**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 pruebas**

**Grupo**: C2.029

**Número del estudiante**: 1

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

**Fecha**: 03/07/2025

|  |  |
| --- | --- |
| **Apellidos, Nombre** | **Correo** |
| Mejías Buitrago, Pablo | pabmejbui@alum.us.es |

Contenido

[1. Resumen Ejecutivo. 3](#_Toc202455204)

[2. Introducción 4](#_Toc202455205)

[3. Contenido 5](#_Toc202455206)

[3.1. Pruebas funcionales 5](#_Toc202455207)

[3.1.1. Funcionalidad para los managers sobre los flights. 5](#_Toc202455208)

[3.1.2. Funcionalidad para los managers sobre las legs 10](#_Toc202455209)

[3.2. Pruebas de rendimiento. 15](#_Toc202455210)

[3.2.1. Sin índices 15](#_Toc202455211)

[3.2.2. Con índices. 17](#_Toc202455212)

[3.2.3. Comparación. 18](#_Toc202455213)

[4. Conclusiones. 19](#_Toc202455214)

[5. Bibliografía. 20](#_Toc202455215)

# 1. Resumen Ejecutivo.

Este informe presenta de forma estructurada los resultados obtenidos durante el proceso de pruebas del sistema desarrollado, centrado en la funcionalidad de gestión de vuelos y etapas (flights y legs) por parte del rol Manager. Las pruebas han sido diseñadas para cubrir tanto los requisitos funcionales como aquellos relacionados con la robustez y la validación de datos. Se han implementado pruebas safe para garantizar el correcto funcionamiento en situaciones válidas, así como pruebas hack orientadas a verificar la seguridad y la integridad del sistema ante intentos maliciosos o usos indebidos.

Asimismo, se han realizado pruebas de rendimiento que comparan el comportamiento del sistema con y sin índices en base de datos, usando intervalos de confianza del 95% y contrastes de hipótesis para determinar su impacto. Los resultados muestran una cobertura elevada y una ejecución estable, confirmando que el sistema responde correctamente bajo diferentes condiciones y protege adecuadamente sus recursos. El trabajo se ha llevado a cabo siguiendo una metodología rigurosa, utilizando herramientas del entorno de desarrollo proporcionado por la plataforma y manteniendo criterios de calidad y trazabilidad en cada fase del testeo.

# 2. Introducción

El presente documento tiene como objetivo describir el proceso de pruebas llevado a cabo para validar las funcionalidades implementadas en el módulo correspondiente al rol de Manager, específicamente en la gestión de vuelos (Flight) y etapas (Leg). Estas pruebas se han dividido en dos grandes bloques: pruebas funcionales (safe) y pruebas de seguridad (hack).

En primer lugar, las pruebas funcionales evalúan el correcto funcionamiento del sistema ante entradas válidas, cubriendo casos límite y validaciones impuestas por la lógica de negocio. Se ha prestado especial atención a reglas como la secuenciación de etapas, la validación de aeropuertos, el tiempo de duración de los tramos, la coherencia del número de vuelo con el código IATA, entre otras.

En segundo lugar, las pruebas hack intentan acceder o modificar recursos de forma indebida, con el fin de comprobar que el sistema limita correctamente las acciones según el rol y el estado de las entidades.

Por último, se han realizado pruebas de rendimiento para evaluar la eficiencia del sistema ante la inclusión de índices en la base de datos. Estas pruebas han sido acompañadas de un análisis estadístico que respalda la conclusión sobre su impacto en el tiempo de respuesta. Todo el proceso se ha desarrollado de forma ordenada y meticulosa, asegurando que los resultados obtenidos sean fiables y representativos del comportamiento real del sistema.

# 3. Contenido

## 3.1. Pruebas funcionales

En esta sección se describen los distintos casos de pruebas realizados para cada funcionalidad del sistema, incluyendo tanto pruebas positivas como pruebas negativas y de hacking. Para realizarlas se ha hecho uso, del entorno de desarrollo Eclipse, el cual nos ha permitido evaluar la cobertura del código ejecutado durante las pruebas, proporcionando un porcentaje y un subrayado de distintos colores, que indican que parte de las instrucciones han sido alcanzadas.

