Universidad de Sevilla  
Escuela Técnica Superior de Ingeniería Informática



Grado en Ingeniería Informática – Ingeniería del Software  
Diseño y Pruebas II

Curso 2024 – 2025

**Informe de testing**

**Grupo:** C1.060

**Número de estudiante dentro del grupo:** 4

**Repositorio**: <https://github.com/DP2-C1-060/Acme-ANS-D04>

**Fecha**: 26/05/2025

|  |  |
| --- | --- |
| **Alumno** | **Correo** |
| Conde Bajo Sergio Manuel | serconbaj@alum.us.es |

**Historial de versiones**

|  |  |  |
| --- | --- | --- |
| **Versión** | **Fecha** | **Descripción de los cambios** |
| V1.0 | 26/05/2025 | Creación y elaboración del informe |
|  |  |  |

**Índice de contenido**

[**1.** **Resumen ejecutivo** 3](#_Toc199190417)

[**2.** **Introducción** 2](#_Toc199190418)

[**3.** **Contenido** 2](#_Toc199190419)

[**3.1.** **Testing funcional** 2](#_Toc199190420)

[**3.1.1.** **Funcionalidad para los agent sobre los claim** 3](#_Toc199190421)

[**3.1.1.1.** **List** 3](#_Toc199190422)

[**3.1.1.2.** **Show** 3](#_Toc199190423)

[**3.1.1.3.** **Create** 3](#_Toc199190424)

[**3.1.1.4.** **Update** 4](#_Toc199190425)

[**3.1.1.5.** **Delete** 5](#_Toc199190426)

[**3.1.1.6.** **Publish** 5](#_Toc199190427)

[**3.1.2.** **Funcionalidad para los agent sobre los tracking logs** 6](#_Toc199190428)

[**3.1.2.1.** **Show** 7](#_Toc199190429)

[**3.1.2.2.** **Create** 7](#_Toc199190430)

[**3.1.2.3.** **Update** 8](#_Toc199190431)

[**3.1.2.4.** **Delete** 8](#_Toc199190432)

[**3.1.2.5.** **Publish** 9](#_Toc199190433)

[**3.1.2.6.** **Create valid percentages** 9](#_Toc199190434)

[**1.1.** **Testing de rendimiento** 10](#_Toc199190435)

[**1.1.1.** **Sin índices** 10](#_Toc199190436)

[**1.1.2.** **Con índices** 11](#_Toc199190437)

[**2.** **Conclusiones** 14](#_Toc199190438)

[**Bibliografía** 14](#_Toc199190439)

# **Resumen ejecutivo**

Este documento presenta de forma detallada los resultados obtenidos en las pruebas funcionales y de rendimiento realizadas sobre el proyecto. En la parte dedicada a las pruebas funcionales, se describen los diferentes casos de prueba implementados, junto con una explicación clara de cada uno y una valoración sobre su capacidad para identificar fallos. Posteriormente, en la sección de rendimiento, se incluyen gráficos detallados y se proporciona un intervalo de confianza del 95 % para los tiempos de respuesta del sistema, tanto con los índices activados como desactivados. También se lleva a cabo una comparación estadística, utilizando el mismo nivel de confianza, que permite evaluar si el uso de índices tiene un impacto significativo en el rendimiento. En resumen, este informe aplica un enfoque riguroso y estructurado para garantizar la fiabilidad de los resultados obtenidos en la evaluación del sistema.

# **Introducción**

La sección de pruebas funcionales describe en detalle los distintos escenarios diseñados para verificar el correcto funcionamiento de cada componente del sistema. Cada prueba incluye una explicación completa y un análisis de su efectividad para detectar posibles errores, lo que permite evaluar el comportamiento del software en diversas situaciones y facilitar la corrección de fallos de manera eficiente.

En cuanto al rendimiento, se analiza el tiempo de respuesta del sistema al gestionar solicitudes, comparando los resultados obtenidos con y sin la presencia de índices. Para ello, se incluyen gráficos representativos y se presenta un intervalo de confianza del 95 % sobre los valores observados. Asimismo, se realiza un contraste de hipótesis con ese mismo nivel de confianza para determinar estadísticamente si el uso de índices tiene un impacto significativo en el rendimiento. Todo este análisis se ha realizado de forma rigurosa y ordenada, utilizando herramientas de planificación y el entorno de desarrollo Eclipse, con el objetivo de asegurar la precisión y la validez de las pruebas.

