Testing Report Student #1
C1.019
Adolfo Borrego González

Grupo	C1.019
Repositorio	https://github.com/adolfoborrego/Acme-ANS
Student #1	**ID**: 29584665H **UVUS**: XXB5458 **Nombre**: Borrego González, Adolfo Agustín **Roles**: Project Manager
Student #2	**ID**: 77873179D **UVUS**: SSK0456 **Nombre**: Martínez Díaz, Ignacio **Roles**: Analyst
Student #3	**ID**: 12830191D **UVUS**: PVL1690 **Nombre**: Mir Ceballos, Miguel **Roles**: Operator
Student #4	**ID**: 52077055H **UVUS**: TCP2748 **Nombre**: Sánchez Carmona, Germán **Roles**: Developer
Student #5	**ID**: 54794337B **UVUS**: CFV7375 **Nombre**: Regidor García, Miguel **Roles**: Tester

Índice

1. Introducción

1.1. Propósito del documento

El propósito de este documento es presentar de forma estructurada y rigurosa los resultados obtenidos durante el proceso de pruebas del sistema desarrollado por el equipo. El informe tiene como objetivo principal verificar que las funcionalidades implementadas cumplen correctamente con los requisitos definidos y que el sistema ofrece un comportamiento estable, seguro y eficiente tanto a nivel funcional como de rendimiento.

A través de este documento se detalla la ejecución de pruebas funcionales, diferenciando entre versiones inseguras (.hack) y versiones protegidas (.safe), así como el análisis del rendimiento del sistema en diferentes condiciones, incluyendo comparativas estadísticas e intervalos de confianza. Este análisis permite no solo identificar errores o vulnerabilidades, sino también justificar decisiones técnicas tomadas durante el desarrollo para mejorar la calidad final del producto.

2. Functional Testing

2.1. Introducción

En esta sección se recogen las pruebas funcionales realizadas sobre las principales funcionalidades del sistema. Para cada una de ellas se han preparado dos versiones: una insegura (**.hack**) que permite detectar errores o fallos de validación, y otra protegida (**.safe**) que incorpora medidas de seguridad.

El objetivo de estas pruebas es comprobar que todo funciona como se espera, tanto en condiciones normales como en casos maliciosos. Las funcionalidades probadas pertenecen a los módulos **flight** y **leg**, y se han cubierto operaciones como crear, borrar, listar, publicar, mostrar y actualizar. Gracias a este enfoque hemos podido comparar fácilmente los errores que aparecen en la versión insegura y confirmar que están solucionados en la versión segura.

2.2. Cobertura de las pruebas

texto

Paquete	Cobertura
airlineManager.flight	96,2%
airlineManager.leg	95,8%

texto

Servicio	Cobertura
AirlineManagerFlightListService	95,3%
AirlineManagerFlightShowService	97,7%
AirlineManagerFlightCreateService	91,0%
AirlineManagerFlightUpdateService	93,7%
AirlineManagerFlightDeleteService	100,0%
AirlineManagerFlightPublishService	98,7%

texto

Servicio	Cobertura
AirlineManagerLegListService	97,7%
AirlineManagerLegShowService	92,3%
AirlineManagerLegCreateService	94,8%
AirlineManagerLegUpdateService	95,5%
AirlineManagerLegDeleteService	98,1%
AirlineManagerLegPublishService	98,1%

2.3. Casos de prueba

A continuación se muestran los casos de prueba realizados sobre las funcionalidades del módulo flight, tanto en su versión insegura (**.hack**) como en la segura (**.safe**). El objetivo es comprobar que cada operación responde correctamente y detectar posibles errores o vulnerabilidades.

Para Flight:

Caso de prueba	Descripción	Eficacia
list.safe	Para comprobar el .safe del list solo entré a la pestaña de list flight y me moví sobre las distintas pestañas para generar varias peticiones GET y obtener buena cobertura.	No hubo problema de nada ya que solo tenía que hacer una petición GET.
list.hack	Accedí con la URL de list flights desde otros Realm para comprobar que no podían acceder.	Ningún problema ni bug encontrado.
show.safe	Acceder a distintos flight para poder ver todos los valores comprobando flights que estuviesen publish y que no lo estuviese.	Ningún problema ni bug encontrado.
show.hack	En este caso solo pude probar acceder desde distintos realms y desde manager distintos para ver que no tenía acceso a los Flights.	
create.safe	Entró a la parte de crear y compruebo con datos vacíos, los límites, datos que no se cumplen y un dato real.	Ningún problema ni bug encontrado.
create.hack	Intento crear un Flight que no tenga nada, comprobando sus validaciones accediendo de distinto Realm y probando datos que no pueden ser.	Ningún problema ni bug encontrado.
update.safe	Hacemos lo mismo que en el .safe del create probando destinos casos con sus límites, dejando huecos en blanco y haciendo un update correcto.	Ningún problema ni bug encontrado.
update.hack	Probamos hackearlo intentando acceder desde otro Realm distinto e intento cambiar datos que no pueden estar para ver si se actualizan.	Ningún problema ni bug encontrado.
delete.safe	Solo probamos borrar colocando la id arriba para poder hacer un get y que la cobertura cubriese también el unbind.	Ningún problema ni bug encontrado.
delete.hack	probamos intentar borrar desde otro	Ningún problema ni bug encontrado.

