dichos tiempos se reducen de forma consistente, apenas alcanzando los 60 milisegundos.



<i>After</i>		<i>Before</i>	
Media	34,6336991	Media	48,6129188
Error típico	2,06163028	Error típico	2,80996676
Mediana	19,8632	Mediana	27,6299
Moda	#N/D	Moda	#N/D
Desviación es	32,7922371	Desviación es	44,6952574
Varianza de la	1075,33081	Varianza de la	1997,66604
Curtosis	1,89654044	Curtosis	-0,55669657
Coeficiente de	1,32219718	Coeficiente de	0,84391935
Rango	187,9101	Rango	199,3698
Mínimo	3,9914	Mínimo	4,2034
Máximo	191,9015	Máximo	203,5732
Suma	8762,32587	Suma	12299,0685
Cuenta	253	Cuenta	253
Nivel de confia	4,06022079	Nivel de confia	5,53401138
Interval (ms)	30,5734783	Interval (ms)	43,0789074
Interval(s)	0,03057348	Interval(s)	0,04307891

Prueba z para medias de dos muestras		
	<i>After</i>	<i>before</i>
Media	34,63369908	48,6129188
Varianza (conocida)	1075,330814	1997,66604
Observaciones	253	253
Diferencia hipotética de las medias	0	
z	-4,011087476	
P(Z<=z) una cola	3,02199E-05	
Valor crítico de z (una cola)	1,644853627	
Valor crítico de z (dos colas)	6,04397E-05	
Valor crítico de z (dos colas)	1,959963985	

# 5.Software Profiling

Además de las pruebas funcionales y de rendimiento, se ha realizado un test de Software Profiling con el objetivo de analizar el consumo de recursos y la eficiencia del sistema a nivel de ejecución. Esta técnica ha permitido identificar los métodos con mayor carga de procesamiento, el uso de memoria y la distribución del tiempo de CPU entre las distintas operaciones.

El *profiling* ha sido útil para confirmar que los puntos críticos del sistema (como los métodos publish y update de FlightAssignment) no solo hacen más llamadas a repositorio, sino que también concentran una mayor parte del uso de recursos.

Como puede observarse en las dos capturas adjuntas, la introducción de índices en la base de datos ha tenido un efecto directo y positivo en el rendimiento del sistema. Las gráficas y métricas muestran de forma clara que los tiempos de respuesta son significativamente menores cuando los índices están aplicados, en comparación con la versión sin optimización.

Los métodos más exigentes, como publish y update en la entidad FlightAssignment, reducen su tiempo de ejecución tras la aplicación de los índices, demostrando así que el sistema responde más rápido y con mayor eficiencia en los puntos donde más lo necesita.

Sin índices:

Name	Self Time (CPU)	Total Time (CPU)
acme.features.flightcrewmember.flightassignment.FlightAssignmentUpdateService. <b>unbind</b> ()	0,0 ms (- %)	775 ms (13,4 %)
acme.features.flightcrewmember.flightassignment.FlightAssignmentUpdateService. <b>validate</b> ()	0,0 ms (- %)	601 ms (10,4 %)
acme.features.flightcrewmember.flightassignment.FlightAssignmentsCreateService. <b>unbind</b> ()	0,0 ms (- %)	501 ms (8,7 %)
acme.features.flightcrewmember.flightassignment.FlightAssignmentPublishService. <b>unbind</b> ()	0,0 ms (- %)	501 ms (8,6 %)
acme.features.flightcrewmember.flightassignment.FlightAssignmentsCreateService. <b>authorise</b> ()	0,0 ms (- %)	498 ms (8,6 %)
acme.features.flightcrewmember.flightassignment.FlightAssignmentPublishService. <b>authorise</b> ()	0,0 ms (- %)	401 ms (6,9 %)
acme.features.flightcrewmember.flightassignment.FlightAssignmentShowService. <b>unbind</b> ()	0,0 ms (- %)	312 ms (5,4 %)
acme.features.flightcrewmember.flightassignment.FlightAssignmentPublishService. <b>validate</b> ()	0,0 ms (- %)	304 ms (5,3 %)
acme.features.flightcrewmember.flightassignment.FlightAssignmentsCreateService. <b>validate</b> ()	0,0 ms (- %)	299 ms (5,2 %)
acme.features.flightcrewmember.activitylog.ActivityLogCreateService. <b>authorise</b> ()	0,0 ms (- %)	200 ms (3,5 %)
acme.features.flightcrewmember.flightassignment.FlightAssignmentsCreateService. <b>bind</b> ()	0,0 ms (- %)	198 ms (3,4 %)
acme.features.flightcrewmember.activitylog.ActivityLogShowService. <b>authorise</b> ()	0,0 ms (- %)	197 ms (3,4 %)
acme.features.flightcrewmember.flightassignment.FlightAssignmentsLegLandedListService. <b>unbind</b> ()	0,0 ms (- %)	102 ms (1,8 %)
acme.features.flightcrewmember.activitylog.ActivityLogCreateService. <b>unbind</b> ()	0,0 ms (- %)	102 ms (1,8 %)
acme.features.flightcrewmember.flightassignment.FlightAssignmentPublishService. <b>bind</b> ()	0,0 ms (- %)	101 ms (1,7 %)
acme.features.flightcrewmember.activitylog.ActivityLogShowService. <b>unbind</b> ()	0,0 ms (- %)	101 ms (1,7 %)
acme.features.flightcrewmember.flightassignment.FlightAssignmentUpdateService. <b>authorise</b> ()	0,0 ms (- %)	100 ms (1,7 %)

