INFORME DE TESTING INDIVIDUAL

C1.020 | https://github.com/DP2-C1-021/Acme-ANS-C2

|  |  |
| --- | --- |
| **Nombre** | **Correo Corporativo** |
| Álvaro Jiménez Osuna | [alvjimosu@alum.us.es](mailto:alvjimosu@alum.us.es) |
| Paula Sánchez Gómez | [Pausagom1@alum.us.es](mailto:Pausagom1@alum.us.es) |
| Mario Reyes Apresa | [marreyapr@alum.us.es](mailto:marreyapr@alum.us.es) |
| Virginia Mesa Pérez | [virmesper@alum.us.es](mailto:virmesper@alum.us.es) |

|  |  |
| --- | --- |
| **Historial de versiones** | **Comentarios** |
| v1.1 | Realización del documento |
|  |  |

03 DE JULIO DE 2025

ÁLVARO JIMÉNEZ OSUNA | STUDENT #3

**Tabla de Contenido**

1. [Testing Funcional 2](#_bookmark0)
   1. [Introducción 2](#_bookmark1)
   2. [Metodología 2](#_bookmark2)
   3. [Casos de prueba por característica 3](#_bookmark3)
      1. [Para FlightAssignment 3](#_bookmark4)
      2. [Para ActivityLog 8](#_bookmark5)
   4. [Calidad de la cobertura 12](#_bookmark6)
   5. [Conclusiones finales 13](#_bookmark7)
2. [Testing de Rendimiento 13](#_bookmark8)
   1. [Entorno y protocolo 13](#_bookmark9)
   2. [Recogida de datos 13](#_bookmark10)
   3. [Estadísticas Descriptivas 14](#_bookmark11)
   4. [Gráficos de eficiencia por característica 15](#_bookmark12)
   5. [Hipótesis y conclusión 17](#_bookmark13)
3. [Testing de Mutaciones 17](#_bookmark14)
   1. [Primera mutación: Intercambio de valores 17](#_bookmark15)
   2. [Segunda mutación: Eliminar línea de código en validación 18](#_bookmark16)
   3. [Tercera mutación: Cambiar lógica de authorise 19](#_bookmark17)
   4. [Cuarta mutación: Eliminar dataset en el unbind 19](#_bookmark18)
   5. [Quinta mutación: Alterar operador de consulta 20](#_bookmark19)
4. [Tabla de Revisión 20](#_bookmark20)

# Testing Funcional

## Introducción

En esta sección del documento se pretende describir un listado con los casos de prueba realizados, agrupados por características, para las entidades **FlightCrewMember** y **ActivityLog**.

Para cada caso de prueba, se va a proporcionar una descripción y cuál es su eficacia para detectar errores.

## Metodología

Para probar las entidades obligatorias del estudiante tres, se han aplicado pruebas **“.safe”** y **“.hack”**. Para las primeras se ha probado que las entidades se listen, muestren y permitan un CRUD válido y funcional, permitiendo todo tipo de valores válidos y rechazando aquellos que no cumplen con la validación.

Por otro lado, para las segundas, generalmente se han dividido en pruebas de **“url”** y en pruebas de **valores**, principalmente por dos motivos. El primero es que solía dar problemas hacer las dos pruebas juntas, y el segundo es que permitía ser mucho más específico en cada una de las pruebas.

Por último, se muestra una tabla con la cobertura total alcanzada. En lo que respecta a los paquetes globales:

|  |  |
| --- | --- |
| **Paquete** | **Cobertura alcanzada** |
| crewMember.assignment | 99,2% |
| crewMember.activityLog | 98,8% |

Por otro lado, en lo que respecta a cada servicio específico de FlightAssignment:

|  |  |
| --- | --- |
| **Servicio** | **Cobertura alcanzada** |
| FlightAssignmentController | 100% |
| FlightAssignmentCreateService | 99.3% |
| FlightAssignmentDeleteService | 97.4% |
| FlightAssignmentListService | 100% |
| FlightAssignmentListServiceUncompletedLegs | 100% |
| FlightAssignmentPublishService | 99.4% |
| FlightAssignmentShowService | 100% |
| FlightAssignmentUpdateService | 98.8% |

Para terminar, para los servicios específicos de ActivityLog:

|  |  |
| --- | --- |
| **Servicio** | **Cobertura alcanzada** |
| ActivityLogController | 100% |
| ActivityLogCreateService | 99.5% |
| ActivityLogDeleteService | 97.2% |
| ActivityLogListService | 99.4% |
| ActivityLogPublishService | 98.9% |
| ActivityLogShowService | 99.4% |
| ActivityLogUpdateService | 97.5% |

Para los controladores de ambas entidades se alcanza un 100% de la cobertura, lo que significa que se prueban todas las líneas de código contenidas en ellos (es decir, todos los servicios que ofrecen).

