INFORME DE TESTING INDIVIDUAL

Grupo 1-C1.020 | https://github.com/PDJ6975/Acme-ANS-D04-25.5.0

Nombre	Correo Corporativo
Antonio Rodríguez Calderón	antrodcal@alum.us.es
Adrián Ramírez Gil	adrramgil@alum.us.es
Jianwu Hu	jiahu3@alum.us.es
Pablo Castrillón Mora	pabcasmor1@alum.us.es
Pablo Olivencia Moreno	pabolimor@alum.us.es

26 DE MAYO DE 2025

DISEÑO Y PRUEBAS II

Tabla de Contenido

1. Testing Funcional

1.1 Introducción

Esta sección del informe tiene como objetivo presentar un listado de los casos de prueba llevados a cabo, organizados según las características específicas de las entidades *Flight* y *Leg*.

Para cada caso de prueba, se dará una descripción y se indicará cuál es su eficacia a la hora de detectar errores.

1.2 Metodología

Para verificar el correcto funcionamiento de las entidades requeridas por el primer estudiante, se han llevado a cabo pruebas '.safe' y '.hack'. Las pruebas '.safe' han evaluado que las entidades puedan listarse, visualizarse y manejarse mediante operaciones CRUD válidas y operativas, aceptando únicamente valores correctos y rechazando aquellos que no superan las validaciones establecidas.

En el caso de las pruebas '.hack', se realizaron pruebas de tipo GET/POST hacking con el objetivo de verificar que el sistema responde adecuadamente frente a intentos de acciones no autorizadas o maliciosas.

Por último, se adjunta una tabla con la cobertura alcanzada tanto para el paquete de Flight como para el paquete de Leg:

Paquete	**Cobertura alcanzada**
manager.flight	96,3%
manager.leg	96,4%

Adentrándonos en la parte específica de Flight:

Servicio	**Cobertura alcanzada**
ManagerFlightCreateService	85,5%
ManagerFlightDeleteService	96%
ManagerFlightListService	100%
ManagerFlightPublishService	100%
ManagerFlightShowService	100%
ManagerFlightUpdateService	99%

En cuanto a Leg:

Servicio	**Cobertura alcanzada**
ManagerFlightCreateService	98,2%
ManagerFlightDeleteService	100%
ManagerFlightListService	99%
ManagerFlightPublishService	100%
ManagerFlightShowService	100%
ManagerFlightUpdateService	90%

Para los controladores de cada una de las entidades mencionadas anteriormente, se alcanza un 100% de cobertura, lo que significa que se prueban todas las líneas que los componen.

1.3 Casos de prueba por características

1.3.1 Flight

Caso de prueba	Tipo	Descripción	Eficacia
list.safe	Renderizado	Comprobar que el listado funciona correctamente	No se detectaron incidencias
list.hack	Hack	Se comprueba que miembros con otro rol distinto al de manager no puedan ver el listado	No se detectaron incidencias
show.safe	Renderizado	Se comprueba que se muestre correctamente el formulario de vuelo	No se detecta ninguna incidencia
show.hack	Hack	Se comprueba que únicamente pueda acceder manager al que	No se detecta ninguna incidencia, debido al correcto

		está asociado el vuelo	funcionamiento del validate()
create.safe	Casos positivos y negativos del formulario	Se comprueba tanto que se admiten los valores permitidos como que se rechacen los valores no permitidos en cada atributo	No se detecta ninguna incidencia
create.hack	Hack	Se comprueba que miembros con otro rol no puedan acceder al formulario, por cada desplegable se comprueba su correcto comportamiento ante intentos de hackeo y por último también se comprueban los atributos readOnly de manera que los valores introducidos desde la consola sean ignorados.	Se detecto una incidencia en un atributo readOnly que fue resuelta en el bind() del servicio correspondiente
update.safe	Casos de prueba positivos y negativos del formulario	Se comprueba lo mismo que en create.safe pero esta vez con la funcionalidad de actualización	Al igual que en el create.safe, se resolvió un conflicto relacionado con un atributo readOnly.
update.hack	Hack	También relacionado directamente con el create.hack, ya que se	Al igual que en los 3 anteriores, se corrige el atributo readOnly.

