

Universidad de Sevilla

Escuela Técnica Superior de Ingeniería Informática



Grado en Ingeniería Informática – Ingeniería del Software

Diseño y Pruebas 2

Curso 2024 – 2025

## **Informe de pruebas**

### **D04 – Testing**

**Alberto Carmona Sicre – Student 4**

# Informe de pruebas

## 1. Información general

Fecha: 15/10/2025

Grupo: C3.033

Repositorio: <https://github.com/DP2-C1-035/Acme-ANS-C3>

Autor: Alberto Carmona Sicre (albcarsic@alum.us.es)

## 2. Tabla de contenidos

### Contenido

1.	Información general .....	2
2.	Tabla de contenidos.....	2
3.	Resumen ejecutivo .....	2
4.	Tabla de revisiones .....	3
5.	Introducción .....	3
6.	Contenidos .....	4
6.1	Testing funcional.....	4
6.1.1	Claim.....	4
6.1.2	Tracking Log .....	9
6.2	Testing de rendimiento.....	14
6.2.1	Análisis de rendimiento .....	14
6.2.2	Intervalo de confianza .....	18
6.2.3	Hypothesis contrast .....	18
7.	Conclusiones .....	19
8.	Bibliografía .....	19

## 3. Resumen ejecutivo

En este documento se recoge toda la información relevante del testing llevado a cabo por el Student 4 en el proyecto DP2-C3-03, detallando el plan seguido a la hora de realizar los casos de prueba. También se detalla el rendimiento obtenido al repetir los tests sin índices y con índices y una comparativa estadística sobre el rendimiento en cada caso.

#### 4. Tabla de revisiones

Número de revisión	Descripción	Fecha
Revisión 1	Creación del documento y adición de todos los apartados	15/10/2025

#### 5. Introducción

En el presente documento se describe la metodología de testing que ha usado el Student 4 durante el entregable D04.

El documento contiene un resumen ejecutivo que proporciona una visión clara de los puntos más importantes del documento, así como la información necesaria para entenderlo. Además, el documento cuenta con una lista de versiones como se puede comprobar en la tabla de versiones.

En cuanto al contenido de este, está dividido en dos capítulos, el primero con varios apartados que se corresponden con la cobertura total lograda en el código y casos de prueba para las distintas features implementadas. En el segundo apartado se llevará a cabo un análisis de rendimiento del código realizados en diferentes circunstancias

## 6. Contenidos

### 6.1 Testing funcional

#### 6.1.1 Claim

##### **AssistanceAgentClaimListCompletedService**

**Safe testing:** para el testing de este servicio solo era necesario listar las reclamaciones completadas de un agente de asistencia. No había forma de realizar un test negativo.

**Hacking:** no se realizaron pruebas.

##### **AssistanceAgentClaimListUndergoingService**

**Safe testing:** para el testing de este servicio solo era necesario listar las reclamaciones pendientes de un agente de asistencia. No había forma de realizar un test negativo.

**Hacking:** se realizaron dos pruebas:

- Listado de reclamaciones de un agente de servicio sin haber iniciado sesión (Role: Anonymous).
- Listado de reclamaciones de un agente de servicio con rol incorrecto (Role: Consumer).

**Bugs:** sin bugs.

##### **AssistanceAgentClaimShowService**

**Safe testing:** para el testing de este servicio se muestra una única reclamación de un agente de servicio como caso de prueba positivo. No era posible realizar un caso de prueba negativo.

**Hacking:** se realizaron cuatro pruebas:

- Mostrar la reclamación de un agente de servicio que no ha iniciado sesión (Role: Anonymous).
- Mostrar la reclamación de un agente de servicio con un rol incorrecto (Role: Consumer).
- Mostrar una reclamación de un agente de servicio que no es poseedor de dicho reclamación (Role: Assistance Agent).

**Bugs:** sin bugs.

##### **AssistanceAgentClaimCreateService**

**Safe testing:**

- **Casos de prueba positivos:** crear una reclamación correcta.
- **Casos de prueba negativos:** creación de reclamaciones probando todas las restricciones de los campos:

- Formulario vacío.