Con índices:

Name	Self Time (CPU)	Total Time (CPU)
acme.features.flightcrewmember.flightassignment.FlightAssignmentUpdateService. <b>bind</b> ()	0,0 ms (- %)	418 ms (14,4 %)
acme.features.flightcrewmember.flightassignment.FlightAssignmentUpdateService. <b>unbind</b> ()	0,0 ms (- %)	400 ms (13,8 %)
acme.features.flightcrewmember.flightassignment.FlightAssignmentUpdateService. <b>authorise</b> ()	0,0 ms (- %)	313 ms (10,8 %)
acme.features.flightcrewmember.flightassignment.FlightAssignmentsCreateService. <b>unbind</b> ()	0,0 ms (- %)	298 ms (10,3 %)
acme.features.flightcrewmember.flightassignment.FlightAssignmentPublishService. <b>validate</b> ()	0,0 ms (- %)	298 ms (10,3 %)
acme.features.flightcrewmember.flightassignment.FlightAssignmentShowService. <b>unbind</b> ()	0,0 ms (- %)	205 ms (7,1 %)
acme.features.flightcrewmember.flightassignment.FlightAssignmentsCreateService. <b>validate</b> ()	0,0 ms (- %)	199 ms (6,9 %)
acme.features.flightcrewmember.flightassignment.FlightAssignmentsCreateService. <b>authorise</b> ()	0,0 ms (- %)	197 ms (6,8 %)
acme.features.flightcrewmember.flightassignment.FlightAssignmentsCreateService. <b>bind</b> ()	0,0 ms (- %)	196 ms (6,8 %)
acme.features.flightcrewmember.flightassignment.FlightAssignmentUpdateService. <b>validate</b> ()	0,0 ms (- %)	98,9 ms (3,4 %)
acme.features.flightcrewmember.flightassignment.FlightAssignmentPublishService. <b>bind</b> ()	0,0 ms (- %)	93,7 ms (3,2 %)
acme.features.flightcrewmember.activitylog.ActivityLogCreateService. <b>authorise</b> ()	0,0 ms (- %)	93,2 ms (3,2 %)
acme.features.flightcrewmember.flightassignment.FlightAssignmentPublishService. <b>unbind</b> ()	0,0 ms (- %)	90,3 ms (3,1 %)
acme.features.flightcrewmember.activitylog.ActivityLogDeleteService. <b>authorise</b> ()	0,0 ms (- %)	0,0 ms (0 %)
acme.features.flightcrewmember.activitylog.ActivityLogUpdateService. <b>bind</b> ()	0,0 ms (- %)	0,0 ms (0 %)
acme.features.flightcrewmember.flightassignment.FlightAssignmentPublishService. <b>authorise</b> ()	0,0 ms (- %)	0,0 ms (0 %)



## 6. Hardware Profiling

### 1. Current Disk Queue Length (verde)

Durante la ejecución, se observaron pequeños picos esporádicos, lo que indica que el acceso a disco no representa un cuello de botella crítico, aunque sí existen momentos puntuales de mayor carga.

### 2. Bytes Total/sec (azul)

Al igual que con la cola de disco, se detectaron picos ocasionales, generalmente coincidiendo con operaciones más complejas como la actualización o publicación de asignaciones. Estos picos son normales y no representan una degradación del sistema.

### 3. % Committed Bytes In Use (rosa)

La línea se mantiene estable alrededor del 90%, lo que sugiere que el sistema mantiene una carga de memoria constante, sin fugas ni consumo excesivo. Este comportamiento es positivo, ya que refleja estabilidad en el uso de memoria RAM.

### 4. % Processor Time (rojo)

Representa el **porcentaje del tiempo que el procesador está ocupado ejecutando procesos activos**.

Este fue el contador más inestable, oscilando entre 20% y 40%, lo cual es esperable durante operaciones intensivas. Aunque se observan variaciones, los valores se mantienen dentro de un rango razonable y no indican saturación.

Conclusión.

La memoria RAM se mantiene constante en torno al 90% de uso, sin sobrecargas.

El procesador experimenta oscilaciones normales durante operaciones pesadas, pero sin alcanzar niveles críticos.

Los accesos a disco y el flujo de datos presentan picos puntuales, pero no sostenidos, lo que indica que el sistema responde adecuadamente a los momentos de mayor actividad.

