Universidad de Sevilla  
Escuela Técnica Superior de Ingeniería Informática

TESTING REPORT INDIVIDUAL - D04



**C1.026**

**Repositorio**: https://github.com/agudevbon/Acme-ANS-D04

**Autor:**

Devós Bono, Agustín

agudevbon@alum.us.es

**Fecha:** 20/05/2025

**Tabla de contenidos**

[**Resumen**](#_heading=h.gjdgxs) **2**

[**Tabla de revisión**](#_heading=h.30j0zll) **2**

[**Introducción**](#_heading=h.1fob9te) **2**

[**Contenidos**](#_heading=h.3znysh7) **3**

**Capítulo I: Testing funcional 3**

**Capítulo II: Testing de rendimiento 4**

[**Bibliografía**](#_heading=h.3dy6vkm) **4**

# Resumen

Este documento se detalla el proceso de testing seguido sobre los requisitos individuales de entregas anteriores, con el objetivo de llegar a una conclusión de dichos resultados en materia de mejora del código o del mismo proceso de testing. El documento consta de dos capítulos que tratan de los métodos estadísticos realizados para calcular intervalos de confianza y el contraste de hipótesis, alcanzando una valoración final.

# Tabla de revisión

|  |  |  |
| --- | --- | --- |
| **Versión** | **Fecha** | **Descripción** |
| V1 | 20/05/2025 | Creación y desarrollo del documento |

# Introducción

A continuación, veremos separados en dos capítulos, los métodos estadísticos realizados para analizar los resultados tras el testing formal de los requisitos de todas las entregas del vuelo.

# Contenidos

# Capítulo I: Testing funcional

Este capítulo mostrará el listado con los casos de prueba implementados, agrupados por *features*. Para cada funcionalidad a probar se explicarán los casos positivos y negativos probados en los archivos .safe y los intentos de hackeo realizados en los archivos .hack. Para mayor legibilidad los datos considerados válidos e inválidos para los campos de los formularios se incluirán en un Excel aparte con el nombre Testing Data en la misma carpeta D04 en la que encontró este documento.

**Entidad Flight**

**manager/flight/create:**

Para probar esta funcionalidad en primer lugar, se ha enviado el formulario vacío y después se han probado todos los casos positivos y negativos de cada campo, tras esto, se ha creado un objeto completamente valido.

Para la prueba de hacking se ha intentado acceder al formulario desde un rol distinto a manager e intentar realizar la acción de crear para un id existente o no existente pero distinto de 0. Obteniendo en todos los casos el correspondiente error 500 not authorised.

No se encontraron bugs con el testeo de esta funcionalidad.

**manager/flight/delete:**

Para realizar esta prueba, se ha probado a eliminar un vuelo no publicado sin tramos asignados, otro vuelo no publicado con tramos asignados y un vuelo no publicado con tramos asignados que estén publicados recibiendo en este caso un error de validación que indica al usuario que no es posible borrar un vuelo si sus tramos están publicados.

* Al intentar realizar esta última se detectó un bug que permitía eliminar vuelos con tramos publicados, borrando así objetos *Leg* con draft mode en false, este bug fue corregido tras las pruebas.

Para la prueba de hacking se ha intentado eliminar con el mismo usuario, un vuelo inexistente y un vuelo ya publicado. Con manager2 se ha intentado eliminar un vuelo de manager1 y, por último, con el rol Member se ha intentado eliminar un vuelo. Todas estas pruebas han devuelto correctamente un error 500 not authorised.

**manager/flight/list:**

Para realizar esta prueba, se ha accedido al listado de vuelos un rol manager comprobando que se lista adecuadamente, completando la prueba positiva de dicha *feature*.

Para la prueba de hacking, se ha intentado listar los vuelos de un manager desde el rol Member, recibiendo el correspondiente error de acceso.

No se encontraron bugs durante el testeo de esta funcionalidad.

**manager/flight/publish:**

Para comprobar el correcto comportamiento de esta funcionalidad y sus restricciones se han realizado las siguientes pruebas:

* Publicar satisfactoriamente un vuelo con un tramo asignado y publicado.
* Publicar satisfactoriamente un vuelo con varias tramos asignados y publicados.
* Publicar un vuelo con varios tramos asignados, pero no publicados, y recibir el correspondiente aviso de, no posible.
* Publicar un vuelo sin tramos asignados, y recibir el correspondiente aviso de, no posible.