		comprueban los mismos casos.	
publish.safe	Casos positivos	Se comprueba el correcto funcionamiento de la funcionalidad. En este caso no se comprueban casos de prueba negativos ya que el botón de publicar no está disponible desde la interfaz si el Flight no está disponible para publicarse.	No se detectaron incidencias.
publish.hack	Hack	Se comprobó que solo pudiera acceder al recurso el manager al cual está asociado el Flight que se quiere publicar.	No se detectó ninguna incidencia.
delete.safe	Casos positivos	Exactamente igual al publish.safe	No se detectaron incidencias.
delete.hack	Hack	Exactamente igual al publish.hack	No se detectaron incidencias.

1.3.2 Leg

Caso de prueba	Tipo	Descripción	Eficacia
list.safe	Renderizado	Comprobar que el listado funciona correctamente	No se detectaron incidencias
list.hack	Hack	Se comprueba que miembros con otro rol distinto al de	No se detectaron incidencias

		manager no puedan ver el listado	
show.safe	Renderizado	Se comprueba que se muestre correctamente el formulario de leg	No se detecta ninguna incidencia
show.hack	Hack	Se comprueba que únicamente pueda acceder manager al que está asociado el leg	No se detecta ninguna incidencia, debido al correcto funcionamiento del validate()
create.safe	Casos positivos y negativos del formulario	Se comprueba tanto que se admiten los valores permitidos como que se rechacen los valores no permitidos en cada atributo	Se detectaron algunas incidencias en las validaciones que fueron resueltas en el servicio correspondiente.
create.hack	Hack	Se comprueba que miembros con otro rol no puedan acceder al formulario, por cada desplegable se comprueba su correcto comportamiento ante intentos de hackeo y por último también se comprueban los atributos readOnly de manera que los valores introducidos desde la consola sean ignorados.	No se detectaron incidencias.

update.safe	Casos de prueba positivos y negativos del formulario	Se comprueba lo mismo que en create.safe pero esta vez con la funcionalidad de actualización	Al igual que en el create.safe, se detectaron algunos fallos de validación que fueron resueltos.
update.hack	Hack	También relacionado directamente con el create.hack, ya que se comprueban los mismos casos.	No se detectaron incidencias.
publish.safe	Casos positivos	Se comprueba el correcto funcionamiento de la funcionalidad. En este caso no se comprueban casos de prueba negativos ya que el botón de publicar no está disponible desde la interfaz si el Flight no está disponible para publicarse.	No se detectaron incidencias.
publish.hack	Hack	Se comprobó que solo pudiera acceder al recurso el manager al cual está asociado el Flight que se quiere publicar.	No se detectó ninguna incidencia.
delete.safe	Casos positivos	Exactamente igual al publish.safe	No se detectaron incidencias.
delete.hack	Hack	Exactamente igual al publish.hack	No se detectaron incidencias.

1.4 Calidad de la cobertura

En este apartado se analizará la calidad de la cobertura del código de los servicios tanto de Flight como de Leg.

1.4.1 Flight

La cobertura obtenida para el paquete manager.flight es del 96,3%, lo que evidencia una implementación robusta y bien verificada a través de la suite de pruebas. En particular los servicios de ManagerFlightListService, ManagerFlightPublishService y ManagerFlightShowService alcanzan una cobertura del 100% asegurando una validación completa de sus funcionalidades.

En cuanto a los servicios de crear, actualizar, eliminar, mostrar y publicar, los porcentajes de cobertura varían entre el 85,5% y el 100%, destacando ManagerFlightUpdateService con un 99% y ManagerFlightDeleteService con un 96%. Por otro lado, el servicio ManagerFlightCreateService presenta una cobertura del 85,5%, lo que sugiere la posible existencia de rutas no cubiertas dentro del flujo de ejecución.