	usuario de manager y desde otro realm intentado meter la id y haciendo POST hacking.	
publish.safe	Accedimos a un flight que cumpliese los requisitos y las validaciones hacemos el GET en la URL para que el unbind funcione y publicamos un vuelo.	Ningún problema ni bug encontrado.
publish.hack	Intentamos acceder desde otros usuarios y otros manager para intentar publicar el vuelo, también intentamos hacerlo quitando los valores de readonly para probarlo.	Ningún problema ni bug encontrado.

En esta tabla se recogen los casos de prueba aplicados al módulo leg, diferenciando también entre las versiones **.hack** y **.safe**. Se ha validado que cada funcionalidad se comporte como se espera y que los errores detectados en la versión insegura no estén presentes en la segura.

Para Leg:

Caso de prueba	Descripción	Eficacia
list.safe	Se accedió a la pestaña correspondiente y se navegaron distintas páginas del listado, generando múltiples peticiones GET para asegurar buena cobertura.	Ningún problema ni bug encontrado.
list.hack	Se accedió directamente con la URL de list leg desde otros Realm y desde otros manager ya que necesitamos el id del flight, para validar que no se podía acceder a los datos sin permisos adecuados.	Ningún problema ni bug encontrado.
show.safe	Se accedió a distintos legs ya creados para comprobar que solo se mostraban correctamente aquellos que estuviesen publicados y los que no, para obtener la mayor cobertura, y que el acceso fuera correcto.	Ningún problema ni bug encontrado.
show.hack	Se intentó acceder desde distintos Realms y distintos manager a legs ajenos para comprobar que el sistema no permitía el acceso sin permisos.	Ningún problema ni bug encontrado.
create.safe	Se probó la creación de legs con diferentes combinaciones de datos, incluyendo campos vacíos, valores inválidos, y luego un caso válido.	Ningún problema ni bug encontrado.
create.hack	Se intentó crear un leg con datos incoherentes como fechas erróneas, airports inexistentes o desde otro Realm para validar las restricciones.	Se encontró problema con los aircraft cuando se metia un valor de un aircraft que estaba en MAINTENANCE, no se puede pero se podía.
update.safe	Se realizaron pruebas sobre la edición de un leg modificando campos válidos e inválidos, incluyendo límites de longitud y campos en blanco.	Ningún problema ni bug encontrado.
update.hack	Se intentó actualizar un leg desde otro Realm o con datos que no deberían permitirse (por ejemplo, vuelos no publicados), para ver si se aplicaban los cambios.	Se encontró problema con los aircraft cuando se metia un valor de un aircraft que estaba en MAINTENANCE, no se puede pero se podía.

delete.safe	Se accedió a la funcionalidad de eliminar leg desde el entorno autorizado, verificando que la acción sólo era posible si no había restricciones que lo impidieran.	Ningún problema ni bug encontrado.
delete.hack	Se probó eliminar un leg desde otro Realm o con usuarios no autorizados manipulando el ID directamente por URL.	Ningún problema ni bug encontrado.
publish.safe	Se accedió a un leg válido y se realizó la publicación tras verificar que cumplía los requisitos establecidos. Se comprobó también la ejecución del unbind correctamente.	Ningún problema ni bug encontrado.
publish.hack	Se intentó publicar un leg desde cuentas o roles no autorizados, manipulando el HTML y eliminando restricciones visibles como campos readonly.	Ningún problema ni bug encontrado.

2.4. Conclusiones

Tras realizar las pruebas funcionales sobre los módulos flight y leg, tanto en sus versiones inseguras (**.hack**) como en las protegidas (**.safe**), podemos concluir que el sistema se comporta de forma correcta en la mayoría de los casos. La gran mayoría de las funcionalidades han respondido según lo esperado, sin errores ni comportamientos extraños, lo que demuestra una implementación sólida y estable.

En el caso de las versiones .safe, no se detectaron fallos relevantes, lo que confirma que las medidas de seguridad y validación están correctamente aplicadas. Las pruebas realizadas han abarcado tanto entradas válidas como maliciosas o inválidas, y el sistema supo gestionarlas correctamente.