Para la prueba de hacking se ha intentado publicar con el mismo usuario, un vuelo inexistente y un vuelo ya publicado. Con manager2 se ha intentado publicar un vuelo de manager1 y, por último, con el rol Member se ha intentado publicar un vuelo. Todas estas pruebas han devuelto correctamente un error 500 not authorised.

No se encontraron bugs durante el testeo de esta funcionalidad.

**manager/flight/show:**

Para realizar esta prueba, se ha accedido al listado de vuelos para entrar a los detalles de un vuelo publicado y después los detalles de un vuelo no publicado, comprobando que se muestra adecuadamente.

* Sin embargo se detectó un bug en el show de vuelos publicados, ya que no aparecía el botón de *Tramos asociados.*

Este bug fue resuelto gracias al testing completando la prueba positiva de dicha *feature*.

Para la prueba de hacking se ha intentado mostrar con el mismo usuario, un vuelo inexistente. Con manager2 se ha intentado mostrar un vuelo de manager1 y, por último, con el rol Member se ha intentado mostrar un vuelo. Todas estas pruebas han devuelto el correspondiente error 500 not authorised.

**manager/flight/update:**

Para probar esta funcionalidad en primer lugar, se ha enviado el formulario vacío y después se han probado todos los casos positivos y negativos de cada campo, tras esto, se ha actualizado un objeto completamente valido.

Para la prueba de hacking se ha intentado actualizar con el mismo usuario, un vuelo inexistente y un vuelo ya publicado. Con manager2 se ha intentado actualizar un vuelo de manager1 y, por último, con el rol Member se ha intentado actualizar un vuelo. Todas estas pruebas han devuelto correctamente un error 500 not authorised.

No se encontraron bugs durante el testeo de esta funcionalidad.

**Entidad Leg**

**manager/leg/create:**

Para probar esta funcionalidad se han probado todos los casos positivos y negativos de cada campo del form. En primer lugar, se ha enviado el formulario vacío y después se han probado todos los casos positivos y negativos de cada campo, tras esto, se ha creado un objeto completamente valido.

Para la prueba de hacking se ha intentado crear un tramo con atributos de navegación inválidos. Siendo dichos atributos de navegación *departure(Airport), arrival(Airport), aircraft(Aircraft), flight(Flight),* para todos ellos se ha probado a crear un objeto que referencie a un atributo de navegación inexistente. En concreto para los atributos *aircraft* y *flight* hay casos especiales*.*

* Una aeronave cuyo atributo *status* no sea **'ACTIVE\_SERVICE'**
* Un vuelo publicado del mismo manager, un vuelo no publicado de otro manager y un vuelo publicado de otro manager.

También se intentó acceder al formulario desde un rol distinto a manager obteniendo el correspondiente error 500 en todas las pruebas mencionadas.

No se encontraron bugs durante el testeo de esta funcionalidad.

**manager/leg/delete:**

Para realizar esta prueba, se ha probado a eliminar un tramo no publicado asignado a algún vuelo, para comprobar el correcto funcionamiento del borrado de las asignaciones de la entidad intermedia.

Para la prueba de hacking se ha intentado eliminar con el mismo usuario, un tramo inexistente y un tramo ya publicado. Con manager2 se ha intentado eliminar un tramo de manager1 y, por último, con el rol Member se ha intentado eliminar un tramo. Todas estas pruebas han devuelto correctamente un error 500 not authorised.

No se encontraron bugs durante el testeo de esta funcionalidad.

* **Explicación lineas rojas coverage:** En el service ManagerUserStoryDeleteService.java se implementa un unbind, sin embargo como este servicio no tiene validaciones nunca se ejecuta dicho unbind por lo que dicho método no está cubierto por los tests.

**manager/leg/list-by-Flight:**

Para realizar esta prueba, se ha accedido a los detalles de un vuelo tanto publicado como no publicado para después acceder al listado de tramos asignadas a dicho vuelo comprobando que se lista adecuadamente, completando la prueba positiva de dicha *feature*.

Para la prueba de hacking se ha intentado listar con el mismo usuario, las tramos de un vuelo inexistente. Con manager2 se ha intentado listar las tramos de un vuelo de manager1 y, por último, con el rol Member se ha intentado listar las tramos de un vuelo. Todas estas pruebas han devuelto correctamente un error 500 not authorised.

No se encontraron bugs durante el testeo de esta funcionalidad.

