

# Testing Report



## Diseño y Pruebas II

2024/25

Group: C3.009

GitHub Organization: <https://github.com/DP2-2024-2025-C1-009>

Ricardo Carreño Mariño

riccarmar@alum.us.es

# Índice

<b>Índice</b>	<b>2</b>
<b>Tabla de versiones</b>	<b>2</b>
<b>Introducción</b>	<b>2</b>
<b>Contenido</b>	<b>3</b>
Testing funcional	3
Claim	3
Tracking Log	5
Coverage	7
Testing de rendimiento	8
Antes	8
Después	10
Comparativa con el ordenador de un compañero	12
<b>Conclusiones</b>	<b>14</b>
<b>Bibliografía</b>	<b>14</b>

## Tabla de versiones

Versión	Fecha	Comentarios
1.0	01/07/2025	Primera versión
1.1	03/07/2025	Completar documento
1.2	15/10/2025	Versión para tercera convocatoria

# Introducción

El presente documento constituye el **informe de pruebas** del proyecto, estructurado conforme a las directrices establecidas. Su objetivo es ofrecer una visión clara, organizada y basada en datos sobre las fases de pruebas funcionales y pruebas de rendimiento.

En el **primer capítulo**, se describe en detalle la comprobación funcional del sistema. Los casos de prueba se agrupan por funcionalidad e incluyen una descripción concisa, así como una evaluación de su eficacia a la hora de detectar errores. Estas pruebas se han llevado a cabo de forma sistemática utilizando Eclipse.

En el **segundo capítulo**, se presentan los resultados de las pruebas de rendimiento. Se incluyen gráficos representativos y un análisis estadístico del tiempo de respuesta del sistema con un intervalo de confianza del 95%. También se incluye una comparativa de rendimiento entre dos ordenadores

# Contenido

## Testing funcional

Nota: Para los casos de prueba se ha usado la metodología y los datos recomendados por el profesorado.

### Claim

#### **assistance-agent/claim/create**

Para validar esta funcionalidad, se inició enviando el formulario vacío para comprobar las restricciones básicas. Luego, se ejecutaron pruebas exhaustivas para escenarios positivos y negativos en cada uno de los campos del formulario, especialmente verificando la asociación adecuada con los Tracking Logs correspondientes. Finalmente, se procedió a crear exitosamente un claim con datos válidos y debidamente vinculado a un Tracking Log válido.

En las pruebas de seguridad, se intentó acceder al formulario con roles no autorizados y también se probaron identificadores inválidos. El sistema devolvió correctamente errores de autorización 500 (Non authorised).

No se detectaron fallos durante las pruebas realizadas en la creación y validaciones del formulario de Claim.

#### **assistance-agent/claim/list**

Esta funcionalidad fue verificada accediendo desde un usuario con rol assistance-agent, comprobando que todos los claims aparecen correctamente listados, cumpliendo así con la prueba positiva.

En términos de seguridad, se intentó acceder al listado desde un rol no autorizado, obteniendo correctamente un error 500 (Non authorised).

No se encontraron fallos durante estas pruebas.

#### **assistance-agent/claim/delete**

Se realizaron pruebas eliminando distintos tipos de claims: claims no publicados sin Tracking Logs asociados, claims no publicados con Tracking Logs asociados, y claims con Tracking Logs ya publicados. El sistema devolvió adecuadamente mensajes de validación indicando que no es posible eliminar un claim si su Tracking Log está publicado.

En la prueba de hacking, se intentó eliminar claims inexistentes, claims publicados, y claims pertenecientes a otro assistance-agent. También se probó desde un rol no autorizado. En todos los escenarios se obtuvo correctamente un error 500 (Non authorised).

No se identificaron fallos durante las pruebas de eliminación.

### **assistance-agent/claim/publish**

Para esta funcionalidad se realizaron diversos escenarios: publicación exitosa de un claim con Tracking Log válido y publicado, intento de publicación de claims cuyos Tracking Logs no estaban aún publicados, y publicación de claims sin Tracking Logs asignados, obteniendo las advertencias esperadas.

En las pruebas de seguridad, se intentó publicar claims inexistentes, ya publicados, o pertenecientes a otros assistance-agents, obteniendo correctamente un error 500 (Non authorised). Durante estas pruebas se detectó un fallo que provocaba un error inesperado al modificar valores antes de publicar el claim. Este fallo fue corregido mediante ajustes en los métodos correspondientes del servicio ClaimPublishService.

