03/07/2025

Grupo: C2.064

Repositorio: <https://github.com/paugonpae67/Acme-ANS>

Autora: María Auxiliadora Quintana Fernández (marquifer@alum.us.es)

## Mutation report: Student 4

ACME ANS – C2.064

|  |
| --- |
| TABLA DE CONTENIDOS |

[**RESUMEN EJECUTIVO** 2](#_Toc199176358)

[**TABLA DE REVISIONES** 3](#_Toc199176359)

[**INTRODUCCIÓN** 4](#_Toc199176360)

[**CONTENIDO** 5](#_Toc199176361)

[**CONCLUSIÓN** 1](#_Toc199176362)1

[**BIBLIOGRAFÍA** 12](#_Toc199176363)

|  |
| --- |
| RESUMEN EJECUTIVO |

n este apartado se documenta la realización de **cinco mutaciones deliberadas** en el código fuente del sistema, con el objetivo de verificar la **robustez y eficacia de la batería de pruebas existentes**. Esta actividad responde al Requisito 22 y se enmarca en la práctica de **mutation testing**, una técnica que consiste en introducir pequeños errores en el código (mutantes) para comprobar si los tests son capaces de detectarlos.

Las mutaciones se aplicaron sobre el código asociado a los requisitos 8 y 9 del Student 4, relacionados con las entidades Claim y TrackingLog. Estas modificaciones se hicieron manualmente, de forma controlada.

Tras cada mutación, se ejecutó el conjunto completo de pruebas funcionales automatizadas, observando si los tests **fallaban como se esperaba** (lo que indica que el test es eficaz).

Como resultado de esta evaluación, se realizaron ajustes en las pruebas para cubrir el caso no detectado, reforzando así la calidad del sistema.

Este proceso ha permitido validar la eficacia de los tests existentes, identificar zonas del código poco cubiertas, y mejorar la confiabilidad global del sistema. Se concluye que el sistema responde adecuadamente a errores inesperados.

|  |
| --- |
| TABLA DE REVISIONES |

|  |  |  |
| --- | --- | --- |
| Versión | Fecha | Descripción |
| 1.0 | 03/07/2025 | Versión Inicial. Estructura del documento y Mutation Report |

|  |  |  |
| --- | --- | --- |
| INTRODUCCIÓN |  |  |

En esta sección del informe se documenta el proceso y los resultados derivados de la realización de cinco **mutaciones deliberadas en el código fuente**, en cumplimiento del **Requisito 22** del proyecto ACME-ANS. El objetivo de esta actividad es evaluar la **robustez y efectividad de las pruebas funcionales existentes**, analizando si son capaces de detectar errores introducidos intencionadamente.

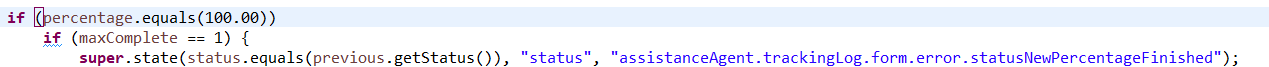
Cada mutación consiste en un pequeño cambio en el comportamiento lógico del sistema, como por ejemplo invertir condiciones, alterar retornos o modificar operadores, simulando errores comunes de programación. Estas mutaciones se aplicaron sobre las funcionalidades correspondientes a los **Requisitos 8 y 9**, específicamente sobre las entidades Claim y TrackingLog, y fueron seleccionadas estratégicamente para afectar tanto flujos normales como casos límite.

Tras aplicar cada mutación, se ejecutaron los conjuntos de pruebas previamente implementados (.safe y .hack) con el fin de observar si las pruebas fallaban como se espera, lo cual indicaría que son efectivas y sensibles ante alteraciones del comportamiento del sistema.

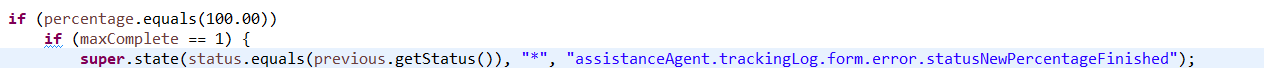
|  |  |  |
| --- | --- | --- |
| CONTENIDO |  |  |

**Mutación 1 – Alteración en la validación del campo status en AssistanceAgentUpdateTrackingLog**

Se ha realizado una mutación en la clase AssistanceAgentUpdateTrackingLog, concretamente en el método validate. Antes de aplicar la mutación, la lógica de validación generaba un error directamente asociado al campo status, permitiendo al sistema mostrar el mensaje de validación justo debajo del campo correspondiente en el formulario.



Tras la mutación, se modificó la ubicación del mensaje de error, haciendo que este se lance como un error global (a nivel de página) en lugar de estar vinculado al campo específico. Aunque el mensaje de error sigue siendo el mismo, el cambio en el contexto de aparición permite comprobar si los tests están correctamente diseñados para detectar este tipo de comportamiento.



Una vez aplicada esta mutación, se ejecutaron nuevamente los tests funcionales, observándose fallos en el archivo update-tracking-log.safe, lo cual confirma que las pruebas están correctamente capturando el comportamiento alterado del sistema. Este resultado era el esperado y demuestra que las pruebas son sensibles a cambios sutiles en la lógica de validación.

A close up of text

AI-generated content may be incorrect.

**Mutación 2 – Alteración en la autorización en AssistanceAgentPublishTrackingLog**

Se ha llevado a cabo una mutación en la clase AssistanceAgentPublishTrackingLog, específicamente en el método authorise. En su implementación original, el sistema impedía publicar un *tracking log* si la *claim* asociada no había sido publicada previamente, tal como se especifica en los requisitos.

A screenshot of a computer

AI-generated content may be incorrect.