## Casos de prueba por característica

### Para FlightAssignment

Comencemos especificando los casos de pruebas para las características **“flight- crew-member/flight-assignment/list-completed”** y **“flight-crew- member/flight-assignment/list-planned”**.

|  |  |  |
| --- | --- | --- |
| **Caso de prueba** | **Tipo** | **Descripción** |
| **list.safe** | Renderizado | Se prueba que el listado se muestre  correctamente para asignaciones completadas y  planeadas. |
| **list.hack** | Hack | Se prueba que:  -Miembros con otro rol no puedan ver el listado. |

#### Pasemos a “flight-crew-member/flight-assignment/show?id=xx”:

|  |  |  |
| --- | --- | --- |
| **Caso de prueba** | **Tipo** | **Descripción** |
| **show.safe (para en y es)** | Renderizado | Se prueba que una asignación se  renderice correctamente sin  ningún error. |
| **show.hack** | Hack | Permite probar:  -Miembros con otro rol no puedan ver los detalles de una  asignación.  -Miembros con mismo rol no puedan ver  asignaciones que no le  corresponden. |

En lo que respecta a **“flight-crew-member/flight-assignment/create”:**

|  |  |  |
| --- | --- | --- |
| **Caso de prueba** | **Tipo** | **Descripción** |
| **create.safe** | Casos positivos y negativos del formulario | Se prueba que todos los campos del formulario estén correctamente  validados, rechazando los valores no admitidos por el  modelo. |
| **create.hack** | Hack (url y valores) | Permite probar:  -Miembros con otro rol no puedan acceder al formulario de creación |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | - No se pueda asignar otro miembro que no sea el “logueado” a la asignación.  -Atributos read-only no puedan ser  modificados.  - Atributos de  selección no puedan ser modificados con  valores ilegales . |  |

En lo que respecta a **“flight-crew-member/flight-assignment/update” :**

|  |  |  |
| --- | --- | --- |
| **Caso de prueba** | **Tipo** | **Descripción** |
| **update.safe** | Casos positivos y  negativos del formulario | Se prueba que todos los campos del formulario estén correctamente validados, rechazando  los valores no admitidos  por el modelo. |
| **update- url.hack** | Hack (url) | Permite probar:  -Miembros con otro rol no puedan acceder a los detalles de la asignación por medio del “update”.  - Miembros del mismo rol no puedan acceder a los detalles de una  asignación que no les pertenece por medio del “update”.  - Asignaciones ya  publicadas no puedan ser actualizadas. |
| **update- values.hack** | Hack (values) | Permite probar que:  -Atributos read-only no puedan ser modificados.  - Atributos de selección no puedan ser  modificados con valores  ilegales. |

En lo que respecta a **“flight-crew-member/flight-assignment/publish”:**

|  |  |  |
| --- | --- | --- |
| **Caso de prueba** | **Tipo** | **Descripción** |
| **publish.safe** | Casos positivos y  negativos del formulario | Se prueba que todos los campos del formulario estén correctamente validados, rechazando  los valores no admitidos  por el modelo. |
| **publish- url.hack** | Hack (url) | Permite probar:  -Miembros con otro rol no puedan acceder a los detalles de la asignación por medio del “publish”.  - Miembros del mismo rol no puedan acceder a los detalles de una  asignación que no les pertenece por medio del “publish”.  - Asignaciones ya  publicadas no puedan volver a publicarse. |
| **publish - values.hack** | Hack (values) | Permite probar que:  -Atributos read-only no puedan ser modificados.  - Atributos de selección no puedan ser  modificados con valores  ilegales. |