- Campos con valores incorrectos, intentando probar todas las restricciones especificadas tanto por las anotaciones como por la lógica del método validate (por ejemplo, códigos duplicados, divisas no permitidas, presupuestos negativos, fechas incorrectas, etc.)

**Hacking:** la única forma de hackear este servicio era intentar entrar al formulario de creación de reclamaciones sin haber iniciado sesión (Role: Anonymous) o con un rol incorrecto (Role distinto a Assistance Agent).

**Bugs:** hubo que arreglar un problema por el cual se podían elegir Legs que aun no habían acabado.

## **AssistanceAgentClaimUpdateService**

### **Safe testing:**

- **Casos de prueba positivos:** actualizar una reclamación de manera correcta.

- **Casos de prueba negativos:** actualización de reclamaciones probando todas las restricciones de los campos:

- Formulario vacío

- Campos con valores incorrectos, intentando probar todas las restricciones especificadas tanto por las anotaciones como por la lógica del método validate (se siguieron los mismos casos que en el servicio de creación de reclamaciones).

**Hacking:** la única forma de hackear este servicio con un rol incorrecto era copiar la URL de actualización de una reclamación e intentar ejecutarla, lo que resultaba en una vista de pánico.

En cambio, con rol correcto, pero ejecutando acciones ilegales se probó lo siguiente:

- Actualizar una reclamación publicada (acción ilegal).

- Actualizar una reclamación de otro agente de servicio (acción ilegal).

- Tratar de actualizar los componentes readOnly y Navigation mediante hacking. El resultado fue que se ignoraban los hackeos.

**Bugs:** sin bugs.

## **AssistanceAgentClaimDeleteService**

### **Safe testing:**

- **Casos de prueba positivos:** eliminar reclamación.

- **Casos de prueba negativos:** no existen casos de prueba negativos.

**Hacking:** la única forma de hackear este servicio con un rol incorrecto era copiar la URL de eliminación de una reclamación e intentar ejecutarla, lo que resultaba en una vista de pánico. En cambio, con rol correcto, pero ejecutando acciones ilegales, se probó lo siguiente:

- Eliminar una reclamación publicada (acción ilegal).
- Eliminar una reclamación de otro agente de servicio (acción ilegal).

**Bugs:** sin bugs.

## **AssistanceAgentClaimPublishService**

### **Safe testing:**

- **Casos de prueba positivos:** publicar una reclamación válida.
- **Casos de prueba negativos:** publicar contratos probando las restricciones de los campos:
  - Formulario vacío.

- Campos con valores incorrectos, intentando probar todas las restricciones especificadas tanto por las anotaciones como por la lógica del método validate (por ejemplo, códigos duplicados, divisas no permitidas, presupuestos negativos, fechas incorrectas, etc.)



















**Hacking:** la única forma de hackear este servicio con un rol incorrecto era copiar la URL de publicación de una reclamación e intentar ejecutarla, lo que resultaba en una vista de pánico. En cambio, con rol correcto, pero ejecutando acciones ilegales se probó lo siguiente:

- Publicar una reclamación de otro agente de servicio, tanto publicado como sin publicar (acción ilegal).
- Publicar una reclamación ya publicada del agente de servicio.
- Tratar de actualizar los componentes readOnly y Navigation mediante hacking. El resultado fue que se ignoraban los hackeos.

**Bugs:** sin bugs.

## Cobertura Claim

A continuación, se muestra el porcentaje de cobertura de sentencias de todos los servicios de la entidad Claim:

 <b>acme.features.assistanceAgent.claim</b>	 89,4 %
>  <b>AssistanceAgentClaimDeleteService.java</b>	 59,5 %
>  <b>AssistanceAgentClaimPublishService.java</b>	 91,9 %
>  <b>AssistanceAgentClaimCreateService.java</b>	 92,5 %
>  <b>AssistanceAgentClaimUpdateService.java</b>	 96,1 %
>  <b>AssistanceAgentClaimListPendingService.java</b>	 92,8 %
>  <b>AssistanceAgentClaimListCompletedService.java</b>	 94,0 %
>  <b>AssistanceAgentClaimController.java</b>	 100,0 %
>  <b>AssistanceAgentClaimShowService.java</b>	 100,0 %

