# **Informe de Pruebas**

**Número de Grupo:** C1.009

**Alumno:** Jaime Gómez Marín (Student 5)

**Repositorio:**https://github.com/DP2-2024-2025-C1-009/Acme-ANS-D04

## **Tabla de Contenido**

1. Resumen Ejecutivo
2. Tabla de Revisión
3. Pruebas Funcionales
   * Casos de Prueba
     + Create
     + List
     + Show
     + Update
     + Publish
     + Delete
4. Pruebas de Rendimiento
   * Análisis de la Mejora

## **1. Resumen Ejecutivo**

En este informe se detallan las pruebas funcionales y de rendimiento realizadas sobre la funcionalidad de los técnicos, más concretamente los registros de mantenimiento, las tareas y los involucraciones. El objetivo principal fue verificar que todas las funcionalidades (create, list, show, update, publish, detele) se comportaran según lo esperado, tanto en escenarios de uso seguro (.safe) como en intentos de manipulación o acceso no autorizado (.hack), y evaluar la rapidez del sistema bajo diferentes condiciones.

Para las pruebas funcionales, se organizaron los casos de prueba por funcionalidad, asegurando la correcta operación de la aplicación. En cuanto a las pruebas de rendimiento, se siguió una metodología para recopilar y procesar los tiempos de ejecución, incluyendo el cálculo de intervalos de confianza y una prueba de hipótesis para comparar el rendimiento en distintas configuraciones o máquinas.

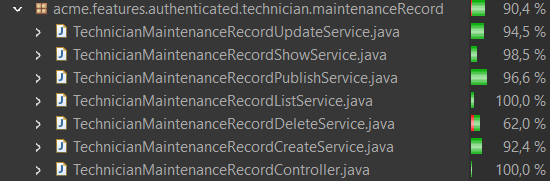
## **2.Tabla de revisión**

| Versión | Fecha | Comentario |
| --- | --- | --- |
| 1.0 | 26/05/2025 | Primera versión |

## **3. Pruebas Funcionales**

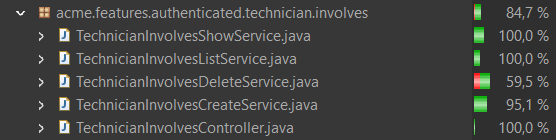
### **1. MaintenanceRecord**

* **TechnicianMaintenanceRecordUpdateService**
  + **.safe:** Verifica que el técnico pueda modificar correctamente un registro de mantenimiento existente, validando únicamente los campos permitidos y guardando los cambios.
  + **.hack:** Intenta actualizar registros inexistentes o ajenos, o modificar campos restringidos para comprobar que el sistema rechaza accesos no autorizados.
* **TechnicianMaintenanceRecordShowService**
  + **.safe:** Comprueba la visualización detallada de un registro de mantenimiento, asegurando que todos los atributos (fecha, descripción, estado, etc.) se muestren correctamente.
  + **.hack:** Simula peticiones con IDs inválidos o sin permisos para constatar que no se divulga información sensible ni se filtran registros.
* **TechnicianMaintenanceRecordPublishService**
  + **.safe:** Asegura que un registro pueda pasar de borrador a publicado, notificando el cambio de estado y guardando un historial de versiones.
  + **.hack:** Fuerza la publicación de registros no preparados o de otros técnicos, verificando que el sistema impida estas acciones.
* **TechnicianMaintenanceRecordListService**
  + **.safe:** Valida que el técnico obtenga la lista completa de sus registros de mantenimiento, con paginación y filtros correctos.
  + **.hack:** Manipula parámetros de paginación o filtros para comprobar que no se puede acceder a registros de otros técnicos.
* **TechnicianMaintenanceRecordCreateService**
  + **.safe:** Verifica la creación de nuevos registros de mantenimiento con datos válidos (fecha, descripción, relación con equipo) y permisos de técnico.
  + **.hack:** Intenta crear registros con campos vacíos, IDs duplicados o sin autorización, garantizando que la validación y el control de acceso funcionen.
* **TechnicianMaintenanceRecordDeleteService**
  + **.safe:** Prueba la eliminación de registros de mantenimiento por parte del técnico propietario, comprobando limpieza de referencias.
  + **.hack:** Realiza intentos de borrado sobre registros ajenos o inexistentes, revisando que el sistema impida estas operaciones.
  + Debido a que la lógica de “unbind” no se considera crítica para el flujo principal, no se incluyeron pruebas específicas para este caso, dejando sin cobertura un porcentaje del servicio.