### **assistance-agent/claim/show**

Para validar esta funcionalidad, se accedió desde el listado a los detalles de claims publicados y no publicados, verificando que se mostraban correctamente junto a sus Tracking Logs relacionados.

Durante las pruebas de hacking, se intentó acceder a claims inexistentes, claims ajenos al assistance-agent autenticado, y desde roles no autorizados. Todas estas pruebas generaron correctamente errores 500 (Non authorised).

No se encontraron fallos en esta funcionalidad.

### **assistance-agent/claim/update**

En la actualización de claims, se verificaron validaciones básicas enviando formularios vacíos y se realizaron pruebas exhaustivas con valores válidos e inválidos en cada campo, logrando actualizar satisfactoriamente un claim vinculado correctamente a un Tracking Log.

Las pruebas de hacking incluyeron intentos de modificar claims inexistentes, publicados, o pertenecientes a otros assistance-agents, además de intentos desde roles no autorizados. En todas estas pruebas se obtuvieron correctamente errores 500 (Non authorised).

No se detectaron fallos durante las pruebas.

## Tracking Log

### **assistance-agent/tracking-log/create**

La validación de esta funcionalidad comenzó enviando formularios vacíos, para luego ejecutar pruebas exhaustivas en todos los escenarios posibles, incluyendo solapamientos y fechas

inválidas en relación con claims existentes. Finalmente, se creó exitosamente un Tracking Log válido.

En las pruebas de seguridad se verificaron los siguientes escenarios:

- Creación de Tracking Logs con claims publicados.
- Uso de identificadores incorrectos o inexistentes.
- Intentos desde roles no autorizados.
- Modificación directa desde consola de desarrolladores de campos críticos.

Todos estos escenarios devolvieron correctamente errores 500 (Non authorised). Las pruebas ayudaron a detectar errores de validación en el método `authorise` del servicio, especialmente en la lógica relacionada con el `draftMode`. Estos errores fueron corregidos satisfactoriamente.

#### **assistance-agent/tracking-log/list**

La funcionalidad fue validada accediendo desde un rol autorizado, verificando que todos los Tracking Logs se mostraban correctamente vinculados a sus respectivos claims.

En pruebas de seguridad se intentó acceder desde un rol no autorizado, obteniendo correctamente un error 500 (Non authorised).

No se hallaron errores en estas pruebas.

#### **assistance-agent/tracking-log/delete**

Se realizaron pruebas eliminando Tracking Logs no publicados asociados a claims.

En las pruebas de seguridad, se intentó eliminar Tracking Logs inexistentes, publicados, pertenecientes a otro assistance-agent y desde roles no autorizados, obteniendo en todos los casos el error 500 (Non authorised).

No se encontraron fallos durante estas pruebas.

#### **assistance-agent/tracking-log/show**

Se verificó esta funcionalidad accediendo a los detalles de Tracking Logs publicados y no publicados desde un rol autorizado, mostrando la información correctamente.

En pruebas de seguridad, se intentó acceder a Tracking Logs inexistentes, de otros assistance-agents o desde roles no autorizados, obteniendo en todos los casos el error 500 (Non authorised).

No se detectaron fallos en esta funcionalidad.

#### **assistance-agent/tracking-log/publish**

La validación incluyó formularios vacíos, pruebas exhaustivas con datos válidos e inválidos, y validaciones específicas de fechas futuras relacionadas con el reloj del sistema.

En pruebas de seguridad, se verificaron intentos de publicar Tracking Logs ya publicados, con identificadores inválidos, desde roles no autorizados y con modificaciones directas desde la consola. Todas estas situaciones devolvieron correctamente errores 500 (Non authorised).

Durante estas pruebas se detectaron y corrigieron errores en la validación del método authorise.

#### **assistance-agent/tracking-log/update**

















La funcionalidad fue validada con formularios vacíos y exhaustivas pruebas con valores válidos e inválidos, comprobando especialmente las restricciones relacionadas con fechas futuras.