Por último, para **“flight-crew-member/flight-assignment/delete” :**

|  |  |  |
| --- | --- | --- |
| **Caso de prueba** | **Tipo** | **Descripción** |
| **delete.safe** | Casos positivos y negativos del formulario | -Se prueba que una asignación se borre  correctamente |
| **delete- url.hack** | Hack (url) | Permite probar:  -Miembros con otro rol no puedan acceder a  los detalles de la  asignación por medio del “delete”.  - Miembros del mismo rol no puedan acceder a los detalles de una asignación que no les pertenece por medio del “delete”.  - Asignaciones ya  publicadas no puedan ser eliminadas. |
| **delete- values.hack** | Hack (values) | Permite probar que:  -Atributos read-only no puedan ser  modificados.  - Atributos de selección no puedan ser  modificados con  valores ilegales. |

### Para ActivityLog

Comencemos especificando los casos de pruebas para las características **“flight- crew-member/activity-log/list?masterId=xx”**

|  |  |  |
| --- | --- | --- |
| **Caso de prueba** | **Tipo** | **Descripción** |
| **list.safe** | Renderizado | Se prueba que el listado de incidentes se muestre correctamente para  asignaciones completadas, ya que solo se pueden  registrar incidentes cuando  la etapa ha terminado |
| **list.hack** | Hack | Se prueba que:  -Miembros con otro rol no puedan ver un listado de  incidencias.  -Miembros del mismo rol no puedan ver un listado de  incidencias de una asignación que no les  pertenece.  - Miembros del rol correcto no puedan ver el listado de incidencias de  asignaciones no  publicadas o planeadas |

#### Pasemos a “flight-crew-member/activity-log/show?id=xx”:

|  |  |  |
| --- | --- | --- |
| **Caso de prueba** | **Tipo** | **Descripción** |
| **show.safe** | Renderizado | Se prueba que una  incidencia se renderice correctamente sin ningún error. |
| **show.hack** | Hack | Permite probar:  -Miembros con otro rol no puedan ver los detalles de una incidencia.  -Miembros con mismo rol no puedan ver  incidencia de asignaciones que no le corresponden. |

En lo que respecta a **“flight-crew-member/activity-log/create?masterId=xx”:**

|  |  |  |
| --- | --- | --- |
| **Caso de prueba** | **Tipo** | **Descripción** |
| **create.safe** | Casos positivos y negativos del formulario | Se prueba que todos los campos del formulario estén correctamente  validados, rechazando los valores no admitidos por el modelo. |
| **create.hack** | Hack (url y valores) | Permite probar:  -Miembros con otro rol no puedan acceder al formulario de creación de un incidente.   * Miembros del mismo rol no puedan acceder al formulario de creación de un incidente asociado a una asignación que no les corresponde. * Miembros del rol correcto no puedan acceder al formulario de creación de un incidente asociado a una asignación en modo borrador o a una asignación   cuya etapa no ha terminado  - La fecha (atributo read- only) no pueda ser  modificada. |

En lo que respecta a **“flight-crew-member/activity-log/update”:**

|  |  |  |
| --- | --- | --- |
| **Caso de prueba** | **Tipo** | **Descripción** |
| **update.safe** | Casos positivos y negativos del formulario | Se prueba que todos los campos del formulario estén correctamente  validados, rechazando los valores no admitidos por el modelo |
| **update- url.hack** | Hack (url) | Permite probar:  -Miembros con otro rol no puedan ver los detalles de una incidencia mediante el “update”.  -Miembros con mismo rol no puedan ver incidencia de asignaciones que no le corresponden mediante el “update”.  - Miembros del rol correcto no puedan tratar de  actualizar incidencias ya  publicadas. |
| **update- values.hack** | Hack (values) | Permite probar que:  -Atributos read-only (una fecha) no puedan ser modificados. |