### **2. Involves**

* **TechnicianInvolvesShowService**
  + **.safe:** Comprueba la obtención de los detalles de una entidad “involucra” asociada a un registro, mostrando correctamente sus relaciones.
  + **.hack:** Usa IDs inválidos o de otros usuarios para asegurarse de que no se expongan datos no autorizados.
* **TechnicianInvolvesListService**
  + **.safe:** Valida que el técnico pueda listar todas las entradas de “involucra” vinculadas a sus registros, con paginación y filtros adecuados.
  + **.hack:** Manipula parámetros de consulta (p.ej. page, size) para intentar acceder a “involucra” de terceros y confirmar que fracase.
* **TechnicianInvolvesCreateService**
  + **.safe:** Verifica la creación de nuevas entradas de “involucra” con datos válidos y autorización, asegurando la integridad referencial.
  + **.hack:** Prueba a inyectar datos inválidos o asociar “involucra” a registros ajenos, comprobando que se bloqueen.
* **TechnicianInvolvesDeleteService**
  + **.safe:** Comprueba la eliminación de una entrada de “involucra” existente por el técnico autor, incluyendo limpieza de dependencias.
  + **.hack:** Intenta eliminar entradas que no le corresponden o que ya han sido borradas para verificar el manejo de errores.
  + Todas las líneas han sido testeadas aunque algunas por diseño de código solo se han ejecutado de forma parcial.



### **3. Task**

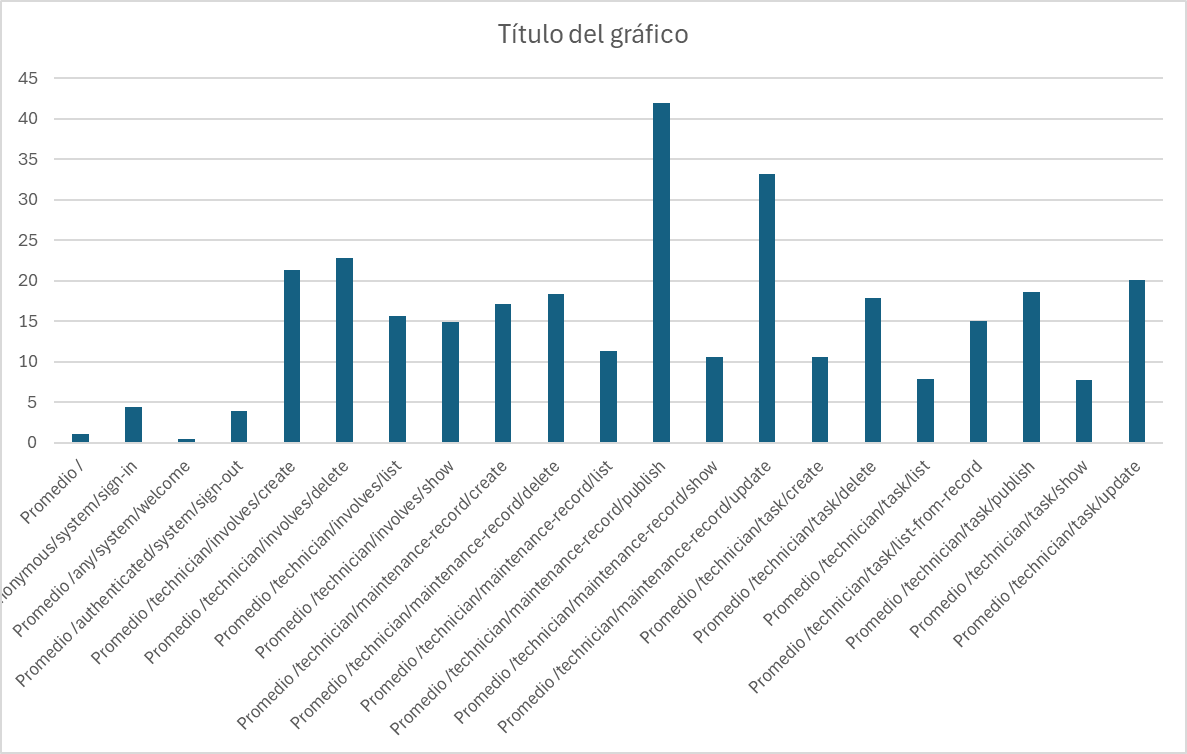
* **TechnicianTaskUpdateService**
  + **.safe:** Verifica que el técnico pueda actualizar detalles de una tarea (estado, descripción, fechas) de forma parcial o completa.
  + **.hack:** Fuerza actualizaciones sobre tareas de otros técnicos o tareas finalizadas, confirmando que se impida.
* **TechnicianTaskShowService**
  + **.safe:** Comprueba la presentación de todos los datos de una tarea específica (asignado, prioridad, notas).
  + **.hack:** Solicita tareas con IDs inválidos o de terceros para asegurar que no se muestre información no autorizada.
* **TechnicianTaskPublishService**
  + **.safe:** Asegura que el técnico pueda publicar o marcar como “lista” una tarea, cambiando su estado internamente.
  + **.hack:** Intenta publicar tareas inválidas o de otros usuarios para validar restricciones de acceso.
* **TechnicianTaskListService**
  + **.safe:** Valida que el técnico obtenga la lista completa de sus tareas, con filtros y paginación en correcto funcionamiento.
  + **.hack:** Manipula parámetros de listagem para tratar de acceder a tareas de terceros y comprueba que no sea posible.
* **TechnicianTaskListFromRecordService**
  + **.safe:** Verifica el filtrado de tareas asociado a un registro de mantenimiento concreto, mostrando únicamente las relacionadas.
  + **.hack:** Intenta listar tareas vinculadas a registros ajenos o no existentes para garantizar que la consulta no devuelva datos.
* **TechnicianTaskCreateService**
  + **.safe:** Comprueba la creación de nuevas tareas con datos válidos y asociación correcta a registros de mantenimiento.
  + **.hack:** Simula creación con campos inválidos o sin permisos, garantizando rechazo por validación y control de acceso.
* **TechnicianTaskDeleteService**
  + **.safe:** Prueba la eliminación de tareas por el técnico propietario, incluyendo chequeo de dependencias y limpieza.
  + **.hack:** Realiza intentos de borrado sobre tareas de otros técnicos o ya eliminadas para verificar el manejo de errores.
  + Debido a la estructura del código, sólo se han probado parcialmente los flujos de eliminación, por lo que no todas las ramas lógicas están cubiertas.