Tras aplicar la mutación, se eliminó dicha condición, permitiendo la publicación de un *tracking log* incluso cuando la *claim* correspondiente aún no había sido publicada. Esta alteración compromete la lógica de autorización y rompe la restricción establecida por el diseño original.

A close-up of a computer screen

AI-generated content may be incorrect.



Posteriormente, se ejecutaron los tests funcionales correspondientes, observándose errores en el fichero publish-tracking-log.hack. Este comportamiento confirma que los tests están correctamente configurados para detectar desviaciones en las reglas de autorización. Como se esperaba, el fallo verificado demuestra la capacidad de las pruebas de *hacking* para identificar vulnerabilidades lógicas introducidas mediante mutaciones en el código.

A close-up of a text

AI-generated content may be incorrect.

**Mutación 3 – Alteración en el método unbind de AssistanceAgentTrackingLogCreateService**

Se realizó una mutación en la clase AssistanceAgentTrackingLogCreateService, específicamente en el método unbind. En la implementación original, el sistema incluía en el *payload* el campo description, lo que permitía que este campo no se mostrara directamente en la vista *show*, pero sí fuera utilizable como criterio de filtro en la lista.

A screenshot of a computer code

AI-generated content may be incorrect.



Tras la mutación, se eliminó la inclusión de dicho campo en el *payload*, impidiendo que el filtro por description funcionara correctamente en la lista.

A computer code with text

AI-generated content may be incorrect.



Posteriormente, se ejecutaron los tests funcionales correspondientes, detectándose errores en el fichero create-claim.hack. Este resultado confirma que los tests están adecuadamente diseñados para captar desviaciones en la forma en que los datos son leídos y manejados desde la base de datos. Como se esperaba, estos fallos demuestran la efectividad de las pruebas de *hacking* para identificar vulnerabilidades lógicas introducidas mediante mutaciones en el código.

Cabe destacar que, aunque se detectaron otros errores iguales, los mencionados representan ejemplos claros de las alteraciones provocadas.

A close up of text

AI-generated content may be incorrect.

**Mutación 4 – Alteración en el método bind de AssistanceAgentUpdateTrackingLog**

Se llevó a cabo una mutación en la clase AssistanceAgentUpdateTrackingLog, específicamente en el método bind. En la implementación original, este método recogía varios campos del formulario, entre ellos el campo resolutionPercentage.

A computer code on a white background

AI-generated content may be incorrect.

Tras aplicar la mutación, se eliminó la inclusión del campo *resolutionPercentage*.

A computer screen shot of a program

AI-generated content may be incorrect.

Posteriormente, se ejecutaron los tests funcionales correspondientes, detectándose errores en el fichero *update-tracking-log.hack*. Este resultado confirma que los tests están correctamente diseñados para captar desviaciones en la forma en que los datos son leídos y gestionados desde la interfaz web. Como se esperaba, estos fallos demuestran la efectividad de las pruebas de hacking para identificar vulnerabilidades lógicas introducidas mediante mutaciones en el código.

Cabe destacar que, aunque se detectaron otros errores similares, los mencionados representan ejemplos claros de las alteraciones provocadas.

A close up of a text

AI-generated content may be incorrect.

**Mutación 5 – Alteración en un atributo de la entidad claim**

Se llevó a cabo una mutación en la clase *Claim*, específicamente en el atributo *passengerEmail*. Este atributo, tal como su nombre indica, almacenaba una dirección de correo electrónico y contaba con la validación asociada *@ValidEmail*.

A close-up of a white background

AI-generated content may be incorrect.

Tras aplicar la mutación, se eliminó la validación *@ValidEmail* de dicho atributo.

A close-up of a computer screen

AI-generated content may be incorrect.

Posteriormente, se ejecutaron los tests funcionales correspondientes, detectándose errores en todos los ficheros de test relacionados. Este resultado confirma que los tests están correctamente diseñados para detectar desviaciones en la validación de datos. Como era de esperar, estos fallos demuestran la efectividad de las pruebas de hacking para identificar vulnerabilidades lógicas introducidas mediante mutaciones en el código.

Cabe destacar que, aunque se detectaron otros errores similares, los mencionados representan ejemplos claros de las alteraciones provocadas.

A close up of a text

AI-generated content may be incorrect.

|  |  |  |
| --- | --- | --- |
| CONClUSIÓN |  |  |

Este informe ha documentado las mutaciones realizadas sobre distintas partes del código relacionadas con los requisitos 8 y 9, específicamente en las clases *AssistanceAgentUpdateTrackingLog*, *AssistanceAgentPublishTrackingLog*, *AssistanceAgentTrackingLogCreateService* y la entidad *Claim*. Se aplicaron cinco mutaciones que introdujeron modificaciones intencionadas en la lógica de autorización, validación y manipulación de datos, con el fin de evaluar la sensibilidad y eficacia de las pruebas funcionales y de hacking implementadas.

Los resultados mostraron que las pruebas fueron capaces de detectar correctamente las desviaciones y vulnerabilidades introducidas, manifestándose en errores esperados en los ficheros de test correspondientes (.safe y .hack). Esto confirma que el conjunto de pruebas está bien diseñado para identificar fallos lógicos y de validación, así como para garantizar la robustez del sistema ante cambios inesperados en el código.

En definitiva, las mutaciones permitieron verificar la capacidad de las pruebas para capturar errores sutiles y validar la integridad de la funcionalidad, demostrando que las estrategias de testing aplicadas son adecuadas y efectivas para mantener la calidad y seguridad del sistema.

|  |  |  |
| --- | --- | --- |
| BIBLIOGRAFÍA  Intentionally blank |  |  |