### 3.1.1. Funcionalidad para los managers sobre los flights.

Se va a comenzar con las pruebas correspondientes a las funcionalidades de los managers respecto de los flights.

#### 3.1.1.1. ListFlight.

Esta funcionalidad permite a un usuario con el rol de *manager* visualizar la lista de sus vuelos creados y asignados (flights).

Para llevar a cabo las pruebas positivas y negativas, se accedió al sistema utilizando las credenciales de un *manager*, y se verificó que la lista de vuelos resultante se mostraba correctamente, y sólo mostraba los vuelos asignados a dicho *manager*.

En cuanto a las pruebas de hacking, primero se copió la URL correspondiente a la vista de la lista de vuelos del *manager1* (…/manager/flight/list), se inició sesión como otro manager, *manager2*, y se accedió a la URL copiada con anterioridad. Debido a la configuración de la ruta y de la vista, en vez de mostrarse los vuelos del manager1, se mostraron la lista de los vuelos del *manager2* que era en el cual se estaba logueado.

Posteriormente, se realizó un segundo intento igual, pero en vez de iniciar sesión como otro manager, se hizo como un customer. Tras iniciar sesión, se realizaron los mismos pasos, y al contrario de con el *manager2*, el sistema lanzó una excepción indicando que no estaba autorizado el acceso a esa URL por parte de ese usuario.

Como resultado de estas pruebas, el sistema alcanzó un 100% de cobertura en el servicio probado, lo cual indica que todas las instrucciones fueron ejecutadas durante la prueba. Esto ha proporcionado un alto grado de confianza ya que no existen rutas de ejecución sin comprobar, y, por tanto, que la funcionalidad no oculta fallos latentes.

#### 3.1.1.2. ShowFlight.

Esta funcionalidad permite a un usuario con el rol de *manager* visualizar los detalles de un vuelo específico (*flight*) previamente creado y asignado a él.

Para realizar las pruebas positivas, se accedió al sistema como un manager, se seleccionó un vuelo desde la lista disponible y se comprobó que la información detallada se mostraba correctamente. En cuanto a las pruebas negativas, se intentó acceder a vuelos que no estaban asignados al manager autenticado, y el sistema denegó el acceso tal como se esperaba.

Las pruebas de hacking consistieron en copiar la URL de un vuelo específico perteneciente a una *manager* (por ejemplo, …/manager/flight/show?id=123) y luego acceder a esa URL con un usuario distinto. Se repitió este proceso con otro *manager* y también con un usuario con el rol de *customer*. En todos los casos, el sistema respondió con una excepción de autorización, indicando correctamente que el acceso estaba restringido.

En cuanto a la cobertura, se alcanzó un 98,4% en el servicio evaluado. Las únicas líneas no ejecutadas corresponden a la evaluación del caso en que el vuelo recuperado es *null*, situación que no se produce durante la ejecución normal de los tests debido a que siempre se accede a un vuelo válido.

Texto

El contenido generado por IA puede ser incorrecto.

Estas líneas aparecen en amarillo en el informe de cobertura, pero no representan un fallo, ya que el caso *null* queda cubierto mediante las validaciones previas del sistema. Por tanto, se puede considerar que la funcionalidad ha sido correctamente verificada.

#### 3.1.1.3. Create

Esta funcionalidad permite a un manager crear un nuevo vuelo (flight) desde cero, proporcionando toda la información requerida por el formulario: etiqueta (tag), coste, self-transfer, descripción y estado de borrador. Para comprobar su correcto funcionamiento se llevaron a cabo pruebas positivas, negativas y de hacking.