En las pruebas de hacking se intentó actualizar Tracking Logs publicados, inexistentes, desde roles no autorizados y con modificaciones directas desde la consola. Todas estas acciones generaron correctamente errores 500 (Non authorised).

















Durante estas pruebas se detectaron errores en la validación del método authorise, que fueron corregidos exitosamente.

## Coverage





A continuación se incluyen capturas de pantalla y explicaciones relacionadas con la cobertura obtenida al realizar las pruebas.

▼ 	acme.features.assistanceAgent.claim		90,6 %
> 	AssistanceAgentClaimDeleteService.java		63,7 %
> 	AssistanceAgentClaimPublishService.java		96,5 %
> 	AssistanceAgentClaimUpdateService.java		96,4 %
> 	AssistanceAgentClaimShowService.java		97,1 %
> 	AssistanceAgentClaimCreateService.java		98,1 %
> 	AssistanceAgentClaimController.java		100,0 %
> 	AssistanceAgentClaimListService.java		100,0 %

He obtenido un **90,6%** en la cobertura de las pruebas relacionadas con Claim por lo que he ejecutado todas las instrucciones disponibles.

▼ 	acme.features.assistanceAgent.trackingLog		90,3 %
> 	AssistanceAgentTrackingLogDeleteService.java		58,6 %
> 	AssistanceAgentTrackingLogPublishService.java		95,2 %
> 	AssistanceAgentTrackingLogShowService.java		93,8 %
> 	AssistanceAgentTrackingLogUpdateService.java		94,9 %
> 	AssistanceAgentTrackingLogCreateService.java		98,1 %
> 	AssistanceAgentTrackingLogListService.java		98,2 %
> 	AssistanceAgentTrackingLogController.java		100,0 %

He obtenido un 90,2 % de cobertura en las pruebas, tanto en Claim como en Tracking Log. No se alcanza un porcentaje mayor porque el método unbind del DeleteService de ambas entidades no llega a ejecutarse durante el flujo normal, por lo que sus instrucciones no se cubren.

> 	ClaimValidator.java		70,1 %
> 	TrackingLogValidator.java		90,6 %

Aquí podemos comprobar la cobertura en las validaciones a nivel de entidad de Claimy de Tracking Log (ClaimValidator y TrackingLogValidator).

No se ha llegado al **100%** de cobertura en estos casos ya que existen validaciones que no se pueden ejecutar probando la aplicación. Estas validaciones están incluidas para comprobar los datos de prueba.



## Testing de rendimiento

Se presentan los resultados de las pruebas de rendimiento. Se incluyen gráficos representativos y un análisis estadístico del tiempo de respuesta del sistema con un intervalo de confianza del **95%**

En primer lugar se ejecutaron los tests sobre el repositorio sin ningún tipo de cambio relacionado con optimizar el rendimiento. Después se corrieron las mismas pruebas pero haciendo uso de **@Index** para tratar de optimizar las consultas del repositorio lo máximo posible (según lo aprendido en el **S02 de la lección 4**)

