Universidad de Sevilla  
Escuela Técnica Superior de Ingeniería Informática

**Student 4 Testing Report**



Grado en Ingeniería Informática – Ingeniería del Software  
Diseño y Pruebas 2.

Curso 2024 – 2025

|  |  |  |
| --- | --- | --- |
| **Grupo de prácticas: C1.004** | | |
| **Autores por orden alfabético** | **Rol** | **Descripción del rol** |
| Bernardos Ruiz, Alberto | Developer | Persona encargada de desarrollar el código. |
| García León, Guillermo | Developer | Persona encargada de desarrollar el código. |
| Fernández Román, Santiago | Project Manager | Persona encargada de tomar decisiones de diseño y vigilar el correcto desarrollo. |

**Control de Versiones**

|  |  |  |
| --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** |
| 26/05/2025 | v1.0.0 | Desarrollo de la primera versión. |
| 02/07/2025 | v2.0.0 | Versión de convocatoria de julio. |
|  |  |  |
|  |  |  |

**Índice de contenido**

[**1.** **Introducción:** 2](#_Toc202434818)

[**2.** **Testing funcional:** 2](#_Toc202434819)

[**3.** **Testing de rendimiento:** 5](#_Toc202434820)

[**4.** **Conclusión:** 10](#_Toc202434821)

# **Introducción:**

En este documento se detalla la realización de pruebas funcionales y de rendimiento realizadas sobre el proyecto. El objetivo principal ha sido verificar el correcto funcionamiento de las funcionalidades implementadas y evaluar el rendimiento de la aplicación, tanto antes como después de la incorporación de mejoras estructurales como la implementación de índices en las entidades. Las pruebas se han dividido en pruebas funcionales —donde se comprueba el comportamiento del sistema frente a diferentes escenarios de uso y ataque— y pruebas de rendimiento —donde se analiza el impacto de optimizaciones técnicas en los tiempos de respuesta del sistema.

# **Testing funcional:**

**Pruebas de funcionalidades de claims:**

* **List pending:**
* .safe: se listan las claims pendientes por completar asociadas al usuario agent1.
* .hack: se intentan listar las claims desde un usuario no logueado. Resulta en error de autorización.
* **List complete:**
* .safe: se listan las claims pendientes por completar asociadas al usuario agent1.
* .hack: se intentan listar las claims desde un usuario no logueado. Resulta en error de autorización.
* **Show:**
* .safe: se accede a una claim usando el usuario correcto.
* .hack: se intenta acceder a una claim desde un usuario no logueado y un usuario incorrecto. Ambos casos resultan en error de autorización.
* **Create:**
* .safe: se prueba a crear una claim con valores de atributos vacíos, con valores inválidos y con valores correctos. Solo en el último caso permite crear la claim.
* .hack: se intenta crear una claim editando atributos de solo lectura y atributos enumerados con valores incorrectos. En el primer caso se ignora la edición y en el segundo da error.
* **Delete:**
* .safe: se borra una claim usando el usuario correcto.
* .hack: se intenta borrar una claim sin publicar desde un usuario anónimo, desde un usuario incorrecto. Se intenta borrar una claim publicada desde el usuario correcto. En todos los casos da error de autorización.
* **Update:**
* .safe: usando el usuario correcto, se prueba a crear una claim con valores de atributos vacíos, con valores inválidos y con valores correctos. Solo en el último caso permite actualizar la claim.
* .hack: usando un usuario no logueado y un usuario incorrecto, se inteneta editar una claim. En ambos casos da error de autorización.

Usando un usuario correcto, se intenta editar una claim editando atributos de solo lectura y atributos enumerados con valores incorrectos. En el primer caso se ignora la edición del atributo y en el segundo da error.

* **Publish**:
* .safe: usando el usuario correcto, se prueba a publicar una claim con valores de atributos vacíos, con valores inválidos y con valores correctos. Solo en el último caso permite publicar la claim.
* .hack: usando un usuario no logueado y un usuario incorrecto, se intenta publicar una claim. En ambos casos da error de autorización.

Usando un usuario correcto, se intenta publicar una claim ya publicada y una no publicada editando atributos de solo lectura y atributos enumerados con valores incorrectos. En el primer caso da error de autorización, en el segundo se ignora la edición del atributo y se publica exitosamente. En el tercer caso da error.