Al igual que en otros módulos, se han implementado validaciones adicionales en método como authorise y validate, esto hace que, en muchos casos, no puedan ser evaluadas muchas de las funcionalidades directamente a través de la interfaz.

1.4.2 Leg

La cobertura alcanzada para el paquete manager.leg es del 96,4%, lo que refleja la alta calidad en la implementación y verificación de los casos de prueba.

Destacan especialmente los servicios ManagerLegDeleteService, ManagerLegPublishService y ManagerLegShowService, todos ellos con una cobertura del 100%, lo que garantiza una validación completa de sus respectivas funcionalidades.

En lo que respecta a los servicios de crear, actualizar, eliminar, mostrar y publicar, los niveles de cobertura oscilan entre el 90% y el 100%. Sobresale el

ManagerLegCreateService con un 98,2%, seguido por ManagerLegListService con un 99% y ManagerLegUpdateService con un 90%.

Como en los demás módulos, se han incorporado mecanismos de validación adicionales con métodos como el authorise y el validate, lo que hace que muchas de las funcionalidades no se puedan probar directamente desde la interfaz de usuario, al igual que pasa con la entidad Flight.

1.5 Conclusiones finales

En resumen, se ha implementado un conjunto de pruebas que abarca la mayor parte del código correspondiente a las entidades asignadas al estudiante uno. Independientemente de que siempre se pueden mejorar las validaciones, se ha logrado construir un sistema sólido y seguro, capaz de responder eficazmente ante la mayoría de las situaciones.

2. Testing de Rendimiento

2.1 Entorno y protocolo

Para el desarrollo de las pruebas se empleó un equipo con las siguientes características:

Modelo	RAM	CPU	GPU	Disco	SO
Lenovo Ideapad S540	20 GB	Intel Core i7-8565U CPU @ 1.80GHz	NVIDIA GeForce GTX 1650	500 GB	Windows 11 Home

Este estudio se basará exclusivamente en los casos de prueba que he ejecutado personalmente como estudiante uno, utilizando la versión 25.5.0 tanto del proyecto como del framework. En la primera ejecución, mis entidades no contarán con índices, con el fin de evaluar si el rendimiento mejora tras su implementación.

2.2 Recogida de datos

La recopilación de datos, tanto previa como posterior a la implementación de índices, se ha realizado siguiendo los procedimientos descritos en la teoría. Para ello, se ha analizado el archivo '.trace' generado por la aplicación, aplicando un proceso de filtrado que permitió obtener un fichero depurado, del cual se extrajo información relevante como los gráficos.

2.3 Estadísticas Descriptivas

Tras filtrar las peticiones generadas durante la ejecución de los casos de prueba, se elaboró una tabla con estadísticas descriptivas de los tiempos de respuesta. Tanto en la traza original como en la obtenida después de añadir los índices, se aprecia una media cercana a 28,4 ms y una desviación estándar cercana a los 42,6 ms. Los valores desglosados según la traza son los siguientes:

2.3.1 Traza anterior a los índices

Antes	
Media	28,46253442
Error típico	1,79937175
Mediana	7,7519
Moda	#N/D
Desviación estándar	42,61890905
Varianza de la muestra	1816,371408
Curtosis	3,208701153
Coeficiente de asimetría	1,973026289
Rango	254,1781
Mínimo	1,2743
Máximo	255,4524
Suma	15967,48181
Cuenta	561
Nivel de confianza(95,0%)	3,534342543

Interval(ms)	24,92819188	31,99687696
Interval(s)	0,024928192	0,031996877

2.3.2 Traza posterior a los índices

Después	
Media	28,42094991
Error típico	1,79965691
Mediana	7,4423
Moda	1,5592
Desviación estándar	42,62566317
Varianza de la muestra	1816,947161
Curtosis	2,502838885
Coeficiente de asimetría	1,892871404
Rango	184,9409
Mínimo	1,2654
Máximo	186,2063
Suma	15944,1529
Cuenta	561
Nivel de confianza(95,0%)	3,534902655