**AssistanceAgentClaimDeleteService: 59.5.4%**

**AssistanceAgentClaimPublishService: 91.9%**

**AssistanceAgentClaimCreateService: 92.5%**

**AssistanceAgentClaimUpdateService: 96.1%**

**AssistanceAgentClaimListPendingService: 92.8%**

**AssistanceAgentClaimListCompletedService: 94.0%**

**AssistanceAgentClaimShowService: 100%**

Siendo la cobertura total del paquete `acmé.features.assistanceAgent.claim` un 89.4%.

En términos generales, la cobertura de código de los servicios de la entidad Assistance Agentship es bastante satisfactoria. Las únicas líneas de código no cubiertas se deben a restricciones quizás demasiado estrictas en la autorización de los métodos, las cuales consideran casos que no pueden probarse sin el uso de herramientas externas, como Postman. Debido a esto, algunas ramas del código no se alcanzan, al igual que ciertas líneas encargadas de la internacionalización al español.

También una línea de código que encontramos en todos los métodos:

```
assert object != null
```

Esta línea de código no se llega a cubrir del todo en ningún método, pero está presente en todas las plantillas de los servicios proporcionados en la metodología de la asignatura y se consultó con el Consumere y no se consideró necesario cubrirla del todo.

Caso especial: **AssistanceAgentClaimShowService**. Todo el método `unbind` de este servicio no se llega a cubrir, al no poder realizar pruebas negativas que resulten en errores de lógica de negocio, como se explicó anteriormente. Esto se debe a que el método `unbind` se encargaría de

reconstruir la vista en caso de que se produzca un error, pero al no poder producirse, no se llega a ejecutar.



## 6.1.2 Tracking Log

### **AssistanceAgentTrackingLogListService**

**Safe testing:** para el testing de este servicio solo era necesario listar los registros de seguimiento de una reclamación de un agente de servicio. No había forma de realizar un test negativo.

**Hacking:** se realizaron las siguientes pruebas:

- Listado de registros de seguimiento de una reclamación de un agente de servicio sin haber iniciado sesión (Role: Anonymous).
- Listado de registros de seguimiento de una reclamación de un agente de servicio con rol incorrecto (Role: Consumer).
- Listado de registros de seguimiento de una reclamación de un agente de servicio que no es poseedor de tal reclamación.

**Bugs:** sin bugs.

### **AssistanceAgentTrackingLogShowService**

**Safe testing:** para el testing de este servicio se muestra un registro de seguimiento de una reclamación de un agente de servicio como caso de prueba positivo. No era posible realizar un caso de prueba negativo.

**Hacking:** se realizaron las siguientes pruebas:

- Mostrar un registro de seguimiento de una reclamación de un agente de servicio que no ha iniciado sesión (Role: Anonymous).
- Mostrar un registro de seguimiento de una reclamación de un agente de servicio con un rol incorrecto (Role: Consumer).
- Mostrar un registro de seguimiento de una reclamación de un agente de servicio que no es poseedor de dicha reclamación (Role: Assistance Agent).

**Bugs:** sin bugs.

### **AssistanceAgentTrackingLogCreateService**

**Safe testing:**

- **Casos de prueba positivos:** crear un registro de seguimiento válido.
- **Casos de prueba negativos:** creación de registros de seguimiento probando todas las restricciones de los campos:
  - Formulario vacío.

- Campos con valores incorrectos, intentando probar todas las restricciones especificadas tanto por las anotaciones como por la lógica del método validate (por ejemplo, códigos duplicados, divisas no permitidas, presupuestos negativos, fechas incorrectas, etc.)

**Hacking:** la única forma de hackear este servicio era intentar entrar al formulario de creación de registros de seguimiento sin haber iniciado sesión (Role: Anonymous), con un rol incorrecto (Role distinto a Assistance Agent) o con rol correcto, pero con creación de un registro de seguimiento en una reclamación de un agente de servicio que no es poseedor de dicha reclamación.

