# 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 student 05 |

Miembros:

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

Esteban López Pérez (estlopper@alum.us.es)

Jesús Pons Morís (jesponmor@alum.us.es)

Kevin Amador Calzadilla (kevamacal@alum.us.es)

Manuel Zurita Fernández (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……………………………………………………….....5

Conclusiones…………………………………………………………………………......8

Bibliografía………………………………………………………………………………8

# Resumen ejecutivo

En este informe se presentan las pruebas funcionales y de rendimiento que se realizaron para el estudiante 5. El objetivo era garantizar que todas las funcionalidades se comportaran como se esperaba y evaluar la rapidez de respuesta del sistema en condiciones normales.

# Introducción

# El presente informe tiene como objetivo explicar el proceso mediante el cual se ha realizado el testing funcional y el análisis de rendimiento del sistema, con el fin de garantizar un proyecto de alto nivel. Para ello, se llevaron a cabo diversas pruebas relacionadas con los requisitos funcionales 8 y 9 del estudiante #5, centrados en las operaciones sobre registros de mantenimiento y tareas.

# Este documento ha sido elaborado siguiendo las directrices proporcionadas en la plataforma de enseñanza virtual. En primer lugar, se incluye una portada con las credenciales del autor del reporte, así como una tabla de versiones donde se detallan las modificaciones realizadas, clasificadas por número y fecha. A continuación, se presenta un resumen ejecutivo que contextualiza al lector sobre el contenido general del informe, seguido de una introducción al documento que describe de forma breve y clara los propósitos del mismo. Finalmente, se expone la estructura que sigue el documento, la cual se describe a continuación.

# El informe está dividido en dos capítulos principales, ambos centrados en el proceso de testing del proyecto. El primer capítulo aborda las pruebas funcionales, presentando un listado de casos de prueba organizados por funcionalidad. Cada caso incluye una breve descripción y una valoración de su eficacia en la detección de errores.

# El segundo capítulo se enfoca en las pruebas de rendimiento. Este apartado incorpora gráficos representativos y un análisis estadístico del tiempo de respuesta del sistema en dos escenarios: antes y después de optimizar la búsqueda en la base de datos mediante el uso de índices. Se incluye un intervalo de confianza del 95 % y se realiza un contraste de hipótesis utilizando un Z-test para determinar si los cambios introducidos producen mejoras estadísticamente significativas.

# Contenido

## Pruebas funcionales

Para cada funcionalidad se han realizado un conjunto de pruebas con casos positivos (.safe) y casos negativos (.hack), ahora se dirá con más detalle la metodología seguida para cada funcionalidad:

MaintenanceRecord

* list.safe: Comprueba que un técnico pueda ver correctamente los registros de mantenimiento publicados
* list-mine.safe: Comprueba que un técnico pueda ver correctamente los registros de mantenimiento que el ha creado, estén publicados o no.
* create.safe: Comprueba que cuando un técnico ingrese datos correctos la petición funcione correctamente.
* delete.safe: Comprueba que cuando un técnico quiera eliminar un registro de mantenimiento que cumpla los requisitos la petición sea exitosa.
* publish.safe: Comprueba que un técnico pueda publicar un borrador.
* show.safe: Comprueba que se muestre un registro de mantenimiento correctamente.
* update.safe: Comprueba que se actualice correctamente un registro de mantenimiento.
* create.hack: Se rellena el formulario con datos erróneos para ver si cumplen con los límites y luego se intenta hackear mediante inspección de los select introduciendo valores erróneos.
* delete.hack: Se intenta eliminar un registro de mantenimiento ya publicado y luego con un usuario que no sea técnico se intenta eliminar uno.
* publish.hack: Se intenta publicar un registro ya publicado y luego con un usuario que no sea un técnico se intenta publicar uno.
* show.hack: Se intenta mostrar un registro de mantenimiento inexistente introduciendo un id inexistente y luego con otro usuario que no sea un técnico se comprueba que no tenga acceso.
* update.hack: Se sigue el mismo procedimiento que con el create.hack.

Texto

El contenido generado por IA puede ser incorrecto.

Task

* list.safe: Comprueba que un técnico pueda ver correctamente las tareas publicadas
* list-mine.safe: Comprueba que un técnico pueda ver correctamente las tareas que el ha creado, estén publicadas o no.
* create.safe: Comprueba que cuando un técnico ingrese datos correctos la petición funcione correctamente.
* delete.safe: Comprueba que cuando un técnico quiera eliminar una tarea que cumpla los requisitos la petición sea exitosa.
* publish.safe: Comprueba que un técnico pueda publicar un borrador.
* show.safe: Comprueba que se muestre una tarea correctamente.
* update.safe: Comprueba que se actualice correctamente una tarea.
* create.hack: Se rellena el formulario con datos erróneos para ver si cumplen con los límites y luego se intenta hackear mediante inspección de los select introduciendo valores erróneos.
* delete.hack: Se intenta eliminar una tarea ya publicada y luego con un usuario que no sea un técnico se intenta eliminar uno.
* publish.hack: Se intenta publicar una tarea ya publicada y luego con un usuario que no sea un técnico se intenta publicar una.
* show.hack: Se intenta mostrar una tarea inexistente introduciendo un id inexistente y luego con otro usuario que no sea un técnico se comprueba que no tenga acceso.
* update.hack: Se sigue el mismo procedimiento que con el create.hack.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

Clase de relación entre registros y tareas (MaintenanceTaskRelation)

* list.safe: Se comprueba que se muestre correctamente un listado de tareas vinculadas a un registro de mantenimiento.
* create.safe: Se comprueba que al vincular una tarea a un registro funcione correctamente.
* delete.safe: Se comprueba que se pueda desvincular una tarea a un registro de mantenimiento.
* create.hack: Se comprueba que cuando se intente vincular una tarea inexistente mediante inspección del elemento a un registro no lo permita.
* delete.hack: Se comprueba que un usuario que no sea un técnico no pueda eliminar un asociación y que luego no se pueda eliminar una tarea inexistente de un asociación.