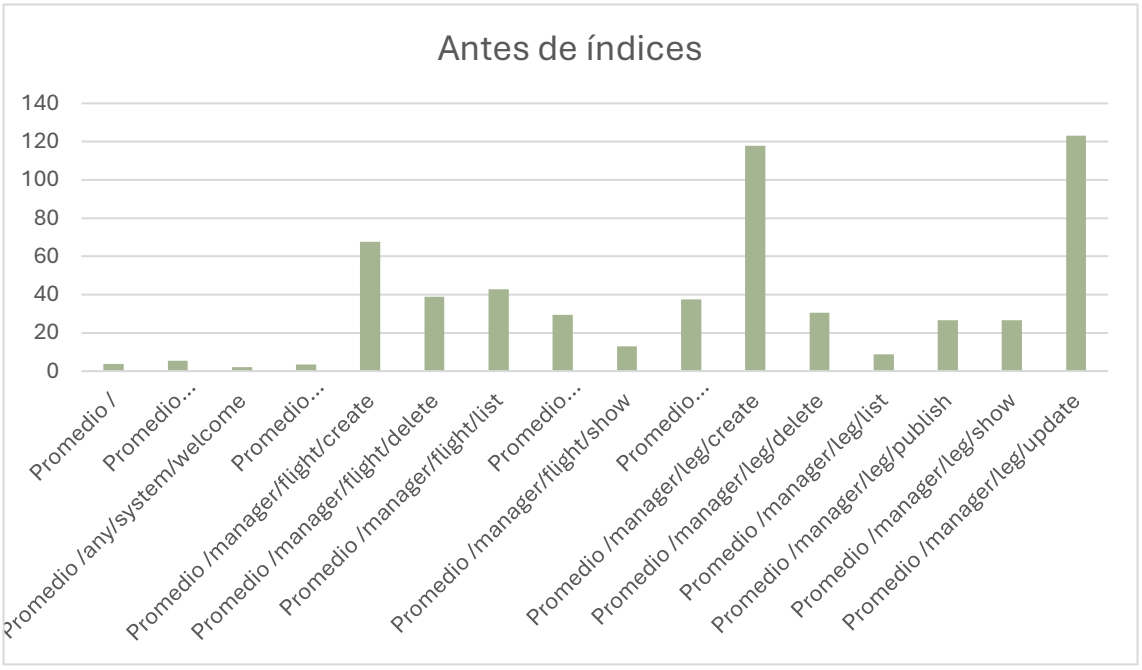
Interval (ms)	24,88604726	31,95585257
Interval (s)	0,024886047	0,031955853