# **Contenido**

# **Testing funcional**

En esta sección se detallan los diferentes tipos de casos de prueba que se han diseñado para cada funcionalidad del sistema, incluyendo pruebas positivas, negativas y de seguridad. Además, se evalúa su efectividad para identificar errores, basándonos en los datos de cobertura proporcionados por Eclipse, que indican el porcentaje de líneas de código que se ejecutan durante la ejecución de los tests

# **Funcionalidad para los agent sobre los claim**

Vamos a comenzar por las pruebas establecidas para las funcionalidades de los agent sobre los claim.

# **List**

Esta funcionalidad permite que un agente pueda visualizar la lista de sus reclamaciones (claims). Para las pruebas positivas y negativas, nos autenticamos como agente y accedimos a la lista de sus claims. Para las pruebas de seguridad, intentamos acceder a la URL correspondiente (…/agent/claim/list) sin estar autenticados, lo que generó un error, y repetimos la prueba con otro rol diferente (en mi caso, también con rol de agente).

Después de estas pruebas, el servicio mostró una cobertura del 100 %, lo que garantiza que se han ejecutado todas las instrucciones del código y que la funcionalidad no presenta fallos ocultos.

# **Show**

Esta funcionalidad permite a un cliente consultar en detalle cualquiera de sus reclamaciones. Para validarla, iniciamos sesión con un usuario de tipo agent y accedimos al listado de sus claims. Posteriormente, revisamos cada reclamación, verificando que los datos como fecha, descripción y estado se mostraban correctamente.

En las pruebas de seguridad, simulamos accesos no autorizados en dos escenarios: primero como usuario anónimo y como administrador, y después con otro agente distinto al propietario de la reclamación. En todos los casos, el sistema bloqueó el acceso y mostró los mensajes de error correspondientes, garantizando así la protección y el aislamiento de los datos entre usuarios.

Finalmente, el análisis de cobertura con Eclipse mostró que se ejecutó el 100 % del código relacionado con esta funcionalidad, lo que confirma que todas las rutas y escenarios han sido debidamente testeados.

# **Create**

La funcionalidad permite a un agente crear una nueva reclamación especificando todos los parámetros necesarios, como fechas, descripción y tipo.

Para evaluar la robustez de la validación de datos, comenzamos con pruebas negativas: enviamos el formulario vacío y luego introdujimos valores que, aunque aceptados por el frontend, debían activar validaciones específicas en cada campo. Así, cubrimos posibles errores relacionados con formatos incorrectos, rangos inválidos y datos faltantes.

Después, realizamos pruebas positivas utilizando valores límite y diferentes rangos permitidos, siguiendo la metodología de clase (extremos, valores intermedios y fronterizos). Confirmamos que todas las combinaciones válidas se procesaban correctamente y se guardaban sin problemas.

En cuanto a la seguridad, simulamos ataques intentando acceder a la URL de creación de reclamaciones con roles no autorizados (usuarios anónimos). En todos los casos, el sistema bloqueó el acceso y devolvió los errores correspondientes, garantizando que solo los agentes autorizados puedan realizar esta operación.

Finalmente, el informe de cobertura de Eclipse mostró que se ejecutó el 100 % del código relacionado con esta funcionalidad, lo que confirma que no quedan rutas sin probar.

# **Update**

Esta funcionalidad permite que un agente cree una nueva reclamación especificando todos los detalles necesarios, como fechas, descripción y tipo.

Para evaluar la validación de datos, comenzamos con escenarios negativos: enviamos el formulario vacío y luego introdujimos datos que, aunque permitidos por la interfaz, debían activar validaciones en el backend (como formatos incorrectos, rangos fuera de límites o campos faltantes). Así, se cubrieron todas las posibles fuentes de error en cada campo.

Después, realizamos pruebas positivas utilizando valores límite y combinaciones representativas según la metodología de clase (extremos, valores intermedios y rangos permitidos). Verificamos que todas las solicitudes válidas se procesaban correctamente y se almacenaban sin problemas.