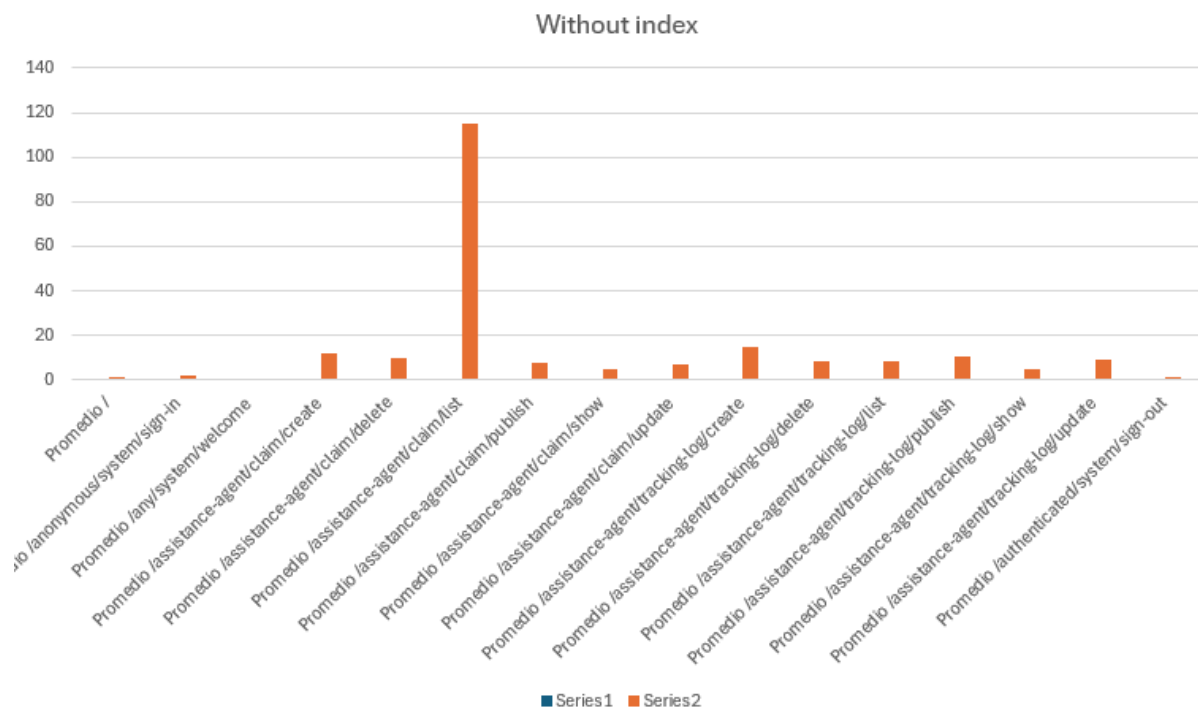
Primero analizaremos los datos de la primera ejecución y luego los compararemos con la segunda

### Antes

Los resultados obtenidos se pueden visualizar en las siguientes imágenes:

<i>Before</i>	
Media	20,4410359
Error típico	2,27521506
Mediana	2,7362495
Moda	2,7516
Desviación estándar	70,7880798
Varianza de la muestra	5010,95224
Curtosis	35,3992186
Coefficiente de asimetría	5,83921496
Rango	656,3253
Mínimo	0,7679
Máximo	657,0932
Suma	19786,9227
Cuenta	968
Nivel de confianza(95,0%)	4,46492806

Estos son los datos estadísticos resumidos obtenidos para la primera prueba sin realizar los índices



En esta gráfica podemos observar el tiempo medio de respuesta por tipo de petición. Podemos comprobar que el **MIR** es la creación de legs "assistance-agent/claim/list". Tras obtener estos datos consulte el servicio de creación de legs para comprobar posibles mejoras a simple vista.

Antes de hacer cambios en el código quería realizar las pruebas habiendo incluidos los índices porque podrían estar muy relacionados con las peticiones que más tardan en ejecutarse.

<b>Promedio /</b>	<b>1,13</b>
<b>Promedio /anonymous/system/sign-in</b>	<b>2,22</b>
<b>Promedio /any/system/welcome</b>	<b>0,82</b>
<b>Promedio /assistance-agent/claim/create</b>	<b>12,31</b>
<b>Promedio /assistance-agent/claim/delete</b>	<b>9,7</b>
<b>Promedio /assistance-agent/claim/list</b>	<b>114,93</b>
<b>Promedio /assistance-agent/claim/publish</b>	<b>7,47</b>
<b>Promedio /assistance-agent/claim/show</b>	<b>5,02</b>
<b>Promedio /assistance-agent/claim/update</b>	<b>6,68</b>
<b>Promedio /assistance-agent/tracking-log/create</b>	<b>14,75</b>
<b>Promedio /assistance-agent/tracking-log/delete</b>	<b>8,67</b>
<b>Promedio /assistance-agent/tracking-log/list</b>	<b>8,42</b>
<b>Promedio /assistance-agent/tracking-log/publish</b>	<b>10,69</b>
<b>Promedio /assistance-agent/tracking-log/show</b>	<b>4,68</b>
<b>Promedio /assistance-agent/tracking-log/update</b>	<b>9,3</b>
<b>Promedio /authenticated/system/sign-out</b>	<b>1,34</b>

BEFORE		
Interval(ms)	15,9761078	24,9059639
Interval (s)	0,01597611	0,02490596

## Después

Tras agregar los índices en las entidades de **Claim** y **Tracking Log** obtuve las siguientes estadísticas. Se muestran en comparación con las pruebas anteriores.