Sin embargo, se detectó un fallo importante tanto en flight.create.hack como en leg.create.hack y sus respectivas versiones de update, donde fue posible asignar un aircraft que estaba en estado MAINTENANCE. Esta situación no debería ser posible, ya que violaba las reglas del dominio del sistema. Este error también se reprodujo en algunos casos de prueba safe, por lo que deberá corregirse en el código.

En resumen, las pruebas han servido para confirmar la funcionalidad correcta del sistema en la mayoría de los escenarios, identificar un caso concreto que necesita corrección, y validar que la protección frente a acciones no autorizadas funciona adecuadamente.

3. Performance Testing

3.1. Introducción

En esta sección se analiza el rendimiento del sistema a partir de los tiempos de respuesta obtenidos al ejecutar las funcionalidades principales. Se han realizado pruebas en dos situaciones distintas: en mi portátil y en mi ordenador sobremesa, para comprobar cómo afecta esta optimización al tiempo que tarda el sistema en procesar las peticiones.

Los datos se han recogido de forma automática y se han utilizado para calcular medias, intervalos de confianza al 95% y contrastes de hipótesis. Esto nos permite comparar los resultados entre las dos versiones y saber con datos reales cuál tiene mejor rendimiento.

3.2. Gráficos de eficiencia medios

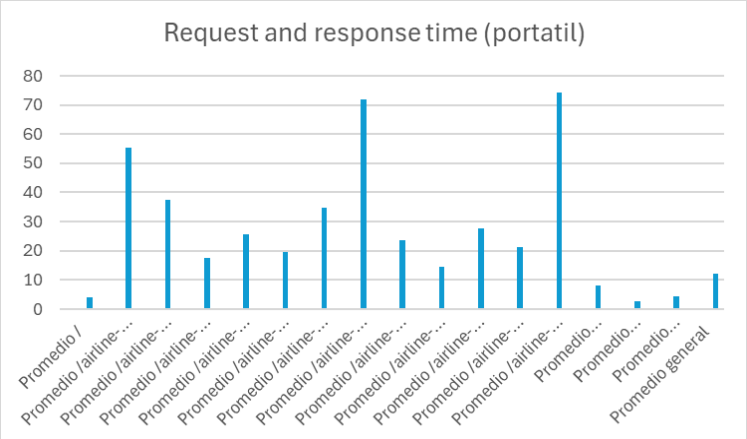
Para analizar el rendimiento del sistema, se midieron los tiempos de respuesta de las funcionalidades principales en dos equipos distintos: un portátil (primer gráfico) y un ordenador de sobremesa (segundo gráfico). Las gráficas reflejan los tiempos promedio por cada ruta del sistema, permitiendo identificar qué operaciones son más costosas en términos de tiempo.

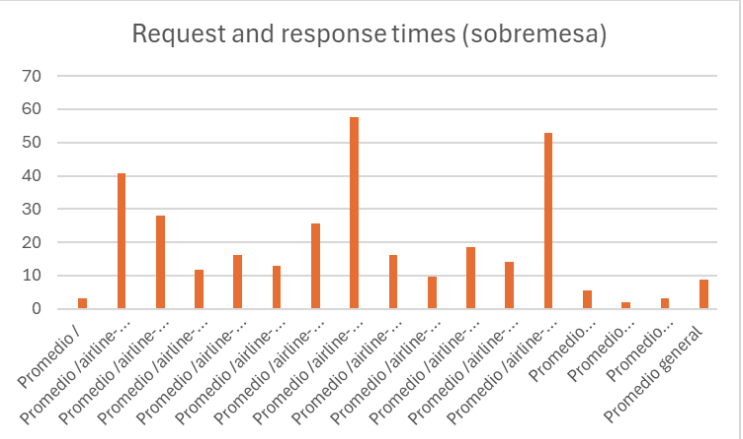
En el gráfico del portátil, se observan picos claros en operaciones como /flight/create, /flight/update y /leg/update, superando los 70 ms en algunos casos. Esto sugiere que estas funcionalidades implican procesos más complejos o mayor interacción con la base de datos.

Por otro lado, en el gráfico del ordenador de sobremesa, aunque se repiten los mismos puntos críticos, los valores son consistentemente más bajos. Por ejemplo, /flight/create ronda los 55 ms, y /leg/update los 52 ms, lo que supone una mejora notable respecto al portátil.

En ambos casos, las operaciones como /, /sign-in, /welcome o /sign-out presentan los tiempos más bajos, lo cual es esperable al ser funciones más simples.

El promedio general en el portátil fue de **12,16 ms**, mientras que en el sobremesa fue de **8,65 ms**, confirmando que el equipo de sobremesa tiene mejor rendimiento y tiempos de respuesta más estables. Esta comparación evidencia la influencia del hardware en la eficiencia del sistema y permite priorizar qué rutas podrían necesitar más atención a nivel de optimización.