En lo que respecta a **“flight-crew-member/ activity-log /publish” :**

|  |  |  |
| --- | --- | --- |
| **Caso de prueba** | **Tipo** | **Descripción** |
| **publish.safe (para en y es)** | Casos positivos y negativos del formulario | Se prueba que todos los campos del formulario estén correctamente  validados, rechazando los valores no admitidos por el modelo (principalmente la longitud al ser los campos  String). |
| **publish- url.hack** | Hack (url) | Permite probar:  -Miembros con otro rol no puedan ver los detalles de una incidencia mediante el “publish”.  -Miembros con mismo rol no puedan ver incidencia de asignaciones que no le corresponden mediante el “publish.”  - Miembros del rol correcto no puedan tratar de  publicar incidencias ya  publicadas. |
| **publish - values.hack** | Hack (values) | Permite probar que:  -Atributos read-only (una fecha) no puedan ser modificados. |

Por último, para **“flight-crew-member/ activity-log /delete” :**

|  |  |  |
| --- | --- | --- |
| **Caso de prueba** | **Tipo** | **Descripción** |
| **delete.safe (para en y es)** | Casos positivos y  negativos del formulario | Se prueba que una incidencia se borre  correctamente bien al incluir valores correctos  o incorrectos. |
| **delete- url.hack** | Hack (url) | Permite probar:  -Miembros con otro rol no puedan acceder a los detalles de la incidencia por medio del “delete”.  - Miembros del mismo rol no puedan acceder a los detalles de una  incidencia que no les pertenece por medio del “delete”.  - Incidencias ya  publicadas no puedan ser borradas. |
| **delete- values.hack** | Hack (values) | Permite probar que:  -Atributos read-only (una fecha) no puedan ser modificados (se debe borrar la incidencia  aunque se modifique  este campo). |

## Calidad de la cobertura

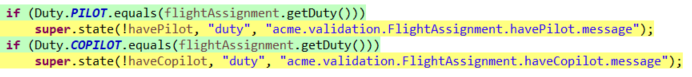
En esta sección del documento se pretende describir la calidad de la cobertura para los servicios de las entidades FlightAssignment y ActivityLog. Tanto para el listado como para los detalles, se alcanza un 100% de cobertura (menos para el listado de los incidentes, con un 99%), lo que significa que se cubren todas y cada una de las líneas de código.

Para los servicios de CRUD, la cobertura gira alrededor del 94-98% debido a los métodos de validación.

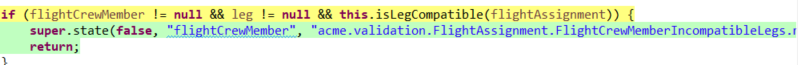
En algunos casos relativos a las acciones CRUD (tanto de Flight Assignment como en ActivityLog, no se da el 100% de cobertura debido a las siguientes líneas:



* En este caso “authorised2” se inicializa como true y sólo cambia si una leg no existe en el repositorio, no se ha podido replicar este error en el testeo pero creemos que no presenta ningún fallo en la aplicación.



* Se desconoce el motivo de esta línea en amarillo ya que durante el testeo podemos ver el mensaje de validación asociado a la misma.



* Se piensa que el motivo de esta última línea es alguna parte del paréntesis que no se llega a acceder, de nuevo el funcionamiento de la validación se ha comprobado y es correcto.

En el caso especial de Delete ocurre lo siguiente:

Texto

El contenido generado por IA puede ser incorrecto.

* Que se debe a que al borrar no se accede al método unbind ya que no hace falta guardar ningún dato.

Por estos motivos alcanzar el 100% de cobertura en todos los servicios es prácticamente imposible, o por lo menos inviable.

## Conclusiones finales

En definitiva, se han desarrollado un conjunto de pruebas que cubre la gran mayoría de líneas de código de las entidades del estudiante tres. Se ha cumplido en gran medida el objetivo de obtener un sistema robusto y seguro ante la gran mayoría de escenarios que se dan en el mundo real.

# Testing de Rendimiento

## Entorno y protocolo

Para el desarrollo de esta sección se va a emplear un equipo con las siguientes características:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Modelo** | **RAM** | **CPU** | **GPU** | **Disco** | **SO** |
| Victus By HP Gaming Laptop 16-s0xxx | 32.0  GB | AMD Ryzen 7  7840HS w/ Radeon 780M Graphics | Nvidia RTX  4060 8 GB | 954  GB | Windows 11  Home |

Para este estudio se va a emplear únicamente los casos de prueba que he ejecutado individualmente como estudiante tres, empleando la versión C2 del proyecto y del framework la 25.6.0. En el primer lanzamiento, no se emplearán índices en mis entidades, para ver si el rendimiento mejora una vez se implementen.