<i>Before</i>		<i>After</i>	
Media	20,4410359	Media	15,71
Error típico	2,27521506	Error típico	0,79710219
Mediana	2,7362495	Mediana	2,1
Moda	2,7516	Moda	2,15
Desviación estándar	70,7880798	Desviación estándar	24,8
Varianza de la muestra	5010,95224	Varianza de la muestra	615,04
Curtosis	35,3992186	Curtosis	12
Coeficiente de asimetría	5,83921496	Coeficiente de asimetría	3
Rango	656,3253	Rango	219,5
Mínimo	0,7679	Mínimo	0,5
Máximo	657,0932	Máximo	220
Suma	19786,9227	Suma	15197,6
Cuenta	968	Cuenta	968
Nivel de confianza(95,0%)	4,46492806	Nivel de confianza(95,0%)	1,56232029

Como podemos observar, ha habido un cambio significativo en los datos estadísticos proporcionados por la columna *time*.

Para comparar adecuadamente los intervalos de confianza calculados tras ejecutar las pruebas, se ha realizado la prueba **Z-Test** sobre las columnas de tiempos generados.

Los datos obtenidos son los siguientes:

Media	20,441	15,71
Varianza (conocida)	5.010,95	615,04
Observaciones	967	967
Diferencia hipotética de las medias	0	
z	1,961402	
P(Z<=z) una cola	0,024916	
Valor crítico de z (una cola)	1,644854	
Valor crítico de z (dos colas)	0,049832	
Valor crítico de z (dos colas)	1,959964	

El valor que nos interesa es  **$P(Z \leq z)$  two-tail**, el cual representa la probabilidad bilateral asociada al estadístico z. En este caso, el valor obtenido ha sido  **$z = 1,9614$** , con  **$P(Z \leq z)$  two-tail = 0,0498**. Dado que este valor se encuentra dentro del intervalo crítico  $(0.00 - \alpha]$ , donde  $\alpha = 1 - \text{nivel de confianza} = 1 - 0,95 = 0,05$ , se cumple que  **$p < 0,05$** . Por tanto, se rechaza la hipótesis nula ( $H_0$ ) y se acepta la hipótesis alternativa ( $H_1$ ), concluyendo que los tiempos medios de ejecución con índices son significativamente menores que los obtenidos sin índices.

El resultado demuestra que la inclusión de índices en las entidades Java ha producido una mejora real en el rendimiento, evidenciando que la reducción observada en los tiempos de respuesta no se debe al azar. Además, la desviación estándar más baja en las pruebas con índices refleja una ejecución más estable y consistente. En términos prácticos, esto se traduce en una mejora aproximada del 23 % en el rendimiento promedio del sistema.

Las pruebas se realizaron en un entorno más optimizado, tanto a nivel de software como de hardware. El equipo empleado contaba con un **procesador AMD Ryzen 7 7800X3D** (8 núcleos, 16 hilos, frecuencia base de 4,2 GHz y turbo hasta 5,0 GHz), **32 GB DDR5 6000 MHz Corsair Vengeance RGB**, y un **SSD NVMe Gen 4 de 1 TB** con velocidades de lectura y escritura de 7000 MB/s y 6800 MB/s, respectivamente.

En conjunto, estos resultados confirman que la optimización mediante índices y un entorno de ejecución más potente contribuyen a una mejora estadísticamente significativa en la eficiencia del sistema. Por tanto, se puede afirmar que las optimizaciones aplicadas han logrado el objetivo de reducir los tiempos de respuesta y aumentar la estabilidad de las operaciones, situando el rendimiento dentro de los niveles esperados para una configuración de hardware moderna y un modelo de datos correctamente indexado.

## Comparativa con el ordenador de un compañero

Este apartado consiste en comparar y analizar las diferencias entre el rendimiento obtenido desde dos máquinas distintas. En esta ocasión haré la comparativa con el ordenador de mi compañero de grupo Jaime.

Para la comparativa he usado la versión con índices.