Texto

El contenido generado por IA puede ser incorrecto.

## Análisis de rendimiento

A continuación, se realizará un análisis del rendimiento del sistema mediante la ejecución de las pruebas funcionales mencionadas anteriormente. Las pruebas se realizaron en dos escenarios diferentes:

- Sin índices para la optimización de consultas: El archivo de Excel "tester-performance-clean.xlsx" contiene los resultados de rendimiento promedio de las operaciones de prueba, como se ilustra en la siguiente gráfica.

Gráfico, Gráfico de barras, Gráfico en cascada

El contenido generado por IA puede ser incorrecto.

Como se puede observar la funcionalidad más demandante es publicar un registro seguido de crear un registro lo cual puede ser debido a que han tenido las pruebas más demandantes, Estas funcionalidades llegan a tardar 30 milisegundos. Abajo dejo las estadísticas obtenidas con los datos de la gráfica.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Before* | |  |  |  |  |
|  |  |  | Interval (ms) | 11,0292219 | 15,2540468 |
| Media | 13,1416344 |  | Interval (s) | 0,01102922 | 0,01525405 |
| Error típico | 1,06751143 |  |  |  |  |
| Mediana | 9,95545 |  |  |  |  |
| Moda | #N/D |  |  |  |  |
| Desviación estándar | 12,0775132 |  |  |  |  |
| Varianza de la muestra | 145,866324 |  |  |  |  |
| Curtosis | 9,77689293 |  |  |  |  |
| Coeficiente de asimetría | 2,83252243 |  |  |  |  |
| Rango | 75,4143 |  |  |  |  |
| Mínimo | 3,3634 |  |  |  |  |
| Máximo | 78,7777 |  |  |  |  |
| Suma | 1682,1292 |  |  |  |  |
| Cuenta | 128 |  |  |  |  |
| Nivel de confianza(95,0%) | 2,11241247 |  |  |  |  |

Ahora pasamos a los datos obtenidos una vez se refactorizó el código con índices para optimizar las queries.

Gráfico, Gráfico de barras

El contenido generado por IA puede ser incorrecto.

A primera vista se puede ver que ninguna petición supera los 30 milisegundos aun así no hay mejora significativa, aunque en este caso el publish y el create de un registro de mantenimiento cambian puestos siendo el create el que más tarda y el publish el segundo. Abajo adjunto las estadísticas obtenidas con los datos de la gráfica.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *After* | |  |  |  |  |
|  |  |  | Interval(ms) | 8,76814017 | 12,317663 |
| Media | 10,54290156 |  | Interval(s) | 0,00876814 | 0,01231766 |
| Error típico | 0,896878851 |  |  |  |  |
| Mediana | 7,19795 |  |  |  |  |
| Moda | #N/D |  |  |  |  |
| Desviación estándar | 10,14702588 |  |  |  |  |
| Varianza de la muestra | 102,9621342 |  |  |  |  |
| Curtosis | 12,79914059 |  |  |  |  |
| Coeficiente de asimetría | 3,139082532 |  |  |  |  |
| Rango | 69,4889 |  |  |  |  |
| Mínimo | 3,1075 |  |  |  |  |
| Máximo | 72,5964 |  |  |  |  |
| Suma | 1349,4914 |  |  |  |  |
| Cuenta | 128 |  |  |  |  |
| Nivel de confianza(95,0%) | 1,774761391 |  |  |  |  |

Ahora veamos estos datos contrastados en el Z-test.

|  |  |  |
| --- | --- | --- |
| Prueba z para medias de dos muestras |  |  |
|  |  |  |
|  | *Before* | *After* |
| Media | 13,1416344 | 10,5429016 |
| Varianza (conocida) | 145,866324 | 102,962134 |
| Observaciones | 128 | 128 |
| Diferencia hipotética de las medias | 0 |  |
| z | 1,86387417 |  |
| P(Z<=z) una cola | 0,03116969 |  |
| Valor crítico de z (una cola) | 1,64485363 |  |
| P(Z<=z) dos cola | 0,06233938 |  |
| Valor crítico de z (dos colas) | 1,95996398 |  |

El p-value obtenido del Z-test es 0.06233938. Este valor es mayor que Alpha, que en este caso es 0.05. Con estos datos podemos concluir que no se ha obtenido unos cambios significativos al añadir los índices. Aun así, al ser un dato próximo a Alpha se debería de recopilar más datos.

# Conclusiones

La cobertura de código obtenida tras las pruebas es del 99% para los registros de mantenimiento y las tareas requisitos del estudiante 5. Estas pruebas han ayudado a encontrar bugs y hackeos posibles que luego han sido arreglados para cumplir con los estándares definidos por el cliente, así como el aseguramiento de la seguridad con el objetivo de cumplir con el Reglamento General de Protección de Datos (RGPD) y la Ley Orgánica 3/2018. Por otra parte también se ha visto como los índices ayudan al rendimiento, aunque en este caso no se haya obtenido datos significativos.

# Bibliografía

Intencionalmente en blanco