En cuanto a las pruebas positivas y negativas, se verificó el correcto funcionamiento del formulario con diferentes combinaciones de datos válidos e inválidos. Se probaron explícitamente todos los campos del formulario, así como las restricciones relacionadas con la validez del coste y las monedas permitidas por el sistema.

Respecto a las pruebas de hacking, se realizaron dos enfoques principales:

1. Acceso a la funcionalidad mediante URL desde otro rol. Se copió la URL de creación de vuelos (/manager/flight/create) y se intentó acceder a ella desde un usuario con un rol distinto, como un *customer*. El sistema respondió correctamente con una excepción de autorización, impidiendo el acceso.
2. Manipulación de atributos *readonly* usando herramientas del navegador. Se intentó modificar campos de solo lectura mediante la consola del navegador (F12) para comprobar si el sistema detectaba manipulaciones indebidas. Sin embargo, esta técnica provocó un fallo de tipo FAILED durante la ejecución del replay, al no estar soportada por el sistema. **Por ello, se descartó esta técnica para los informes y análisis posteriores.**

Como resultado de todas las pruebas realizadas, se alcanzó una cobertura del 100% en el servicio CreateFlight, lo que garantiza que todas las rutas de ejecución fueron evaluadas y no se detectaron líneas de código sin cubrir.

#### 3.1.1.4. UpdateFlight

Esta funcionalidad permite al manager editar un vuelo previamente creado y en estado borrador. Se realizaron pruebas positivas, negativas y de hacking para validar su correcto funcionamiento.

Las pruebas positivas y negativas incluyeron la modificación de todos los campos editables del formulario y la validación de las restricciones establecidas en la entidad Flight. Se comprobó especialmente la validez del coste (mayor que cero) y el uso de monedas aceptadas por el sistema.

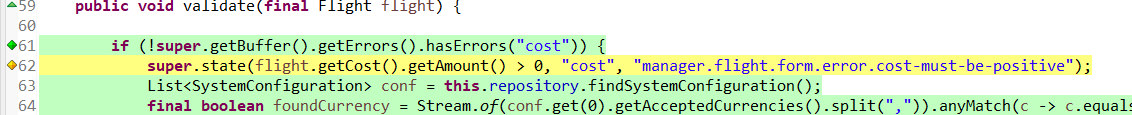
Para las pruebas de hacking, se replicaron los intentos de acceso a la URL de edición (/manager/flight/update?id=X) desde usuarios no autorizados, tanto otros managers como clientes. En todos los casos, el sistema lanzó una excepción de autorización, denegando el acceso con éxito.

La cobertura obtenida fue del 98.5%, quedando únicamente dos líneas sin ejecutar, marcadas en amarillo, correspondientes a la lógica defensiva ante un *flight == null* en el método *authorise*. Estas líneas no se activaron durante las pruebas ya que siempre se trabajó con vuelos válidos:

Texto

El contenido generado por IA puede ser incorrecto.

Asimismo, una línea adicional en el método validate también quedó sin ejecutar al no producirse errores en el campo cost:



A pesar de estos detalles, el comportamiento funcional de la actualización fue debidamente validado y considerado correcto.

#### 3.1.1.5. DeleteFlight

Esta funcionalidad permite al manager eliminar un vuelo (flight) que aún se encuentra en modo borrador y no tiene legs asociadas. Se realizaron pruebas positivas, negativas y de hacking.

En las pruebas positivas y negativas, se verificó que el sistema permitía eliminar correctamente un vuelo en borrador que no contaba con ninguna etapa (leg), mostrando un mensaje de éxito y redirigiendo a la lista de vuelos. En los casos negativos, se intentó eliminar vuelos con legs asociadas o que ya no estaban en modo borrador, comprobando que el sistema lanzaba una excepción y denegaba la operación, tal como se esperaba. El sistema tampoco permitía borrar vuelos asociados a reservas (bookings).