**Funcionalidades de tracking logs:**

* **List:**
* .safe: se listan los tracking logs pendientes por completar asociadas al usuario agent1.
* .hack: se intentan listar los tracking logs desde un usuario no logueado y un usuario incorrecto. Ambos resultan en error de autorización.
* **Show:**
* .safe: se accede a un tracking log usando el usuario correcto.
* .hack: se intenta acceder a un tracking log desde un usuario no logueado y un usuario incorrecto. Ambos casos resultan en error de autorización.
* **Create:**
* .safe: se prueba a crear un tracking log con valores de atributos vacíos, con valores inválidos y con valores correctos. Solo en el último caso permite crear el tracking log.
* .hack: se intenta crear un tracking log editando atributos de solo lectura y atributos enumerados con valores incorrectos. En el primer caso se ignora la edición y en el segundo da error.
* **Delete:**
* .safe: se borra un tracking log usando el usuario correcto.
* .hack: se intenta borrar un tracking log sin publicar desde un usuario anónimo, desde un usuario incorrecto. Se intenta borrar un tracking log publicada desde el usuario correcto. En todos los casos da error de autorización.
* **Update:**
* .safe: usando el usuario correcto, se prueba a crear un tracking log con valores de atributos vacíos, con valores inválidos y con valores correctos. Solo en el último caso permite actualizar el tracking log.
* .hack: usando un usuario no logueado y un usuario incorrecto, se intenta editar un tracking log. En ambos casos da error de autorización.

Usando un usuario correcto, se intenta editar un tracking log editando atributos de solo lectura y atributos enumerados con valores incorrectos. En el primer caso se ignora la edición del atributo y en el segundo da error.

* **Publish:**
* .safe: usando el usuario correcto, se prueba a publicar un tracking log con valores de atributos vacíos, con valores inválidos y con valores correctos. Solo en el último caso permite publicar el tracking log.
* .hack: usando un usuario no logueado y un usuario incorrecto, se intenta publicar un tracking log. En ambos casos da error de autorización.

Usando un usuario correcto, se intenta publicar un tracking log ya publicado y uno no publicado editando atributos de solo lectura y atributos enumerados con valores incorrectos. En el primer caso da error de autorización, en el segundo se ignora la edición del atributo y se publica exitosamente. En el tercer caso da error.

# **Testing de rendimiento:**

Para las pruebas de rendimiento se ha utilizado la función de replay sobre los tests funcionales, obteniendo los datos de tiempo de carga de cada petición.

Se han realizado pruebas de rendimiento sobre el código sin implementar índices en las entidades y sobre el código implementando los índices.

De esta forma podemos observar el rendimiento general de la aplicación, así como el impacto del uso de índices en el rendimiento de esta.

