Testing Report

* **Group Number**: C1.047
* **Repository**: <https://github.com/JoaquinBorjaLeon/C1-047-Acme-ANS-D04>
* Workgroup Members and Corporate Emails:

Joaquín Borja León: [joaborleo@alum.us.es](mailto:joaborleo@alum.us.es) Ariel Escobar Capilla: [ariesccap@alum.us.es](mailto:ariesccap@alum.us.es) Héctor Guerra Prada: [hecguepra@alum.us.es](mailto:hecguepra@alum.us.es)

Juan Carlos León Madroñal: [jualeomad@alum.us.es](mailto:jualeomad@alum.us.es) **José Ángel Rodríguez Durán:** [**josroddur@alum.us.es**](mailto:josroddur@alum.us.es)

* **Date**: 26/05/2025

## Tabla de contenido

### Contenido

1. [Executive Summary 1](#_bookmark0)
2. [Revision Table 2](#_bookmark1)
3. [Contents 2](#_bookmark2)
4. [Performance testing 4](#_bookmark3)
5. [Software Profiling 8](#_bookmark4)
6. [Hardware Profiling 9](#_bookmark5)

# Executive Summary

En este informe se presentarán las pruebas tanto funcionales como de rendimiento que se llevaron a cabo acerca de mis requisitos como estudiante 5. El objetivo de dichas pruebas se centró en asegurar que todas las funcionalidades implementadas funcionarían según lo estipulado en los requisitos, además de evaluar el rendimiento del sistema en las condiciones normales de uso.

Para las pruebas funcionales, los casos testeados se organizaron por funcionalidad y legalidad. Cada caso se centra en una función específica, de forma que una de las pruebas compruebe lo que se espera que el usuario haga y la otra compruebe que funciona la seguridad ante usuarios que intenten saltarse las normas que impone la página web. Esto es esencial para chequear que la aplicación se comporta como quiere el cliente.

Para las pruebas de rendimiento, seguí la metodología indicada en la guía de la sesión: recopilé los tiempos de ejecución a partir de los archivos .trace y procesé los datos utilizando Excel. Generé gráficos y calculé intervalos de confianza del 95 % para evaluar si los tiempos de respuesta del sistema se mantenían dentro de los límites que el cliente consideraba aceptables. Las pruebas se ejecutaron en dos configuraciones distintas: una utilizando la base de datos sin ningún índice adicional, y otra con los índices relevantes aplicados. Posteriormente, se llevó a cabo una comparación estadística entre ambas configuraciones para determinar el impacto del uso de índices en el rendimiento.

Resumiendo, este informe refleja las pruebas que se han realizado sobre las funcionalidades asignadas al estudiante 5 dentro de los requisitos del proyecto, con los datos de su rendimiento que ofrecen una comprensión sólida del comportamiento del sistema en condiciones reales.

# Revision Table

|  |  |  |
| --- | --- | --- |
| **Revision Number** | **Date** | **Description** |
| 1.0 | [26/05/2025] | Initial version |

# Contents

A continuación, se presenta la lista de pruebas funcionales realizadas para las funcionalidades del requisito 8 del estudiante 5, acerca de los registros de mantenimiento:

* + Clase maintenanceRecord:
    - **List.safe:** Verifica que los técnicos puedan ver todos los registros de mantenimiento creados por y/o para él. Asegura la recuperación y visualización correcta de los registros pertenecientes al técnico autenticado.
    - **List.hack**: Se simulan intentos de listar los registros creados por y/o para un usuario que carece del rol de técnico.
    - **Show.safe:** Garantiza que los detalles de un registro específico se muestren correctamente, siempre que dicho registro pertenezca al usuario.
    - **Show.hack**: Se prueban intentos de acceso no autorizado a los detalles de los registros mediante la manipulación de URLs con IDs inválidos o de registros que pertenecen a otros usuarios, así como accesos de usuarios que carecen del rol de técnico.
    - **Create.safe:** Prueba la creación de un nuevo registro utilizando datos válidos. También se evalúan los mecanismos de validación del sistema al introducir campos vacíos o datos inválidos.
    - **Create.hack**: Simula el uso indebido de herramientas de desarrollo del navegador para alterar manualmente datos del formulario y forzar la creación de registros con datos que no pueden ser introducidos de manera convencional.
    - **Update.safe:** Verifica que un registro pueda modificarse correctamente mientras esté en modo borrador utilizando entradas válidas, así como que los mecanismos de validación del sistema funcionen correctamente al introducir campos vacíos o datos inválidos.
    - **Update.hack**: Comprueba que el sistema no autorice la modificación de registros que no estén en modo borrador, que no existan o que pertenezcan a otros usuarios, además de que no autorice el acceso a ningún registro si el usuario carece del rol de técnico, manipulando las URLs. Además, comprueba que no autorice la introducción de datos imposibles de introducir de manera convencional en los cuerpos de las solicitudes.
    - **Publish.safe:** Verifica que un registro pueda dejar de estar en modo borrador de manera correcta, comprobando que cumple los requisitos estipulados para ello.
    - **Publish.hack**: Verifica que no sea posible publicar registros ajenos, ya sean de otro técnico o por un usuario que carece del rol de técnico, o registros ya publicados, mediante la modificación de las rutas en las solicitudes.
    - **Delete.safe**: Comprueba que un registro de mantenimiento puede eliminarse, siempre que este si aún no haya sido publicado.
    - **Delete.hack**: Verifica que el sistema impida eliminar registros ya publicados o que no pertenezcan al usuario, modificando el ID en la URL. Además, verifica que un usuario que carezca del rol de técnico pueda eliminar un registro mediante la modificación de URLs.

La cobertura de test alcanzada de la entidad MaintenanceRecord ha sido del 94.2 por ciento

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

###### Vemos que todas las funcionalidades cumplen con el objetivo de una cobertura superior al 90% a excepción de la de eliminado, que por diseño necesita de una función *unbind* que realmente no usa.

* + Clase maintenanceRecordTask:

###### **List.safe:** Verifica que los técnicos puedan ver todas las tareas pertenecientes a un registro de mantenimiento asignado al mismo.

###### **List.**hack: Verifica que un técnico no pueda acceder a la lista de tareas de un registro de mantenimiento no asignado al mismo, y que un usuario que carezca del rol de técnico no pueda acceder a la lista de tareas de ningún registro de mantenimiento.

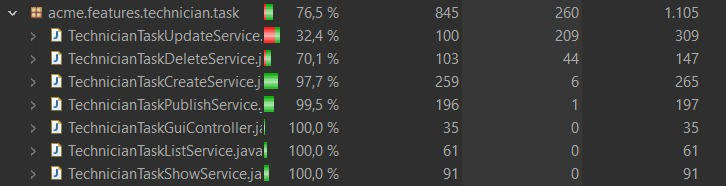
* + - **Show.safe**: Verifica que un técnico pueda visualizar los Ids de cualquier tarea y cualquier registro de mantenimiento asignados entre sí, siempre que tanto la tarea como el registro le hayan sido asignados a él mismo.
    - **Show.hack**: Se prueban intentos de acceso no autorizado a las IDs de los registros y las tareas asignadas entre sí mediante la manipulación de URLs con IDs inválidos o de registros y/o tareas que pertenecen a otros usuarios, así como accesos de usuarios que carecen del rol de técnico.
    - **Create.safe**: Evalúa la asignación de una tarea a un registro de mantenimiento de un nuevo registro de actividad asociado a una asignación de vuelo válida. Se prueban también casos con datos incompletos o inválidos para comprobar la validación del sistema.
    - **Create.hack**: Intenta asignar tareas a un registro al cual ya está asignado nuevos registros de actividad a asignaciones de vuelo que no pertenecen al usuario o que no están disponibles, mediante la alteración del payload de la solicitud.
    - **Delete.safe**: Comprueba que un técnico pueda eliminar una relación entre un registro y una tarea siempre que ambos estén asignados a él mismo.
    - **Delete.hack**: Verifica que el sistema impida eliminar registros ya publicados o que no pertenezcan al técnico, modificando el ID en la URL o el cuerpo de la petición. Además, tampoco permite eliminar ninguna de estas relaciones por un usuario que carezca del rol de técnico.