3.3. Estadísticas descriptivas

Para comparar el rendimiento del sistema entre el portátil y el ordenador de sobremesa, se han calculado una serie de estadísticas descriptivas a partir de los tiempos de respuesta recogidos.

La **media** en el portátil fue de **14,22 ms**, mientras que en el sobremesa fue de **10,09 ms**, lo que confirma que el equipo de sobremesa ofrece un mejor rendimiento en general. También se observa que la mediana en ambos casos es inferior a la media (8,46 ms en portátil y 5,63 ms en sobremesa), lo cual indica que hay valores extremos altos que aumentan la media, algo confirmado por la alta asimetría (3,88 en portátil y 4,65 en sobremesa).

La desviación estándar también es más alta en el portátil (21,73 ms) que en el sobremesa (16,70 ms), lo que significa que los tiempos de respuesta son más variables y menos predecibles en el portátil.

En cuanto al intervalo de confianza al 95% de la media, este es más estrecho en el sobremesa (**[8,71 ms – 11,47 ms]**) que en el portátil (**[12,43 ms – 16,02 ms]**), lo que refuerza la conclusión de que el sobremesa ofrece un rendimiento más constante y fiable.

Estos resultados muestran diferencias claras en el comportamiento del sistema según el hardware utilizado, y sirven como base para realizar un análisis más profundo mediante contrastes de hipótesis.

Portatil		
Media	14,22115821	
Error típico	0,913338	
Mediana	8,458570345	
Moda	#N/D	
Desviación estándar	21,72900014	
Varianza de la muestra	472,1494472	
Curtosis	17,85496692	
Coeficiente de asimetría	3,885411013	
Rango	154,1446	
Mínimo	1,6502	
Máximo	155,7948	
Suma	8049,175548	
Cuenta	566	
Nivel de confianza(95,0%)	1,793952516	
Interval (ms)	12,4272057	16,0151117
Interval (ms)	0,012427206	0,01601511

Ordenador Sobremesa		
Media	10,0935238	
Error típico	0,70203416	
Mediana	5,63605	
Moda	7,6661	
Desviación estándar	16,7019224	
Varianza de la muestra	278,954213	
Curtosis	26,84482	
Coeficiente de asimetría	4,65267371	
Rango	158,0037	
Mínimo	1,1411	
Máximo	159,1448	
Suma	5712,93446	
Cuenta	566	
Nivel de confianza(95,0%)	1,37891553	
Interval (ms)	8,71460825	11,4724393
Interval (s)	0,00871461	0,01147244

3.4. Hipótesis y conclusiones

Para confirmar si las diferencias en los tiempos de respuesta entre las dos configuraciones probadas (con y sin índices) son significativas, se ha realizado una prueba z para la comparación de medias de dos muestras independientes. Los resultados muestran una mejora clara en los tiempos tras aplicar los índices en la base de datos.

La **media** de respuesta antes de aplicar los índices fue de **12,16 ms**, mientras que después de aplicarlos fue de **8,65 ms**. Esta diferencia se acompaña de una menor **varianza** (472 antes frente a 278 después), lo que indica que el sistema se comporta de forma más estable y predecible con índices.

El valor de **z calculado fue 3,295**, y el valor **p (dos colas) fue de 0,00098**, claramente por debajo del umbral de significación del 0,05. Esto significa que podemos rechazar la hipótesis nula y concluir que la mejora es estadísticamente significativa. Es decir, usar índices mejora notablemente el rendimiento.

Además, al comparar los resultados entre el **portátil con y sin índices**, también se observa una mejora clara, aunque menos marcada que en el sobremesa. Esto se debe a que las consultas que se realizan no son especialmente complejas, **por lo que la base de datos puede optimizarlas bien cuando hay índices**, incluso en equipos con menor capacidad.

En resumen, los datos confirman que aplicar índices no solo mejora el tiempo de respuesta en general, sino que también permite que el sistema funcione mejor en equipos menos potentes, como el portátil. Además, se observa que el ordenador de sobremesa, gracias a su mejor rendimiento de hardware, consigue los tiempos más bajos y estables.

Prueba z para medias de dos muestras		
	Before	*After*
Media	12,1634892	8,65859864
Varianza (conocida)	472,149447	278,954213
Observaciones	664	664
Diferencia hipotética de las medias	0	
z	3,29540318	
P(Z<=z) una cola	0,0004914	
Valor crítico de z (una cola)	1,64485363	
P(Z<=z) dos colas	0,00098281	
Valor crítico de z (dos colas)	1,95996398	

4. Historial de versiones

Versión	Fecha	Descripción
0.0	25/05/2025	Borrador inicial
1.0	26/05/2025	Correcciones de functional y performance

5. Bibliografía

Intentionally Blank