## Recogida de datos

Para la obtención de datos, tanto antes como después de implementar índices, se han seguido los pasos explicados en la teoría, analizando la información del “.trace” generado por la aplicación y filtrándola para obtener un fichero limpio del que poder obtener información útil como gráficos.

## Estadísticas Descriptivas

Una vez filtradas las peticiones realizadas en los casos de prueba eliminando las irrelevantes, podemos afirmar que, para el estudio, se utiliza un total de 1398 filas de datos. En ambos casos se presenta una media y desviación similares, de los que se pueden derivar, además, los siguientes datos:

Tabla

El contenido generado por IA puede ser incorrecto.

Como podemos observar, las trazas “antes” y “después” presentan medias de 7.8 y 8.6 y dispersiones alrededor de 10 prácticamente iguales, por lo que en un principio la incorporación de índices no introduce cambios significativos en el rendimiento.

Esto se puede observar también en los intervalos de confianza, que se solapan totalmente. Sin embargo, para ambas trazas se cumple con creces el requisito de rendimiento, ya que los límites superiores son ambos menores a 1 segundo (nuestro requisito base).

Por tanto, una de las conclusiones que podemos sacar es que este equipo ejecuta los casos de prueba de una forma muy rápida. Así, aunque implementemos pequeñas mejoras como los índices, verdaderamente no se va a notar en el tiempo empleado, puesto que de por sí el tiempo base es muy pequeño.

## Gráficos de eficiencia por característica

Uno de los aspectos que más nos interesa del rendimiento, es comprobar cuál es la petición más ineficiente (MIR), y ver si los índices suponen una mejora real en el rendimiento de esta.

Un conjunto de letras blancas en un fondo blanco

El contenido generado por IA puede ser incorrecto.

*Ilustración 3: Gráfico de eficiencia de la traza sin mejorar*

Un conjunto de letras blancas en un fondo blanco

El contenido generado por IA puede ser incorrecto.

*Ilustración 4: Gráfico de eficiencia de la traza mejorada*

Claramente hay una petición mucho más ineficiente que el resto, **“activity-log/list”**.

## Hipótesis y conclusión

Para finalizar con las pruebas de rendimiento, vamos a concluir este informe con una decisión firme sobre los resultados de las dos trazas generadas. Tras comparar el tiempo de las peticiones realizadas en ambas, hemos obtenido los siguientes datos:

Aplicación, Tabla

El contenido generado por IA puede ser incorrecto.

*Ilustración 5: Prueba z para muestras de dos medias*

El tiempo que realmente nos interesa de estos datos es el **valor “p” de dos colas (≈ 0.08)**. Sabiendo que **“α = 1 – nivel de confianza”**, nos queda que este valor “**p > α > 1 – 0.8 -> p > 0.02”**. Como este valor se sitúa a la derecha del intervalo [0, α], podemos afirmar aquello que hemos estado comentado en este informe, y es que verdaderamente el implementar índices no ha supuesto un cambio significativo.

En definitiva, hubiera sido mucho más interesante poder realizar estas secciones con un equipo que tarde más tiempo en ejecutar los casos de prueba, puesto que, al ser el tiempo tan pequeño, al final la varianza en el mismo se va a ver influida generalmente por el ruido en el equipo, y no por las mejoras que se hayan podido implementar en la aplicación. Sin embargo, creía interesante realizar el estudio completo, ya que permite comprender mucho más a fondo cómo funciona la lógica dentro de la aplicación.

# Testing de Mutaciones

En esta sección del documento se pretende describir las mutaciones realizadas para cumplir con el **requisito 22** del estudiante tres, e informar sobre los resultados, describiendo los casos de prueba que los invalidan.

## Primera mutación: Intercambio de valores

Se intercambian los valores del dataset en el **unbind** de “**FlightCrewMemberAssignmentShowService”**. Es decir:

Texto

El contenido generado por IA puede ser incorrecto.