**manager/leg/list:**

Para realizar esta prueba, se ha accedido al listado de tramos de un rol manager comprobando que se lista adecuadamente, completando la prueba positiva de dicha *feature*.

Para la prueba de hacking, se ha intentado listar las tramos de un manager desde el rol Member, recibiendo el correspondiente error de acceso.

No se encontraron bugs durante el testeo de esta funcionalidad.

**manager/leg/publish:**

Para realizar esta prueba, se ha probado a publicar un tramo que se solape con otro tramo del mismo vuelo recibiendo el aviso de dato erróneo corresponediente, tras comprobar todas las combinaciones posibles de solapamiento se publicó el tramo con datos válidos.

Para la prueba de hacking se ha intentado crear un tramo con atributos de navegación inválidos. Siendo dichos atributos de navegación *departure(Airport), arrival(Airport), aircraft(Aircraft), flight(Flight),* para todos ellos se ha probado a crear un objeto que referencie a un atributo de navegación inexistente. En concreto para los atributos *aircraft* y *flight* hay casos especiales*.*

* Una aeronave cuyo atributo *status* no sea **'ACTIVE\_SERVICE'**
* Un vuelo publicado del mismo manager, un vuelo no publicado de otro manager y un vuelo publicado de otro manager.

También se ha intentado publicar con el mismo usuario, un tramo inexistente y un tramo ya publicado. Con manager2 se ha intentado publicar un tramo de manager1 y, por último, con el rol Member se ha intentado publicar un tramo. Todas estas pruebas han devuelto correctamente un error 500 not authorised.

No se encontraron bugs durante el testeo de esta funcionalidad.

**manager/leg/show:**

Para realizar esta prueba, se ha accedido al listado de tramos para entrar a los detalles de un tramo publicada y después los detalles de un tramo no publicada, comprobando que se muestra adecuadamente, completando la prueba positiva de dicha *feature*.

Para la prueba de hacking se ha intentado mostrar con el mismo usuario, un tramo inexistente. Con manager2 se ha intentado mostrar un tramo de manager1 y, por último, con el rol Member se ha intentado mostrar un tramo. Todas estas pruebas han devuelto el correspondiente error 500 de no autorizado.

No se encontraron bugs durante el testeo de esta funcionalidad.

**manager/leg/update:**

Para probar esta funcionalidad en primer lugar, se ha enviado el formulario vacío y después se han probado todos los casos positivos y negativos de cada campo, tras esto, se ha actualizado un objeto completamente valido.

* Ya que en la funcionalidad de creado no se muestran los campos de atributos derivados, en este caso el campo de *duración*, no fue hasta el testeo de la funcionalidad de actualización que se detecto un bug en relación a este campo. Si se dejaba alguna de las fechas de salida o llegada como nula se recibía un error NullPointerException ya que no se podía calcular el atributo derivado si faltaban datos. Este error fue arreglado mostrando el campo *duración* vacío en caso de que alguno de los atributos de los que depende sea nulo a la hora de tratar de actualizar.

Para la prueba de hacking se ha intentado crear un tramo con atributos de navegación inválidos. Siendo dichos atributos de navegación *departure(Airport), arrival(Airport), aircraft(Aircraft), flight(Flight),* para todos ellos se ha probado a crear un objeto que referencie a un atributo de navegación inexistente. En concreto para los atributos *aircraft* y *flight* hay casos especiales*.*

* Una aeronave cuyo atributo *status* no sea **'ACTIVE\_SERVICE'**
* Un vuelo publicado del mismo manager, un vuelo no publicado de otro manager y un vuelo publicado de otro manager.

También se ha intentado actualizar con el mismo usuario, un tramo inexistente y un tramo ya publicado. Con manager2 se ha intentado actualizar un tramo de manager1 y, por último, con el rol Member se ha intentado actualizar un tramo. Todas estas pruebas han devuelto correctamente un error 500 not authorised.

Por otro lado, las pruebas realizadas en modo ***cobertura*** han mostrado un porcentaje de cobertura adecuado a las pruebas realizadas, el cual quedaría resumido en las siguientes capturas.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

ACME.CONSTRAINTS



Podemos observar que las validaciones de las entidades no están probadas al 100% esto se debe a dos factores.

* Asserts que no se pueden ejecutar probando la aplicación.
* Validaciones que se manejan en los servicios, pero también, incluidas en los validadores para evitar datos erróneos introducidos al tratar de popular la base de datos.