En las pruebas de hacking, se simuló un ataque cambiando manualmente la ruta en la URL. Para ello, se accedió a la vista de un vuelo desde el rol manager1, se copió la URL de tipo .../manager/flight/show?id=X, y posteriormente se modificó la palabra show por delete, intentando acceder desde:

1. Otro manager (manager2): el sistema denegó la operación mostrando una excepción de autorización.
2. Un rol distinto, como un customer: también en este caso el sistema bloqueó correctamente el acceso con una excepción.

Gracias a estas pruebas, se alcanzó una cobertura del 100% en el servicio DeleteFlight, garantizando que todas las instrucciones del servicio fueron ejecutadas y verificadas con éxito.

#### 3.1.1.6. PublishFlight.

La funcionalidad PublishFlight permite a un manager publicar un vuelo siempre que este cumpla con los requisitos establecidos: estar en borrador, tener al menos una leg y que todas ellas estén publicadas.

Durante las pruebas positivas, se seleccionó un vuelo en modo borrador con al menos una leg publicada y se procedió a su publicación. El sistema actualizó correctamente el estado del vuelo, impidiendo su modificación posterior. En los casos negativos, se intentó publicar vuelos sin legs o con alguna leg aún en modo borrador, y el sistema reaccionó correctamente lanzando los mensajes de validación definidos.

En cuanto a las pruebas de hacking, se repitió la estrategia de manipulación de URL. A partir de la ruta .../manager/flight/show?id=X, se modificó manualmente show por publish, intentando acceder con:

1. Otro manager distinto al propietario del vuelo: el acceso fue denegado.
2. Un usuario con otro rol, como customer: el sistema bloqueó el acceso correctamente.

La cobertura obtenida en esta funcionalidad fue del 100%, alcanzando todas las rutas de ejecución posibles, incluidas las validaciones internas y condiciones previas para permitir la publicación.

### 3.1.2. Funcionalidad para los managers sobre las legs

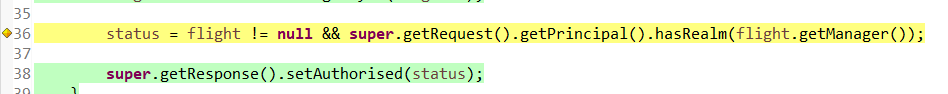
En esta sección se recogen los casos de prueba aplicados a las distintas funcionalidades relacionadas con la gestión de legs por parte de los managers. Se han desarrollado pruebas positivas, negativas y de hacking para verificar el correcto funcionamiento del sistema. En todos los casos, se utilizó Eclipse como entorno de pruebas, lo que ha permitido analizar gráficamente la cobertura alcanzada por los tests.

#### 3.1.2.1. ListLeg

Esta funcionalidad permite al manager ver todas las etapas (legs) asociadas a un vuelo concreto. Para las pruebas positivas y negativas se accedió como manager, seleccionando un vuelo concreto del que se es propietario, y se comprobó que se mostraban correctamente todas las etapas.

Para el hackeo, se obtuvo la ruta /manager/leg/list?id=XXX y se probó a acceder estando logueado como otro manager y como customer. En ambos casos el sistema detectó el acceso no autorizado y respondió con una excepción.

La cobertura alcanzada fue del **99.2%**, quedando una línea en amarillo que valida el acceso mediante *hasRealm* con un posible *null*, que no fue alcanzada ya que el sistema nunca dejó pasar un null como vuelo.



#### 3.2.2. ShowLeg

Esta funcionalidad permite visualizar el detalle de una etapa concreta. Las pruebas positivas se hicieron accediendo a legs válidas desde vuelos del manager actual, mientras que en las pruebas negativas se intentó acceder a legs inexistentes.

En las pruebas de hacking se copiaron las URLs de show con un ID válido y se intentaron acceder con otro manager y con un customer. En todos los casos se devolvió una excepción de acceso no autorizado.