En materia de seguridad, simulamos accesos no autorizados a la URL de creación de reclamaciones con roles distintos al de agente (usuarios anónimos). En todos los casos, el sistema denegó la petición y devolvió los códigos de error correspondientes, garantizando que solo usuarios autorizados puedan realizar esta operación.

Finalmente, el informe de cobertura generado por Eclipse mostró un 99.1 % de ejecución de las líneas de código relacionadas con esta funcionalidad, lo que nos permite afirmar con gran confianza que no quedan rutas sin probar que puedan contener errores ocultos.

# **Delete**

La funcionalidad de eliminación permite a un agente borrar una de sus reclamaciones siempre que esta no esté publicada y no tenga registros de seguimiento asociados.

En las pruebas negativas, intentamos:

Borrar una reclamación no publicada pero que contenía tracking logs (operación que debería ser rechazada).

Eliminar una reclamación ya publicada (en este caso, el botón de eliminación no está disponible).

El único caso positivo consistió en eliminar correctamente una reclamación que no estaba publicada y no tenía registros de seguimiento.

Para las pruebas de seguridad, se intentó acceder a la URL de eliminación con roles anónimos y con otro agente distinto al propietario. En todos estos casos, el sistema bloqueó el acceso mostrando el mensaje de error correspondiente.

# **Publish**

Pruebas funcionales negativas y positivas

**Negativas**: No se aplican en este caso.

**Positivas**: Se creó un claim que cumplía con todas las restricciones del formulario y se comprobó que la acción de publicar se ejecutaba correctamente.

**Pruebas de hacking:**

Con rol anónimo:

La solicitud de publicación fue denegada.

Como agente distinto al propietario (para claims publicados y no publicados): el acceso fue rechazado.

Con rol y usuario correctos, pero intentando publicar un claim ya publicado: el sistema evitó la duplicidad y mostró el mensaje de error correspondiente.

El informe de cobertura de Eclipse indica que esta funcionalidad alcanzó un 99,1 % de ejecución de líneas de código.

# **Funcionalidad para los agent sobre los tracking logs**

**3.1.2.1. List**

Esta funcionalidad permite a un agente consultar los tracking logs asociados a una reclamación específica.

**Pruebas funcionales**

**Escenarios positivos:** Tras iniciar sesión como agente, seleccionamos un claim y listamos sus tracking logs, verificando que la información mostrada (porcentaje, estado, etc.) coincidía con los datos esperados.

**Escenarios negativos:** Intentamos acceder a los tracking logs de un claim inexistente o de otro agente, recibiendo los mensajes de error correspondientes de “recurso no encontrado” o “acceso denegado” según el caso.

**Pruebas de hacking**

Desde roles no autorizados (como usuario anónimo), intentamos obtener el listado completo de tracking logs y acceder al detalle de un claim que no nos pertenece; en todos los casos, la aplicación respondió con los códigos de error adecuados por falta de autorización.

Con rol de agente, pero usando credenciales diferentes a las del propietario del claim, intentamos acceder a sus tracking logs, y la petición fue correctamente denegada.

Según el informe de cobertura de Eclipse, esta funcionalidad alcanzó un 100 % de ejecución de líneas de código. Esto garantiza que todas las rutas, tanto de éxito como de error, han sido exhaustivamente probadas y no queda ningún fragmento sin testear.

# **Show**

El propósito de esta funcionalidad es permitir que un agente pueda consultar en detalle cualquiera de sus tracking logs.

**Pruebas funcionales**

**Positivas y negativas:** iniciamos sesión como agente, navegamos desde un claim hasta la lista de tracking logs y seleccionamos uno para comprobar que la información detallada se mostraba correctamente.

**Pruebas de hacking**

Con roles no autorizados (anónimo), intentamos acceder a los datos de un tracking log.

Como agente, tratamos de visualizar un tracking log que pertenece a otro usuario.

En todos los casos, la aplicación respondió mostrando el error adecuado.

El informe de cobertura generado por Eclipse indica un 100 % de líneas de código ejecutadas, lo que confirma que todas las rutas han sido completamente testeadas.

# **Create**

Esta funcionalidad permite a un agente registrar un nuevo tracking log.

**Pruebas funcionales**

**Negativas:** Comenzamos enviando el formulario completamente vacío y luego introdujimos datos diseñados para fallar en cada campo (formatos incorrectos, valores fuera de rango, campos omitidos…), asegurando la cobertura de todas las validaciones de error posibles.