*Ilustración c: Variables declaradas correctamente*

*Texto

El contenido generado por IA puede ser incorrecto.*

*Ilustración 7: Variables intercambiadas en los datasets*

Este mutante es invalidado por todos los casos de prueba “.safe” de FlightAssignment, ya que en todos ellos se realiza al menos una petición para mostrar los detalles, y, al recuperar información diferente a la grabada por el cambio en el código, salta error al intentar hacer la petición “show”.

Además, este mutante también es combatido en todos los casos “.safe” de ActivityLog, ya que, para poder realizar cualquier acción sobre dicha entidad, es necesario pasar siempre por los detalles de un FlightAssignment, salvo en casos excepcionales donde se tenga la url directa al recurso y se pueda copiar directamente en la navegación.

## Segunda mutación: Eliminar línea de código en validación

Se elimina la primera validación obligatoria del servicio de creación de una asignación (**FlightCrewMemberAssignmentCreateService**). Es decir:

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

*Ilustración 8; Método "validate" con la primera restricción eliminada*

Este mutante es invalidado principalmente en el caso de prueba **“flight-crew- member/flight-assignment/create”**, ya que, entre otras comprobaciones, se prueba la restricción comentada, de forma que se espera el mensaje de error: “**You can't select a leg that hasn't been published”**. Sin

embargo, al comentarse, se permite crear una asignación con un Leg que no se encuentra con DraftMode (Modo Borrador) en False. Cabe resaltar que esta excepción no se tuvo encuentra en la última entrega y fue un caso de suspenso. Podemos observar que ya ha sido corregido.

Este mutante también se combate en otros muchos casos de prueba porque generalmente se utiliza el miembro sin asignaciones para probar los servicios, por lo que casi siempre es obligatorio pasar a través del servicio de creación.

## Tercera mutación: Cambiar lógica de authorise

Se invierte la condición del authorise en el servicio de listado de ActivityLog (**FlightCrewMemberActivityLogListService**) sobre si la asignación de vuelo corresponde al FlightCrewMember logueado inicialmente. Es decir:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

*Ilustración S: Comparativa de lógica original vs invertida*

Esta mutación es detectada inmediatamente por el caso de prueba flight-crew-member/activity-log/create-en.safe. Esto se debe a que, para acceder al formulario de creación de incidentes, primero se debe listar los incidentes asociados a una asignación válida y ya finalizada.

Al haberse mutado la lógica y ya no validar correctamente la pertenencia ni el estado de la asignación, el servicio intenta listar incidentes sobre una asignación inválida (por ejemplo, aún activa o perteneciente a otro usuario), lo que produce un error de autorización interna (500 Internal Server Error).

## Cuarta mutación: Eliminar borrado de logs asociados a la asignación

En esta mutación, se modifica el comportamiento del servicio de borrado de una asignación (**FlightCrewMemberActivityLogDeleteService**) al **comentar la línea que elimina los registros de actividad** (ActivityLog) vinculados a la asignación de vuelo (FlightAssignment).

Texto

El contenido generado por IA puede ser incorrecto.

*Ilustración 10: Línea de codigo comentada que produce el mutante*

Este cambio provoca que **los registros de actividad (ActivityLog) asociados a una asignación no se eliminen** del sistema cuando se borra la asignación principal. En otras palabras, la asignación se elimina, pero los datos relacionados con ella **quedan huérfanos** en la base de datos.

## Quinta mutación: Alterar operador de consulta

Se cambia el operador de la consulta **“thatFlightAssignmentIsOf”** del repositorio **“FlightCrewMemberAssignmentRepository”** para que el scheduledDeparture sea solo **“<”** en vez de **“<=”**. Así:



*Ilustración 11: Consulta actualizada que reffeja el mutante*

Esta query retorna true si **existe al menos una asignación de vuelo** (FlightAssignment) con el ID dado, **y** que esté asignada al miembro de tripulación con el ID correspondiente. Es decir:

* count(fa) > 0 ⇒ devuelve true si hay coincidencias.
* Útil para comprobar si **una asignación pertenece a un tripulante específico**.

Este pequeño cambio provocará lo siguiente:  
Siempre devolverá **false**, porque:

* count(fa) **nunca será menor que 0**.
* count retorna un número entero ≥ 0 (nunca negativo).
* Por tanto, **count(fa) < 0 será siempre false**, aunque la asignación exista y pertenezca al tripulante.