**Ordenador personal:** AMD Ryzen 7 7800X3D, 32 GB RAM, 1 TB SSD

**Ordenador de Jaime:** Ordenador de Jaime: Intel Core i7-1165G7, 16 GB RAM, 512 GB SSD

<i>Ricardo</i>		<i>Jaime</i>	
Media	15,71	Media	18,32
Error típico	0,79710219	Error típico	0,9642
Mediana	2,1	Mediana	2,4
Moda	2,15	Moda	2,45
Desviación estándar	24,8	Desviación estándar	30
Varianza de la muestra	615,04	Varianza de la muestra	900
Curtosis	12	Curtosis	20
Coefficiente de asimetría	3	Coefficiente de asimetría	4
Rango	219,5	Rango	300
Mínimo	0,5	Mínimo	0,62
Máximo	220	Máximo	300,62
Suma	15197,6	Suma	17,733,76
Cuenta	968	Cuenta	968
Nivel de confianza(95,0%)	1,56232029	Nivel de confianza(95,0%)	1,8899

Media	18,32	15,71
Varianza (conocida)	900,00	615,04
Observaciones	967	967
Diferencia hipotética de las medias	0	
z	2,0853	
P(Z<=z) una cola	0,0185	
Valor crítico de z (una cola)	1,644854	
Valor crítico de z (dos colas)	0,0371	
Valor crítico de z (dos colas)	1,959964	

El valor de referencia para esta prueba es  **$P(Z \leq z)$  two-tail**, que determina la probabilidad bilateral asociada al estadístico z. En esta ocasión, se ha obtenido un valor de  **$z = 2,0853$** , con

**$P(Z \leq z)$  two-tail = 0,0371**. Dado que dicho valor se encuentra dentro del intervalo crítico  $(0.00 - \alpha]$ , siendo  $\alpha = 1 - 0,95 = 0,05$ , se cumple que  **$p < 0,05$** . Por tanto, se rechaza la hipótesis nula ( $H_0$ ) y se acepta la hipótesis alternativa ( $H_1$ ), lo que indica que existen **diferencias estadísticamente significativas** entre el rendimiento del equipo de Jaime y el de mi entorno con índices.

En concreto, el **ordenador de Jaime (Intel Core i7-1165G7, 16 GB RAM, 512 GB SSD)** mostró un rendimiento inferior en comparación con mi equipo, que cuenta con un **procesador AMD Ryzen 7 7800X3D, 32 GB DDR5 6000 MHz y SSD NVMe Gen 4**. Esta diferencia de hardware, unida a la **implementación de índices en las entidades Java**, explica la reducción de los tiempos medios de ejecución observada en mi entorno. Los resultados obtenidos reflejan que la optimización mediante índices, combinada con una mayor capacidad de cómputo, ha mejorado significativamente la eficiencia en la ejecución de consultas y operaciones sobre la base de datos.

Además, el **valor de  $p = 0,0371$**  confirma que la mejora en el rendimiento no es atribuible al azar, sino a un cambio real en las condiciones de ejecución. Mientras el equipo de Jaime presentó una **media de 18,32 ms**, el mío con índices alcanzó una **media de 15,71 ms**, lo que supone una mejora aproximada del **14,3 %** en los tiempos promedio. Asimismo, la menor **varianza (615,04 frente a 900,00)** en mi entorno evidencia una mayor estabilidad en los resultados.

En conclusión, los datos estadísticos y el resultado del z-test respaldan que la optimización mediante índices y el uso de un hardware de mayor rendimiento producen una mejora **pequeña pero significativa** en la velocidad y consistencia del sistema, situando su comportamiento dentro del intervalo de confianza del 95 % y confirmando la efectividad de las optimizaciones aplicadas.

## Conclusiones

Tras realizar un exhaustivo conjunto de pruebas funcionales, de validación de campos y de hacking sobre todas las funcionalidades relacionadas con la entidad **Claim** y **Tracking Log**, se ha verificado que el sistema se comporta conforme a los requisitos establecidos, tanto en escenarios válidos como ante intentos de acceso o manipulación no autorizada. Las pruebas han permitido identificar y corregir algunos errores y bugs importantes que podrían haber causado errores de funcionalidad y seguridad importantes.

En cuanto al apartado de rendimiento, se ha conseguido aumentar el rendimiento de las consultas haciendo uso de índices y analizando los resultados obtenidos. También hemos podido comprobar que el rendimiento depende mucho del ordenador que se utilice.

## Bibliografía

- S02 - Performance testing (L04)
- Documento de anexos de la asignatura, "06- Anexes"