* **Gráficos de rendimiento y tiempos promedio:**

**Pruebas sin índices:**

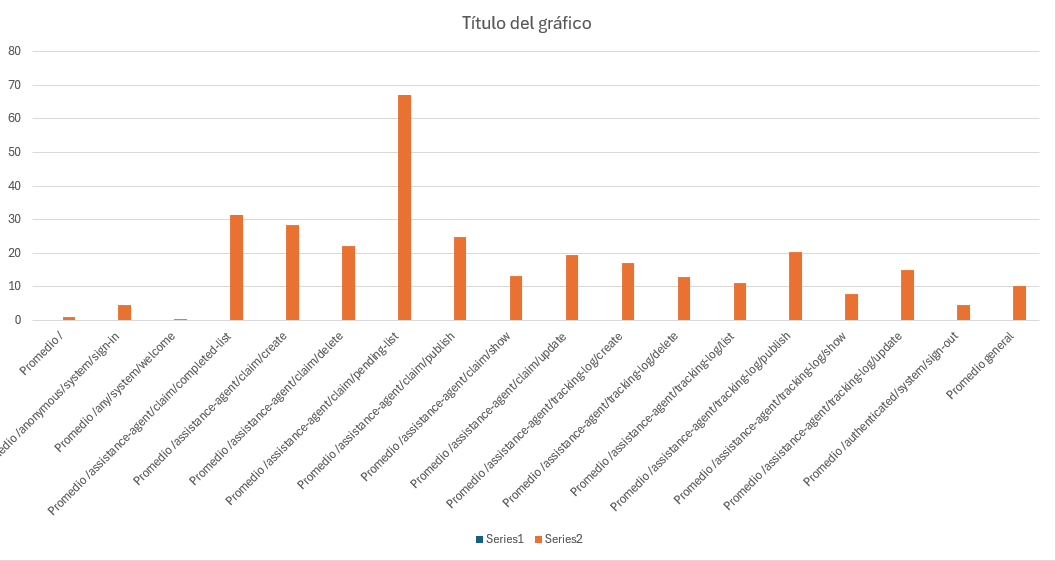
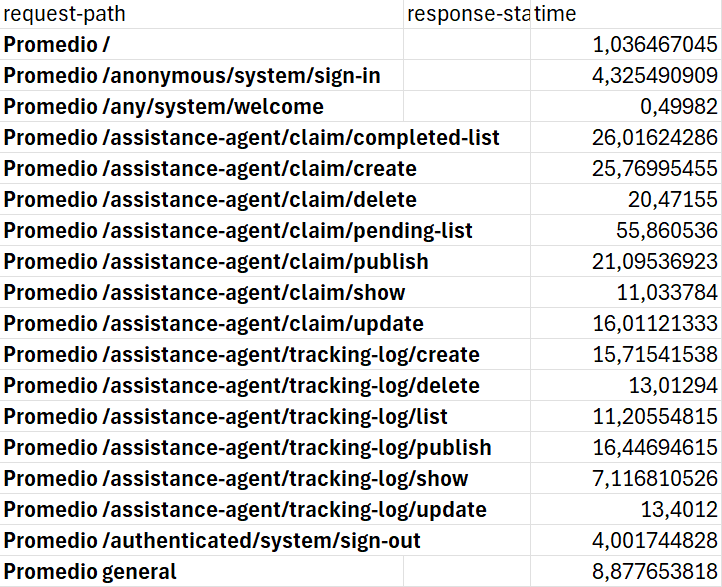
****Imagen 1: Promedios tiempo por instrucción sin índices.(Tiempo en ms)

Imagen 2: Gráfica con los promedios de tiempo por instrucción sin índices. (Tiempo en ms)

**Pruebas con índices:**

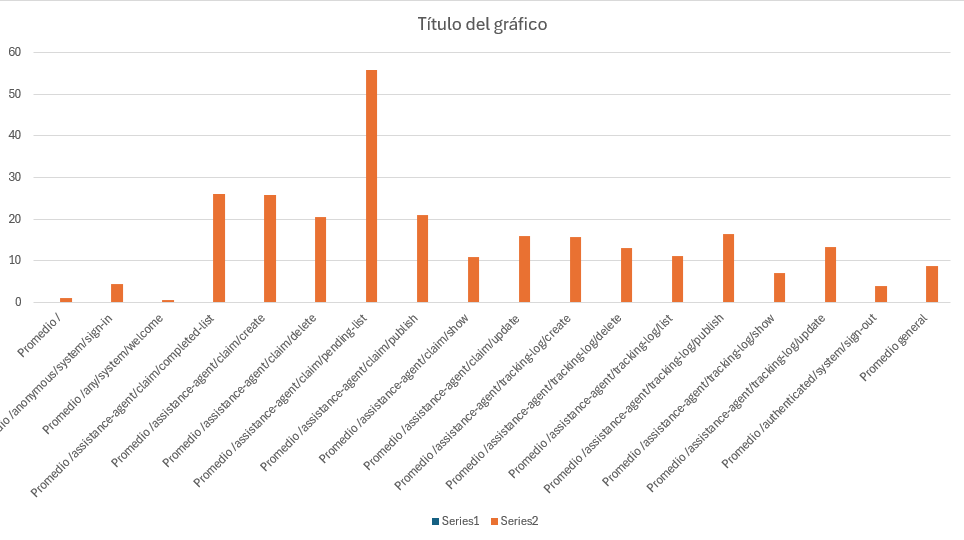
Imagen 3: Promedios tiempo por instrucción con índices. (Tiempo en ms)

Imagen 4: Gráfica con los promedios de tiempo por instrucción con índices. (Tiempo en ms)

Observando las gráficas y los datos obtenemos que la petición mas ineficiente es la de **listar claims incompletas**. Tardando **67** y **55** milisegundos en ambos casos.

Se puede observar también que el uso de índices mejora el rendimiento pasando de un promedio general de **10.16 ms** a uno de **8.88 ms**.