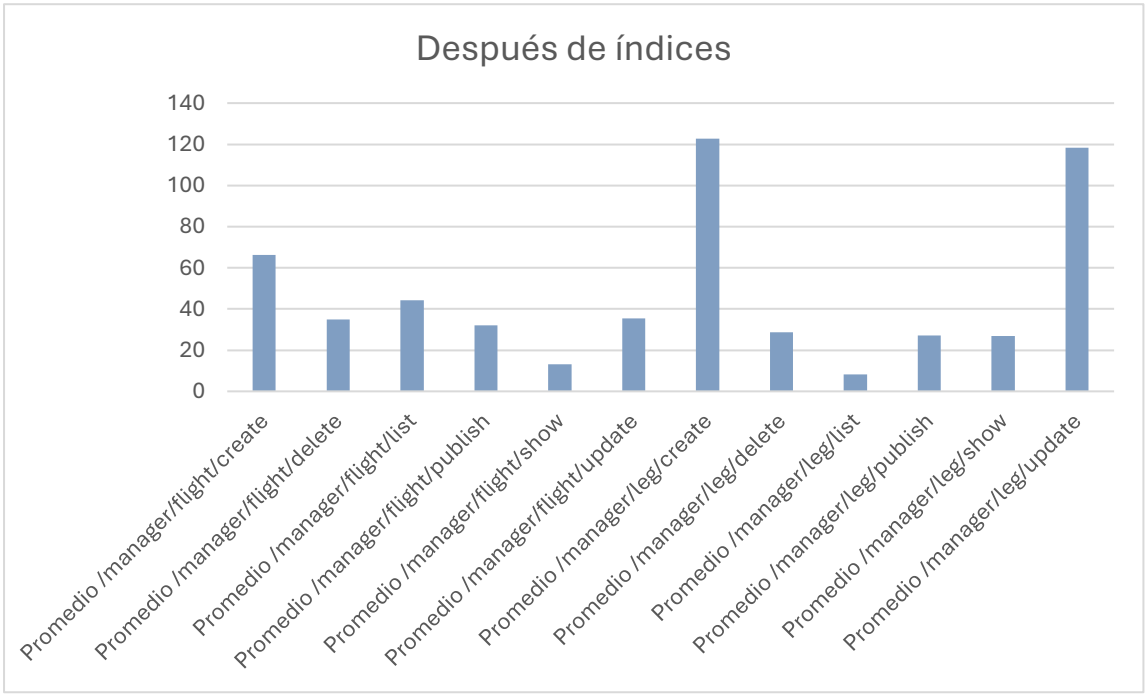
2.3.3 Comparación entre ambas trazas

Comparando las estadísticas de ambas trazas, se aprecia que los tiempos de respuesta son bastante bajos en ambos escenarios, con intervalos de confianza que se superponen por completo, lo cual indica que no existen diferencias estadísticamente significativas entre ambos conjuntos de datos.

2.4 Gráficos de eficiencia por características

Tras generar los gráficos correspondientes tanto a antes de los índices como a después de estos, se observa que el elemento común en ambos de ellos es que las peticiones más costosas en cuanto a tiempo son manager/leg/create y manager/leg/update, ya que requiere cargar datos relacionados con varias entidades diferentes para poder cumplimentar los atributos de la entidad Leg.





Aunque algunas de las funcionalidades muestran ligeras variaciones, en términos generales no se observan mejoras significativas tras la incorporación de los índices. Esto se debe a que los tiempos de respuesta iniciales ya eran bajos, lo que hace más difícil poder conseguir una mejora notable con la implementación de una mejora en el código como la de los índices.

Por tanto, aunque no era estrictamente necesario analizar el MIR, dado que el límite superior de los intervalos de confianza se encontraba muy por debajo del umbral exigido de 1s, este análisis permitió afinar un poco más la comparación entre ambas trazas.

2.5 Hipótesis y conclusión

Con el fin de determinar si las diferencias observadas tienen relevancia estadística, se ha llevado a cabo una prueba Z para dos muestras independientes, asumiendo varianzas conocidas.

Prueba z para medias de dos muestras		
	Before	*After*
Media	25,34907525	25,3101405
Varianza (conocida)	1675894592	167744892
Observaciones	642	642
Diferencia hipotética de las medias	0	
z	2,29756E-05	
P(Z<=z) una cola	0,499990834	
P(Z<=z) dos colas	0,999981668	
Valor crítico de z (una cola)	1,644853627	
Valor crítico de z (dos colas)	0,999981668	
Valor crítico de z (dos colas)	1,959963985	

Como se puede observar, el valor de P(Z<=z) dos colas, es mucho mayor que el valor α sugerido en las diapositivas de teoría, lo que indica claramente que los cambios que se hicieron entre una traza y otra (en este caso la implementación de los índices) no resultaron realmente en mejoras significativas.

3. Tabla de revisión

Versión	Fecha	Descripción de los cambios
1.0	26/05/2025	Primera versión del documento