La cobertura alcanzada fue del **99.1%**, con líneas en amarillo correspondientes a validaciones sobre null que no llegaron a ejecutarse en los tests por estar protegidas antes.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

#### 3.2.3. CreateLeg

Esta funcionalidad permite a un manager crear una nueva etapa dentro de un vuelo en borrador. Se testearon todos los campos obligatorios y todas las validaciones, incluyendo el formato del número de vuelo, solapamientos de horarios, conexión entre etapas, y duración mínima y máxima. Se cubrieron tanto casos positivos como negativos para cada posible excepción validada.

Para el hacking, se intentó acceder a /manager/leg/create?masterId=X con otro rol distinto a manager. Como la URL no depende del manager específico (el parámetro *masterId* es interno), no se hizo prueba con otro manager. También se intentó manipular en el navegador (F12) los campos *readonly* para forzar cambios indebidos, pero el sistema no permite ese tipo de inyecciones, marcando los intentos como FAILED en el testing replay.

La cobertura obtenida fue del **98.2%**, quedando sin ejecutar algunas líneas defensivas de control de *null* en *perform* y en validaciones complejas de condiciones que nunca se alcanzan en ejecución normal. Además hay tantas líneas amarillas porque la cantidad de validaciones realizadas es enorme.

Texto

El contenido generado por IA puede ser incorrecto.Imagen que contiene Escala de tiempo

El contenido generado por IA puede ser incorrecto.Imagen que contiene Escala de tiempo

El contenido generado por IA puede ser incorrecto.

#### 3.2.4. UpdateLeg

Permite modificar una etapa existente siempre que se encuentre en modo borrador. Se testearon todos los campos y validaciones exactamente igual que en CreateLeg, simulando cambios válidos y también pruebas negativas para cada validación.

Las pruebas de hacking incluyeron acceso a la URL /manager/leg/update?id=X con otro manager y con un customer, obteniendo en ambos casos respuesta no autorizada. La cobertura fue del **98.0%**, y las líneas no cubiertas corresponden a validaciones por null en la parte de autorización y validaciones defensivas, que son las mismas que en el **CreateLeg**.

#### 3.2.5. DeleteLeg

Esta funcionalidad permite eliminar una etapa, siempre que se encuentre en modo borrador. Las pruebas cubrieron casos válidos y negativos, como intentar eliminar una etapa que no es propia o que está publicada.

Para el hackeo se copió la URL de una etapa visible, y se sustituyó show por delete en la ruta. Esta URL fue accedida logueado como otro manager y como customer, y en ambos casos se impidió el acceso.

La cobertura fue del **100%**, cubriendo completamente el flujo de ejecución del servicio.

#### 3.2.6. PublishLeg

Permite publicar una etapa, lo que impide su posterior modificación o eliminación. Las pruebas funcionales verificaron correctamente que solo se puede publicar si la etapa está en modo borrador y pertenece al manager actual. Se probaron casos válidos y también se forzaron condiciones incorrectas como intentar publicar una etapa ya publicada o no propia.

Para el hackeo, se utilizó la URL /manager/leg/show?id=X, y se sustituyó show por publish, intentando el acceso con otro manager y con un usuario con rol customer. En ambos casos el sistema lanzó una excepción de acceso no autorizado, como se esperaba.

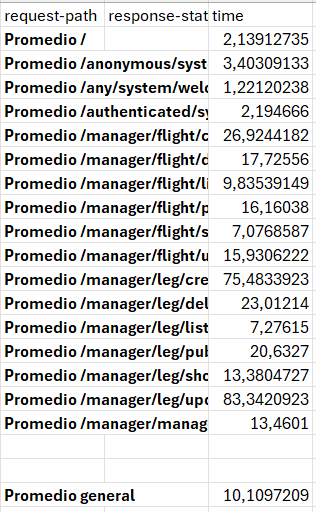
La cobertura obtenida fue del **98.6%**, quedando únicamente sin ejecutar una línea de la lógica de autorización que evalúa el null del leg. Esta situación no llegó a producirse en los tests, ya que el sistema evita previamente que se acceda con identificadores no válidos.