Texto

El contenido generado por IA puede ser incorrecto.

###### La cobertura de test alcanzada en este paquete es de 88.2%, todas las líneas han sido testeadas aunque algunas por diseño de código solo se han ejecutado de forma parcial.

Seguidamente, se presenta la lista de pruebas funcionales realizadas para las funcionalidades del requisito 9 del estudiante 5, acerca de las tareas:

* + **List.safe**: Verifica que el técnico pueda ver todas las tareas que se le han asignado.
  + **List.hack**: Evalúa que el sistema no muestre ningún listado de tareas a un usuario que carezca del rol de técnico.
  + **Show.safe**: Garantiza que se puedan visualizar correctamente los detalles de una tarea específica, siempre que pertenezca al técnico.
  + **Show.hack**: Chequea que un técnico no pueda visualizar los detalles de una tarea que no le esté asignada, y que un usuario que carezca del rol de técnico no pueda visualizar los detalles de ninguna tarea.
  + **Create.safe**: Evalúa la creación de una nueva tarea. Se prueban también casos con datos incompletos o inválidos para comprobar la validación del sistema.
  + **Create.hack**: Comprueba que un usuario que carezca del rol de técnico no pueda crear ninguna tarea. Test no disponible por errores con el framework.
  + **Update.safe**: Prueba la modificación de una tarea existente en modo borrador. Se realizan pruebas tanto con datos válidos como inválidos para asegurar el correcto funcionamiento de las validaciones. Test no disponible por errores con el framework.
  + **Update.hack**: Comprueba que no sea posible modificar registros publicados, ajenos o inexistentes, manipulando los parámetros de la solicitud. Además, comprueba que no se puedan introducir datos que no puedan introducirse de manera no convencional.
  + **Publish.safe:** Verifica que una tarea pueda dejar de estar en modo borrador de manera correcta, comprobando que cumple los requisitos estipulados para ello.
  + **Publish.hack**: Verifica que no sea posible publicar tareas ajenos, ya sean de otro técnico o por un usuario que carece del rol de técnico, o registros ya publicados, mediante la modificación de las rutas en las solicitudes.
  + **Delete.safe**: Confirma que una tarea eliminarse únicamente por el técnico que tiene asignado si aún no ha sido publicada. Se asegura que las tareas ya publicadas no puedan ser eliminadas.
  + **Delete.hack**: Verifica que el sistema impida eliminar tareas ya publicadas o que no pertenezcan al técnico que realiza la solicitud, modificando el ID en la URL. Además, comprueba que un usuario que carezca del rol de técnico pueda eliminar una tarea.

La cobertura de test alcanzada en este paquete es de 76.5%, todas las líneas han sido testeadas aunque algunas por diseño de código solo se han ejecutado de forma parcial. Si fallan los tests que no son testeados por el framework.

# Performance testing

Posteriormente, se han llevado a cabo pruebas de rendimiento con el objetivo de optimizar el acceso a los datos y mejorar los tiempos de respuesta de los distintos servicios involucrados. Para ello, se han introducido índices en los campos clave de las entidades, acelerando de forma significativa las consultas a la base de datos,

especialmente en las consultas de filtrado y búsqueda.

Como resultado, se ha conseguido una mejora notable en los tiempos de ejecución, lo que impacta positivamente en la experiencia del usuario y en la eficiencia general del sistema.

#### Análisis de la mejora

Para complementar las pruebas de rendimiento, se ha realizado un análisis estadístico utilizando Excel y la prueba Z (Z-test) con el objetivo de validar si la mejora en los tiempos de respuesta tras la introducción de los índices es estadísticamente significativa.

El resultado del análisis ha arrojado un valor Pz < 0.00, lo cual permite rechazar la hipótesis nula y concluir que los cambios aplicados han tenido un efecto significativo y positivo en el rendimiento del sistema.

Este resultado se refleja de forma clara en las gráficas de tiempos, donde se observa que servicios como Task -> publish y update, que son los métodos que más consultas realizan a los repositorios y poseen queries más complejas, superaban los 135 milisegundos antes de la optimización. Tras la introducción de los índices, dichos tiempos se reducen de forma consistente, apenas alcanzando los 80 milisegundo