# Testing Report

Grupo C1.054 | Diseño y Pruebas II | 26/05/2025

Versiones

|  |  |  |
| --- | --- | --- |
| Fecha | Versión | Descripción |
| 26/05/2025 | 1.0 | Creación plantilla |
| 26/05/2025 | 2.0 | Testing Report completo |

Miembros:

Adrián Díaz Vázquez ([adrdiavaz@alum.us.es](mailto:adrdiavaz@alum.us.es))

Esteban López Pérez ([estlopper@alum.us.es](mailto:estlopper@alum.us.es)) - Autor

Jesús Pons Morís ([jesponmor@alum.us.es](mailto:jesponmor@alum.us.es))

Kevin Amador Calzadilla ([kevamacal@alum.us.es](mailto:kevamacal@alum.us.es))

Manuel Zurita Fernández ([manzurfer@alum.us.es](mailto:manzurfer@alum.us.es))

Repositorio: <https://github.com/DP2-C1-054/Acme-ANS-D04>

# Índice:

Resumen ejecutivo………………………………………………………………………3

Introducción……………………………………………………………………………..3

Contenido………………………………………………………………………………..3

Pruebas funcionales……………………………………………………………...3

Análisis de rendimiento……………………………………………………….....7

Conclusiones…………………………………………………………………………......9

Bibliografía………………………………………………………………………………9

# Resumen ejecutivo

Este informe recoge una descripción completa de las pruebas funcionales realizadas sobre los servicios correspondientes al estudiante 4, junto con un análisis de rendimiento. El propósito principal es verificar que las funcionalidades implementadas cumplen con los requisitos establecidos y garantizan el cumplimiento de las restricciones de seguridad frente a posibles ataques de hacking.

# Introducción

Se presentan de forma detallada de las pruebas funcionales realizadas sobre los servicios asociados al estudiante 4, así como un análisis de rendimiento enfocado en dichas pruebas. El objetivo principal es verificar que las funcionalidades implementadas cumplen correctamente con los requisitos especificados y garantizan la seguridad de los datos del usuario.

Para ello, se han llevado a cabo dos tipos de pruebas funcionales, centradas en las entidades Claim y TrackingLog, que corresponden a las funcionalidades principales del estudiante 4. En primer lugar, se ha realizado un testing formal con casos positivos y negativos, cubriendo todos los valores permitidos y no permitidos en los formularios disponibles, con el fin de comprobar que las respuestas del sistema son correctas y coherentes. En segundo lugar, se han ejecutado pruebas de hacking orientadas a asegurar que la seguridad del usuario y la integridad de sus datos no puedan ser vulneradas ni manipuladas.

El análisis de rendimiento se ha desarrollado bajo dos escenarios diferenciados: uno sin utilizar índices para la optimización de consultas y otro donde sí se emplean dichos índices. Se han ejecutado las pruebas funcionales en ambos escenarios para comparar los resultados y evaluar el impacto de los índices en el rendimiento del sistema.

# Contenido

## Pruebas funcionales

Se han realizado las siguientes pruebas funcionales para la entidad “Claim” :

**List&list-pending.safe**

Para esta prueba se ha verificado que se listen correctamente todas las reclamaciones del agent1 con el que nos logueamos.

**Show.safe**

En este caso se verifica que los detalles de la reclamación se muestran correctamente para el agent1.

**Create.safe**

Para esta prueba accedemos como agent1 al sistema hasta el formulario de creación de una Claim. Para verificar que todo funciona como debe, en primer lugar, hemos enviado el formulario vacío. Después por cada uno de los campos (dejando el resto con valores vacíos) hemos probado valores positivos y negativos según el contexto de prueba.

Finalmente, tras verificar que los valores válidos el sistema los interpreta correctamente y los valores negativos los rechaza, hemos enviado un formulario correcto verificando que se cree el objeto como debe.

**Delete.safe**

En esta prueba comprobamos que el agent1 pueda eliminar sus reclamaciones independientemente de los valores que presente en su formulario.

**Update.safe**

De forma similar a la prueba de creación, accedemos como agent1 hasta una reclamación cualquiera, dejamos todos los campos de su formulario vacíos y realizamos el mismo procedimiento de verificación de casos positivos y negativos antes comentado.

Finalmente comprobamos que se actualicen correctamente los valores positivos de una reclamación del agent1.

**Publish.safe**

Aquí se realiza el mismo procedimiento de pruebas que en el caso de update.safe, pero probando a publicar una reclamación en lugar de actualizarla.

**Show.hack**

En este primer test de hacking se verifica que una reclamación del agent1 no se muestre al agent2, a cualquier otro usuario logueado en el sistema o a un usuario anónimo, así como una inyectando un id no válido en esas consultas. Se verifica que salte un error de autorización.

**Create.hack**

Aquí probamos a intentar hackear los atributos de navegación del formulario de una reclamación. Este atributo se trata de una leg o tramo de vuelo, y lo que tratamos de hacer en inyectar un id negativo, un id de un tramo de vuelo que no esté publicado o un id de un tramo que esté programado en el futuro. Para todos los casos salta el error de autorización que debe saltar.