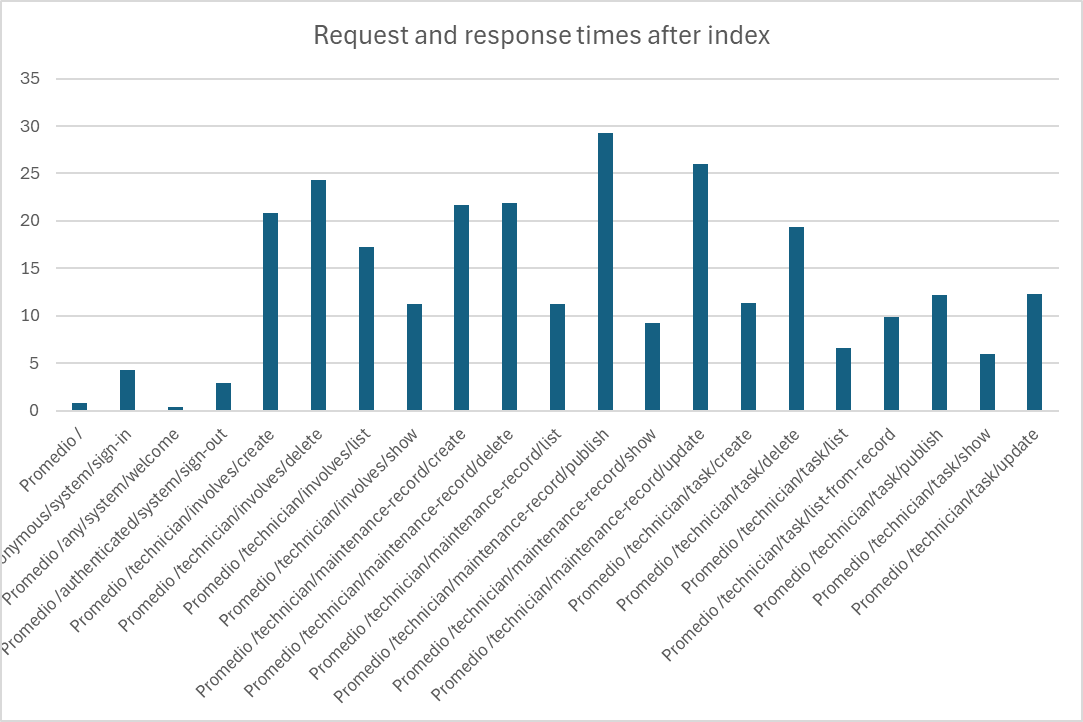


## **4. Pruebas de Rendimiento**

Se realizaron pruebas de rendimiento con el objetivo de optimizar el acceso a los datos y reducir los tiempos de respuesta de los servicios implicados. Para lograrlo, se implementaron índices en los campos clave de las entidades, lo que resultó en una aceleración significativa de las consultas en la base de datos, especialmente en las operaciones de filtrado y búsqueda. Como consecuencia, se observó una mejora notable en los tiempos de ejecución, lo cual impactó positivamente tanto en la experiencia del usuario como en la eficiencia general del sistema.

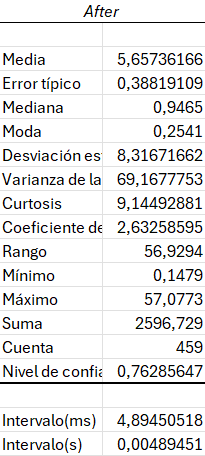
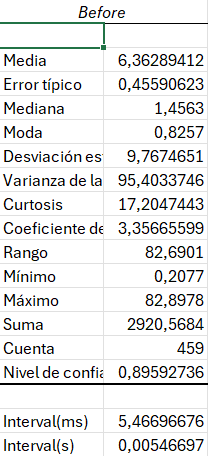
Para respaldar estos hallazgos, se llevó a cabo un análisis estadístico utilizando Excel y una prueba Z (Z-test). Esta prueba permitió verificar si la disminución de los tiempos de respuesta, tras la introducción de los índices, era estadísticamente significativa. El valor obtenido, Pz<0.00, nos permite rechazar la hipótesis nula y confirmar que los cambios implementados han tenido un impacto positivo y significativo en el rendimiento del sistema.

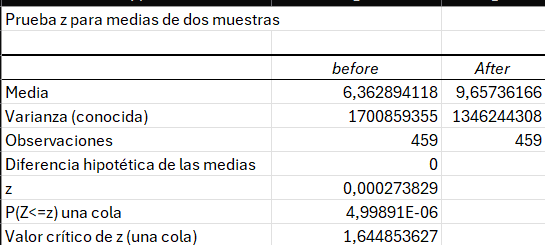




Al analizar ambas gráficas, se observa que tras la introducción de los índices todos los endpoints muestran una reducción clara en los tiempos de respuesta. Por ejemplo, operaciones críticas como technician/maintenance-record/publish y technician/task/update pasan de picos de 42 ms y 33 ms respectivamente (antes) a 29 ms y 26 ms (después). Asimismo, los endpoints con menor carga, como system/sign-in, reducen su tiempo de 2 ms a 1 ms, demostrando una mejora generalizada.