# Capítulo II: Testing de rendimiento

En este capítulo se proporcionarán gráficos adecuados y un intervalo de confianza del 95 % para el tiempo que tarda el vuelo en atender las solicitudes de las pruebas funcionales en dos equipos diferentes, además de un contraste de hipótesis de confianza del 95 % con respecto a cuál es el equipo más potente.

**Intervalo de confianza**

Para este apartado se han recolectado 975 datos de los distintos tests y se ha calculado el **intervalo de confianza con un nivel del 95%** en el mismo dispositivo, pero implementando mejoras de rendimiento en el código.

Tras ejecutar los tests en primer lugar con el código base se consiguió la cobertura indicada anteriormente, podemos observar los resultados en cuanto a tiempo de consulta en la Columna 1 de la imagen inferior. Para tratar de mejorar el tiempo de respuesta en la segunda batería de tests se añadió índices “*@Table”* en las entidades *Flight y Leg.*

Tabla, Excel

El contenido generado por IA puede ser incorrecto.

Para ver los resultados y tiempos de ejecución en mayor detalle puede acceder al archivo **tester-performance-clean.xlsx** adjunto en la entrega

Los tests ejecutados en la primera batería de tests mostraron que la función: **actualizar un trayecto(Leg) era la función menos eficiente (MIR)**, tal y como muestra el siguiente gráfico del tiempo (en ms) de cada feature probada.

Para tratar de mejorar el tiempo de respuesta en la segunda batería de tests se añadió índices “*@Table”* en las entidades Flight *y Leg*. Vemos a continuación los resultados obtenidos tras aplicar este cambio para la mejora de rendimiento de la aplicación.

Tabla

El contenido generado por IA puede ser incorrecto.

Para ver los resultados y tiempos de ejecución en mayor detalle puede acceder al archivo **tester-performance2-clean.xlsx** adjunto en la entrega

Analicemos ahora también el gráfico de los tiempos de respuesta promedio. Tras la implementación de dicho cambio **la consulta más ineficiente (MIR) sigue siendo la actualización de tramos.**

Comparación en paralelo de los análisis de datos para ambas pruebas:

Tabla

El contenido generado por IA puede ser incorrecto.

**Contraste de hipótesis**

Para comparar adecuadamente los intervalos de confianza calculados en ambas máquinas tras ejecutar las pruebas, se ha considerado realizar la prueba Z-Test sobre las columnas de tiempos generados por ambas máquinas (antes y después), quedándonos con el valor del campo de la tabla generada llamado ***Valor crítico de z (dos colas)*** ***=*** ***0.03171.***

Tabla

El contenido generado por IA puede ser incorrecto.

Dicho valor crítico de z nos indica que nos encontramos en el intervalo *( 0.00 - α ]*, donde, ***α*** *= alpha = 1 - Intervalo de Confianza = 1 - 0.95 =* ***0.05***

Al ser z = 0.03171 tenemos que: *0 =< z < a*

Tras la obtención de estos datos podemos concluir que incluir no solo no han reducido el tiempo medio de las consultas si no que la han aumentado en 2 segundos. Sorprende la gran diferencia de tiempo de ejecución por lo que añado a continuación una hipótesis del por qué:

Podemos observar en la primera gráfica de barras que las operaciones que más tardan son:

* Borrar vuelo:
* Borrar relación vuelo-historia de usuario
* Crear relación vuelo- historia de usuario
* Borrar historia de usuario

Identificamos que todas estas consultas crean o borran algún objeto de la entidad *FlightUserStory,* al haber implementado @Index en los dos únicos atributos se ha acelerado toda consulta de creación o borrado de los objetos de esta entidad. Estas 4 consultas eran los picos más altos y con mayor diferencia con respecto al resto de promedios. Al regularse y estandarizarse el promedio de tiempo de las consultas la varianza baja en gran cantidad lo que ha dado lugar, sumado con la bajada de tiempo de consulta base de el resto de entidades tras implementar los índices, ha concluido en una gran implementación de mejora de rendimiento.

Tras los resultados obtenidos, no ha sido necesario realizar un estudio del rendimiento del dispositivo puesto que ya arrojaba buenos valores en el tiempo de ejecución de las pruebas.

Los cálculos detallados del análisis estadístico de este capítulo puede comprobarse en el documento adjunto a esta entrega llamado ***tester-z-test.xlsx***

# Bibliografía

-Documento 08 Annexes, de los contenidos de la plataforma virtual de esta asignatura.

-S02 Performance testing, transparencias del módulo de testing L04 de la asignatura