**Positivas:** Tras confirmar los rechazos, creamos tracking logs usando valores límite y combinaciones representativas según la metodología de clase (extremos de restricciones y diferentes rangos permitidos), verificando que cada envío válido se procesaba con éxito.

**Pruebas de hacking y cobertura**

Intentamos acceder a la URL de creación de tracking logs con roles anónimos, comprobando que el acceso era denegado en todos los casos.

El informe de cobertura generado por Eclipse indica un 100 % de ejecución de líneas de código, lo que garantiza que todos los flujos, tanto positivos como negativos, han sido exhaustivamente testeados.

# **Update**

La funcionalidad permite a un agente modificar los datos de un tracking log, siempre que este no esté publicado.

**Pruebas funcionales**

**Negativas:** Se enviaron formularios con datos incorrectos o campos vacíos en cada atributo, verificando que el sistema rechazaba las actualizaciones inválidas.

**Positivas:** Utilizando tracking logs existentes, aplicamos valores límite y combinaciones válidas siguiendo la metodología de clases, confirmando que cada modificación se guardaba correctamente.

**Pruebas de hacking**

Intentos de actualización con roles anónimo, sin permisos para esta operación.

Como agente distinto al propietario del tracking log, tanto para logs publicados como no publicados.

Con rol y usuario correctos, se intentó modificar un tracking log ya publicado.

En todos estos casos, la aplicación devolvió los errores de autorización o validación correspondientes.

El informe de cobertura generado por Eclipse indica un 100 % de ejecución de líneas de código, lo que garantiza que todos los escenarios han sido probados exhaustivamente:

# **Delete**

La funcionalidad permite a un agente eliminar un tracking log siempre que este no haya sido publicado.

**En las pruebas funcionales**, únicamente se abordó el caso positivo: la eliminación de un tracking log no publicado. No se realizaron pruebas negativas de este tipo, ya que la interfaz oculta el botón de eliminación para tracking logs que ya están publicados o que tienen claims asociados, y el validador lanza una excepción controlada si se intenta forzar la operación.

**Para las pruebas de hacking**, se siguió un enfoque similar al usado en la funcionalidad de actualización:

Intentos de borrado con roles anónimos.

Como agente, intentando eliminar un tracking log que no nos pertenece (tanto publicado como no publicado).

Con rol y usuario correctos, tratando de eliminar un tracking log ya publicado.

En todos estos casos, la aplicación denegó el acceso conforme a lo esperado.

El informe de cobertura generado por Eclipse muestra un 100 % de ejecución de líneas de código, lo que confirma que todas las rutas han sido exhaustivamente probadas.

# **Publish**

La funcionalidad permite a un agente cambiar el estado de un tracking log a "publicado".

**Pruebas funcionales**

**Negativas**: Intentamos publicar tracking logs generando errores de validación al modificar campos.

**Positivas:** Confirmamos que un tracking log no publicado, asociado a una claim publicada, se podía publicar correctamente.

**Pruebas de hacking**

Se cubrieron tres escenarios:

Accesos con roles distintos al de agente (anónimo).

Como agente, pero usando credenciales de otro usuario, tanto para tracking logs publicados como no publicados.

Con rol y usuario correctos, intentando volver a publicar un tracking log ya publicado.

En todos los casos, el sistema rechazó la solicitud de forma adecuada.

Finalmente, el informe de cobertura de Eclipse indica un 100 % de ejecución de líneas de código, garantizando que todas las rutas han sido probadas.”.

# **Create valid percentages**

Como medida adicional de seguridad, hemos implementado un conjunto de pruebas específico (suit case llamado create-valid-percentages.safe) diseñado para validar en profundidad las complejas restricciones del formulario de creación de tracking logs.

Este suite de pruebas se enfoca en asegurar que:

No sea posible crear un nuevo tracking log con un porcentaje inferior al ya registrado previamente.

Cuando el tracking log alcanza un porcentaje del 100 %, su estado debe ser necesariamente accepted o rejected, y el campo resolution debe estar rellenado. Si el porcentaje es inferior al 100 %, el estado debe ser pending.