A continuación, se muestran datos estadísticos relacionados con los anteriores:

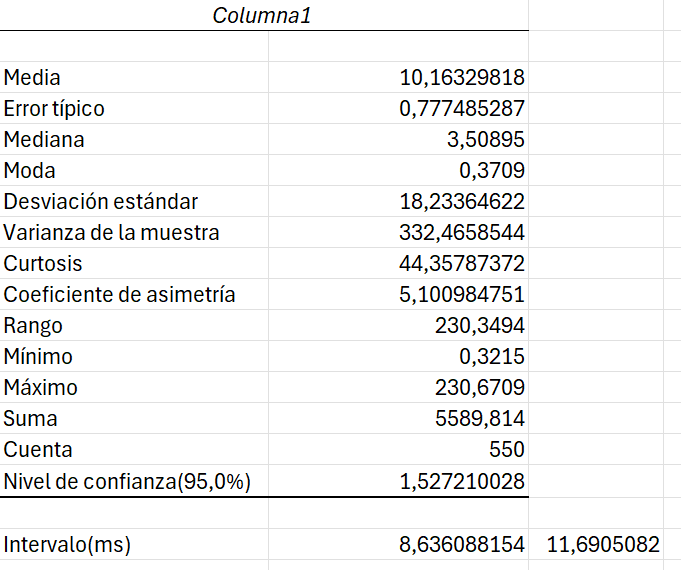
**Pruebas sin índices:**

Imagen 5: Estadisticas de datos de tiempos por instrucción sin índices. (Tiempo en ms)

**Tabla

El contenido generado por IA puede ser incorrecto.Pruebas con índices:**

Imagen 6: Estadisticas de datos de tiempos por instrucción con índices. (Tiempo en ms)

Se puede observar que, aunque los tiempos disminuyan al usar índices, el intervalo de confianza no disminuye significativamente:

* Sin índices: 8.64 – 11.69 ms
* Con índices: 7.49 – 10.27 ms

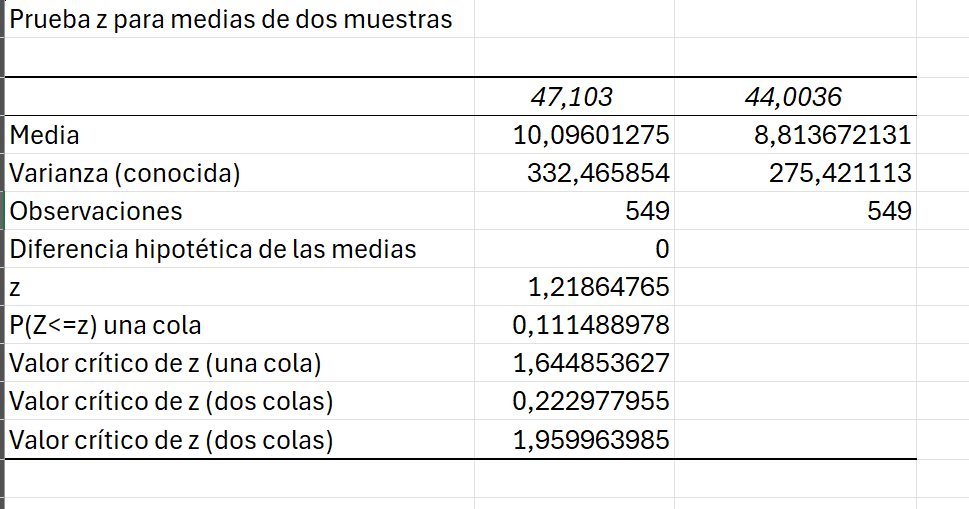
Esto se refuerza con el Z-Test:

Imagen 7: Z-Test de datos de tiempos por instrucción antes y despues de implementar índices (Tiempo en ms)

Como el valor p (columna P(Z<=z) una cola) se encuentra entre (0.05, 1.00), concluimos que los cambios realizados con los índices no resultaron en una mejora significativa de los tiempos.

# **Conclusión:**

Tras la ejecución de las pruebas funcionales, se ha comprobado que el sistema responde correctamente ante los casos esperados y maneja de forma adecuada intentos de uso no autorizado o con datos inválidos.

En cuanto a las pruebas de rendimiento, se ha observado una ligera mejora en los tiempos promedio de respuesta tras la implementación de índices. Sin embargo, los análisis indican que esta mejora no es estadísticamente significativa. Por tanto, se concluye que, aunque la aplicación cumple satisfactoriamente los requisitos funcionales, aún existen oportunidades para optimizar su rendimiento.

.