Además, se ha comprobado que no se pueda acceder a la url de creación desde un usuario que no sea un agente.

**Delete.hack**

En esta prueba se intentan dos cosas, ambas de ellas dan como resultado un error de autorización. En primer lugar, accediendo como agent1 intentamos eliminar una reclamación ya publicada. En segundo lugar, tratamos de eliminarla, pero accediendo como un usuario anónimo, como agent2, y como otro tipo de usuario logueado en el sistema.

**Update.hack**

Aquí hemos tratado de actualizar los formularios introduciendo valores en los atributos de navegación que deben hacer saltar un error de autorización, es decir, lo mismo que se detalla con respecto a los tramos de vuelo en el apartado “create.hack” . En segundo lugar, hemos tratado de actualizar una reclamación ya publicada o accediendo con cualquier otro usuario que no corresponda con el agent1.

**Publish.hack**

En es prueba se ha realizado lo mismo que en el update pero con el método publish.

Con las pruebas antes descritas conseguimos la siguiente cobertura:

A graph of a number of words

AI-generated content may be incorrect.

Para la segunda funcionalidad testeada, “TrackingLog”, el procedimiento de pruebas realizado ha sido similar al descrito anteriormente. Se detallan a continuación dichas pruebas:

**List&show.safe**

Al igual que con Claim, comprobamos que se listen los registros de reclamación y se muestren sus detalles correspondientes para el agent1.

**Create.safe**

Aquí tratamos de probar los valores positivos y negativos en el formulario de creación del registro de reclamación del agent1 y verificamos que se cree correctamente.

**Delete.safe**

En esta prueba comprobamos que el agent1 pueda eliminar sus registros de reclamaciones independientemente de los valores que presente en su formulario.

**Update.safe**

Al igual que el create.safe, tratamos de actualizar un registro de reclamación del agent1 con un formulario vacío y con casos positivos y negativos, siguiendo el procedimiento detallado anteriormente.

**Publish.safe**

El procedimiento es el mismo que para el update.safe pero probando el método de publicación.

**List.hack**

En esta prueba se trata de probar que el listado de registros de una reclamación específica para el agent1 no se muestre para un usuario anónimo, para el agent2 o para otro usuario cualquiera, y que el error de autorización salte correctamente.

**Show.hack**

En esta prueba verificamos que salte un error de autorización al intentar mostrar un registro de reclamación del agent1 en otro usuario cualquiera.

**Create.hack**

Como estos formularios no cuentan con atributos de navegación tratamos de intentar enviar un valor inválido en los campos enumerados y verificamos que salte el error de inválido.

**Delete.hack**

En esta prueba tratamos de eliminar un registro de reclamación publicado o accediendo desde otro usuario que no sea el agent1.

**Update.hack**

Al igual que en delete.hack, probamos lo mismo con el campo update. Además, tratamos de enviar un valor inválido a través del atributo enumerado del formulario.

**Publish.hack**

En esta prueba se hace lo mismo que en el update.hack pero con el método publish.

Con las pruebas antes descritas conseguimos la siguiente cobertura:

A screenshot of a computer

AI-generated content may be incorrect.

## Análisis de rendimiento

El siguiente análisis se ha realizado en dos escenarios. Se ejecutan las pruebas anteriores y se comparan los resultados para un escenario donde no se utilizan índices usados para la optimización y un segundo escenario donde estos índices sí son usados.

Se muestran los resultados promedio de rendimiento de las pruebas realizadas sin índices:

Se muestran los resultados promedio de rendimiento de las pruebas realizadas con índices, se observa claramente que los valores disminuyen, por lo que mejoramos en términos de tiempo:

Podemos observar que los valores promedios disminuyen tras aplicar los índices:

A screenshot of a data sheet

AI-generated content may be incorrect.

Tras aplicar el z-Test se obtienen los siguientes resultados:

A screenshot of a data sheet

AI-generated content may be incorrect.

Observamos que el valor P es mayor a 0.05 por lo que concluimos que la adición de índices no supone como tal una mejora en el rendimiento del sistema, aunque los valores promedios calculados anteriormente hayan disminuido, entonces las diferencias en las estadísticas de las tablas previas realmente no hacen gran efecto en el sistema.

# Conclusiones

El conjunto de pruebas funcionales aplicadas nos permite encontrar fallos, bugs y vulnerabilidades en nuestro sistema, lo que nos ha permitido corregir y analizar en una fase previa a la entrega del producto al usuario final. De la misma manera, nos permiten asegurar los datos del usuario y cumplir con los estándares de calidad del software y la Ley Orgánica 3/2018.

La realización del análisis de rendimiento de dichas pruebas nos permite encontrar aquellas zonas más conflictivas de nuestro sistema para poder aplicar refactorizaciones y mejorar el rendimiento de nuestra aplicación. Técnicas como el Software Profiling nos puede indicar aquellos métodos que más tiempo requieran o los análisis de hardware para encontrar posibles cuellos de botella.

En conclusión, aplicar estas técnicas permiten definir un software robusto y entregar un producto de calidad.

# Bibliografía

Intencionadamente en blanco.