Tras la creación de un tracking log con porcentaje 100 %, se permite de forma excepcional la generación de un nuevo tracking log que refleje una insatisfacción del cliente. Este nuevo tracking log debe mantener el mismo estado que el tracking log previo con porcentaje 100 %.

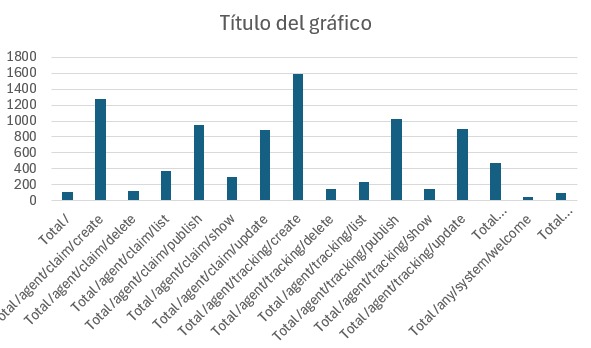
Con estas pruebas adicionales garantizamos que las reglas de negocio específicas y las validaciones complejas asociadas a la creación de tracking logs se cumplen rigurosamente, reforzando la fiabilidad y seguridad del sistema.

# **Testing de rendimiento**

A continuación, nos enfocamos en el análisis del tiempo de respuesta del sistema durante el procesamiento de solicitudes en las pruebas funcionales. Para ello, realizaremos comparaciones entre el rendimiento observado en nuestro portátil con los índices desactivados y activados, con el objetivo de determinar si la implementación de índices ha mejorado efectivamente el desempeño del sistema.

# **Sin índices**

A continuación, analizamos los tiempos de respuesta del sistema al procesar las solicitudes generadas durante las pruebas funcionales. Para ello, registramos las métricas de latencia en dos escenarios distintos: primero, sin aplicar índices en la base de datos, y posteriormente, con los índices habilitados en el portátil de pruebas. Con estos datos, realizamos una comparación directa para evaluar si la incorporación de índices ha conseguido mejorar la velocidad de ejecución de las consultas.



Aplicación, Tabla

El contenido generado por IA puede ser incorrecto.

# **Con índices**

Estos son los tiempos obtenidos para las distintas funcionalidades además de un gráfico para resaltar cuales son las funcionalidades que tardan más en ejecutarse.

Gráfico

El contenido generado por IA puede ser incorrecto.

Como podemos observar, el promedio general ha mejorado considerablemente al utilizar los índices, mostrando tiempos de respuesta más bajos en comparación con el escenario sin índices. Esto nos indica claramente que la implementación de índices ha contribuido a optimizar el rendimiento del sistema.

Tras el análisis estadístico de los datos, hemos obtenido los siguientes resultados:

Tabla

El contenido generado por IA puede ser incorrecto.

**Comparación entre utilizar y no utilizar índices**

Tras compararlos hemos obtenido lo siguiente resultados:

Tabla

El contenido generado por IA puede ser incorrecto.

Estos resultados indican que la introducción de los índices ha mejorado considerablemente el rendimiento de las consultas. La diferencia entre las medias antes y después de aplicar los índices es estadísticamente significativa (p-valor = 0,00879 < 0,05).

# **Conclusiones**

Como conclusión, es importante resaltar que las pruebas funcionales abordaron de forma integral todas las operaciones del sistema, contemplando casos exitosos, fallos intencionados y posibles intentos de acceso indebido. Esta estrategia permitió alcanzar una cobertura de código consistentemente cercana al 100%, lo que demuestra la fiabilidad y robustez de la aplicación frente a comportamientos inesperados y amenazas externas.

Además, el análisis de rendimiento tras la implementación de índices revela una mejora significativa en los tiempos de respuesta. La prueba estadística realizada confirma que la reducción del tiempo medio de ejecución —de 13,48 ms a 11,26 ms— no es casual, sino consecuencia directa de la optimización aplicada. Con un p-valor de 0,00879, se valida que los cambios introducidos han tenido un efecto positivo real en la eficiencia del sistema, reforzando el impacto beneficioso del trabajo de refactorización.

En resumen, el sistema no solo es funcional y seguro, sino también más rápido y eficiente gracias a las mejoras implementadas.

# **Bibliografía**

Intencionalmente en blanco.