**Informe de mutaciones**

* **Proyecto: Acme-ANS**
* **Repositorio:** <https://github.com/AngSanRui/Acme-ANS-C2>
* **Responsable: Ángel Sánchez Ruiz (angsanrui2@alum.us.es)**
* **Grupo: C2.056**

**Fecha: 28/06/2025**

**Índice:**

1 Resumen ejecutivo 2

2 Introducción 2

3 Mutaciones introducidas y resultados 2

3.1 Seguridad 2

3.2 Mantenibilidad 3

3.3 Código duplicado 4

3.4 Seguridad 5

3.5 Mantenibilidad 6

4 Conclusiones 6

|  |  |  |
| --- | --- | --- |
| Versión | Fecha | Cambios |
| 1.0 | 28/06/2025 | Versión Inicial |
| 1.1 | 30/06/2025 | Añadido resumen ejecutivo |

**1-Resumen ejecutivo**

Este informe documenta exclusivamente el proceso de testing mediante mutaciones de código aplicado a las features de flightAssignment y activityLog.

En total cinco mutaciones fueron introducidas incluyendo cambios en la lógica, errores al asignar variables, cambios en queries, y omisión de una instrucción de código.

Los resultados muestran que la suite de pruebas detecta correctamente las mutaciones introducidas, lo que indica un nivel aceptable de cobertura de código. También nos muestra el valor que tiene usar mutaciones a la hora de testear para complementar otros análisis de cobertura.

**2-Introducción**

Las mutaciones en el código son bugs introducidos de forma intencional para comprobar cómo de buena es una test suite. Si una test suite es buena, al ejecutarla deberán producirse errores causados por esas mismas mutaciones, de lo contrario, significará que la test suite no cubre casos en los que esas partes del código con importantes y habría que mejorarla. El concepto de mutación es cercano al de cobertura, ya que se usan para medirla y asegurarse de que haya suficientes test cases.

**3-Mutaciones introducidas y resultados**

Las mutaciones que voy a introducir van a centrarse en las features de activityLog y de flightAssignment, pues es de lo que el student 3 se encarga principalmente en los requisitos individuales.

Previo a introducir las mutaciones, los tests han dado resultados positivos a excepción de al poblar la base de datos ya que se incumple una validación del tracking-log-40 por parte de otro student en sus requisitos individuales, que a momento de realizar el informe no se ha corregido. De nuevo, fuera de eso, los tests de flightAssignment y activityLog no resultan en ningún error.

**3.1-FlightAssignment Create Service**

En el create service de flightAssignment he decidido cambiar una parte de la validación, una que se asegura de que el tramo existe. Adjunto captura de pantalla:

A screen shot of a computer code

AI-generated content may be incorrect.

En la línea 114 se ha pasado de “**legExists = leg != null;**” a “**legExists = leg == null;**”(la captura es tras el cambio), de forma que al crear flight assignments, si los tests cubren correctamente esa parte del código, debería resultar en fallos.

Efectivamente, han saltado fallos grandes:

A close up of text

AI-generated content may be incorrect.

Estos fallos han aparecido en los tests de:

* create-flightAssignment.safe
* delete-flightAssignment.safe
* update-publish-flightAssignment.safe

**3.2-Completed Flight Assignment List Service**

En el servicio para mostrar los flight assignments completados, he cambiado en la función load() la línea 59. Ha pasado de:

**memberIsAvailable = member.getAvailabilityStatus() == AvailabilityStatus.*ON\_VACATION*;**

a

**memberIsAvailable = member.getAvailabilityStatus() == AvailabilityStatus.AVAILABLE;**

cambiando el estado de disponibilidad que tiene el crew member.A screenshot of a computer code

AI-generated content may be incorrect.

Esto hace que cambie el boolean memberIsAvailable, que se usa en el list.jsp de flightAssignment.

Esta mutación ha generado fallos en:

* create-activityLog.safe
* create-flightAssignment.safe
* delete-activityLog.safe
* delete-flightAssignment.safe
* list-show-activityLog.safe
* list-show- flightAssignment.safe
* update-publish-activityLog.safe
* update-publish-flightAssignment.safe
* create-activityLog.hack
* create-flightAssignment.hack
* delete-activityLog.hack
* update-publish-activityLog.hack
* update-publish-flightAssignment.hack

En los que ha causado un impacto mayor, es decir, tiene más fallos, es en update-publish-flightAssignment.hack.

**3.3-FlightCrewMember validator**

La siguiente mutación consiste en intercambiar dos variables del mismo tipo.

En el validador de flightCrewMember, he intercambiado el string phoneNumber por el string initials. Así quedaría después de los cambios:

A screen shot of a computer

AI-generated content may be incorrect.

Esto seguramente introduzca muchos problemas relacionados con la validación al poblar la base de datos.

Tras ejecutar los tests, efectivamente ha dado bastantes errores de validación tanto en crew members como en assignments como en activityLogs. Además, también ha causado otro tipo de fallos(como fallos de get o post) en los siguientes tests:

* create-activityLog.safe
* create-flightAssignment.safe
* delete-flightAssignment.safe
* update-publish-activityLog.safe
* update-publish-flightAssignment.safe
* create-activityLog.hack
* create-flightAssignment.hack
* delete-activityLog.hack
* update-publish-activityLog.hack
* update-publish-flightAssignment.hack

Esta mutación no ha afectado en ningún caso a los tests de list-show

**3.4- FlightAssignment repository**

En la siguiente mutación haré un cambio en el repositorio de activityLog entre un “<” y un “>”. Concretamente, es de la query findSimultaneousLegs. Tras el cambio, la query quedaría así:

@Query("select f from FlightAssignment f where f.id != :assignmentId and f.flightCrewMember.id = :crewMemberId and f.leg.id != :legId and f.leg.scheduledDeparture **>** :scheduledArrival and f.leg.scheduledArrival **<** :scheduledDeparture")

Collection<FlightAssignment> findSimultaneousLegs(int assignmentId, int legId, Date scheduledDeparture, Date scheduledArrival, int crewMemberId);

Al introducir esta mutación, han fallado los siguientes tests:

* create-flightAssignment.hack
* update-publish-flightAssignment.hack

**3.5-ActivityLog update service**

Esta vez, la mutación que voy a introducir al código consiste en comentar una instrucción. La línea que voy a comentar pertenece al perform del update service de activityLog. Es esta:

A screen shot of a computer

AI-generated content may be incorrect.

La línea actualiza la propiedad registrationMoment del activityLog cuando de hace un update.

Al ejecutar los tests, vemos que este cambio lógicamente afecta a:

* update-publish-activityLog.safe
* update-publish-activityLog.hack

**4-Conclusiones**

Las mutaciones probadas en diferentes partes del código han generado, como se esperaba, fallos al reproducir los tests. Esto nos demuestra que aparentemente los casos de prueba no están mal. También nos demuestra que una forma de probar la cobertura (y, dependiendo de la mutación, hasta la lógica) del código es usando mutaciones.

Para diferentes proyectos se pueden probar diferentes tipos de mutaciones. Al seleccionar las mutaciones que iba a probar, me ha costado un poco ya que no he sabido ni podido aplicar algunos de los diferentes tipos que conozco.