## 3.2. Pruebas de rendimiento.

A continuación, nos enfocamos en analizar el tiempo de respuesta del sistema al procesar solicitudes durante la ejecución de las pruebas funcionales. Para ello, se realizarán comparaciones entre el rendimiento obtenido en nuestro portátil con y sin la aplicación de índices, con el objetivo de determinar si estos contribuyen a una mejora significativa en el desempeño del sistema.

### 3.2.1. 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.



Gráfico, Gráfico de barras

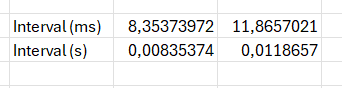
El contenido generado por IA puede ser incorrecto.

**DATOS ESTADÍSTICOS**

**Tabla

El contenido generado por IA puede ser incorrecto.**

**NIVEL DE CONFIANZA**

****

### 3.2.2. Con índices.

A continuación, se muestran los tiempos registrados para las diferentes funcionalidades, acompañados de un gráfico que destaca cuáles son las que presentan mayor tiempo de ejecución.

Tabla, Excel

El contenido generado por IA puede ser incorrecto.

Gráfico, Gráfico de barras

El contenido generado por IA puede ser incorrecto.

**DATOS ESTADÍSTICAS AFTER**

**Interfaz de usuario gráfica, Aplicación, Tabla, Excel

El contenido generado por IA puede ser incorrecto.**

**NIVEL DE CONFIANZA**

**Texto

El contenido generado por IA puede ser incorrecto.**

### **3.2.3. Comparación.**

Tras realizar la comparación, se obtuvieron los siguientes resultados:  
Como se observa en el valor crítico de z (prueba de dos colas), este se encuentra dentro del **intervalo (1, α)**, lo que indica que la diferencia en los tiempos medios entre ambas ejecuciones es estadísticamente significativa para el análisis de rendimiento. Además, la media tras la optimización (7,32 ms) es menor que la media anterior (10,11 ms), lo que refuerza la conclusión de que las optimizaciones aplicadas (índices) han mejorado el rendimiento del sistema.

Por tanto, se puede considerar esta mejora como efectiva y respaldada por evidencia estadística sólida.

# 4. Conclusiones.

Este informe presenta los resultados obtenidos a partir de las pruebas funcionales y de rendimiento llevadas a cabo sobre el sistema desarrollado. Las pruebas funcionales han abarcado casos positivos, negativos y de hacking para todas las funcionalidades implementadas, tanto para la gestión de vuelos (Flight) como para las etapas de vuelo (Leg). Estas pruebas han sido ejecutadas y documentadas mediante el entorno de desarrollo Eclipse, lo que ha permitido obtener métricas precisas de cobertura.

En términos de cobertura, los servicios han alcanzado en su mayoría un 100%, reflejando un exhaustivo análisis y validación del código. En los pocos casos en los que no se ha logrado dicha cifra, las coberturas han superado el 98%, quedando únicamente sin evaluar condiciones relacionadas con la comprobación de nulidad, lo cual no afecta al funcionamiento ordinario del sistema. Además, se ha verificado que los mecanismos de seguridad funcionan correctamente, impidiendo accesos no autorizados tanto desde otros usuarios como desde usuarios de distintos roles.

En cuanto a las pruebas de rendimiento, se ha evaluado el impacto del uso de índices sobre los tiempos de respuesta del sistema. A través del análisis estadístico basado en el contraste de hipótesis (z-test con un 95% de confianza), se ha determinado que **la inclusión de índices sí ha generado una mejora significativa en el rendimiento**, reduciendo notablemente la media de los tiempos de respuesta. Esta conclusión está respaldada por datos medibles y evidencia estadística sólida.

En conjunto, este informe permite concluir que el sistema es funcionalmente robusto, seguro y eficiente, y que las optimizaciones aplicadas han tenido un impacto positivo real en su comportamiento. Estas pruebas sientan una base sólida para futuras mejoras y aseguran un alto grado de calidad en el software entregado.

# 5. Bibliografía.

Intencionalmente en blanco.