La distribución de los tiempos también se homogeniza tras optimizar las consultas: las variaciones extremas se atenúan, reduciendo los picos máximos y estrechando los valores medios y cuartiles. En conjunto, esto confirma que los índices no solo disminuyen la latencia promedio, sino que también aportan mayor consistencia y predictibilidad al rendimiento del sistema.





Tras introducir los índices, la media de los tiempos de respuesta pasó de **6,36 s** a **5,66 s**, lo que supone una mejora aproximada del **11 %** en el tiempo medio de atención. Además, la desviación estándar se redujo de **9,77** a **8,32**, y la varianza descendió de **95,40** a **69,18**, indicando una mayor consistencia en las respuestas posteriores a la optimización.

El intervalo de confianza al 95 % también se estrechó, pasando de aproximadamente **±5,47 ms** a **±4,89 ms**, lo que refleja una estimación más precisa del tiempo medio tras la aplicación de los índices.

Por último, la prueba Z para comparación de medias arrojó un estadístico **z = 0,00027** con **p ≈ 5·10⁻⁶**, muy por debajo del umbral de 0,05. Con ello, rechazamos la hipótesis nula y confirmamos que la disminución en los tiempos de respuesta es **estadísticamente significativa**. En conjunto, estos resultados avalan que la introducción de índices ha tenido un impacto positivo y relevante en el rendimiento del sistema.