**Bugs:** sin bugs.

## **AssistanceAgentTrackingLogCreateExceptionService**

### **Safe testing:**

- **Casos de prueba positivos:** crear registros de seguimiento excepcional válido.

- **Casos de prueba negativos:** creación de registros de seguimiento probando todas las restricciones de los campos:

- Formulario vacío.

- Campos con valores incorrectos, intentando probar todas las restricciones especificadas tanto por las anotaciones como por la lógica del método validate (por ejemplo, códigos duplicados, divisas no permitidas, presupuestos negativos, fechas incorrectas, etc.)

**Hacking:** la única forma de hackear este servicio era intentar entrar al formulario de creación de registros de seguimiento sin haber iniciado sesión (Role: Anonymous), con un rol incorrecto (Role distinto a Assistance Agent) o con rol correcto, pero con creación de un registro de seguimiento en una reclamación de un agente de servicio que no es poseedor de dicha reclamación.

- Tratar de actualizar los componentes readOnly y Navegation mediante hacking. El resultado fue que se ignoraban los hackeos.

**Bugs:** hubo que refinar la lógica para que se pudiesen crear dos registros de seguimiento que compartiesen el mismo porcentaje de resolución. Al final, se optó por eliminar la opción de crear un registro como excepción una vez se crea, y volver a poder crearlo si se elimina.

## **AssistanceAgentTrackingLogUpdateService**

### **Safe testing:**

- **Casos de prueba positivos:** actualizar un registro de seguimiento con datos válidos.

- **Casos de prueba negativos:** actualización de registros de seguimiento probando todas las restricciones de los campos:

- Formulario vacío

- Campos con valores incorrectos, intentando probar todas las restricciones especificadas tanto por las anotaciones como por la lógica del método validate (se siguieron los mismos casos que en el servicio de creación de registros de seguimiento).

**Hacking:** la única forma de hackear este servicio con un rol incorrecto era copiar la URL de actualización de un registro de seguimiento e intentar ejecutarla, lo que resultaba en una vista de pánico.

En cambio, con rol correcto, pero ejecutando acciones ilegales, se probó lo siguiente:

- Actualizar un registro de seguimiento publicado (acción ilegal).
- Actualizar un registro de seguimiento de un agente de servicio con un rol incorrecto (Role: Consumer).
- Actualizar un registro de seguimiento de una reclamación de un agente de servicio que no es poseedor de dicha reclamación (Role: Assistance Agent).
- Tratar de actualizar los componentes readOnly y Navigation mediante hacking. El resultado fue que se ignoraban los hackeos.

**Bugs:** sin bugs.

## **AssistanceAgentTrackingLogDeleteService**

**Safe testing:**

- **Casos de prueba positivos:** eliminar un registro de seguimiento.
- **Casos de prueba negativos:** no existen casos de prueba negativos.

**Hacking:** la única forma de hackear este servicio con un rol incorrecto era copiar la URL de eliminación de un registro de seguimiento e intentar ejecutarla, lo que resultaba en una vista de pánico. En cambio, con rol correcto, pero ejecutando acciones ilegales se probó lo siguiente:

- Eliminar un registro de seguimiento publicado (acción ilegal).
- Eliminar un registro de seguimiento de otro agente de servicio (acción ilegal).

**Bugs:** sin bugs.

## **AssistanceAgentTrackingLogPublishService**

**Safe testing:**

- **Casos de prueba positivos:** publicar varios registros de seguimiento válidos, para completar todo un proceso de publicación de estos.
- **Casos de prueba negativos:** publicar registros de seguimiento probando las restricciones de los campos:
  - Formulario vacío.

- Campos con valores incorrectos, intentando probar todas las restricciones especificadas tanto por las anotaciones como por la lógica del método validate. Se siguieron los mismos casos que en el servicio de creación y actualización de reclamaciones.



















**Hacking:** la única forma de hackear este servicio con un rol incorrecto era copiar la URL de publicación de una reclamación e intentar ejecutarla, lo que resultaba en una vista de pánico. En cambio, con rol correcto, pero ejecutando acciones ilegales se probó lo siguiente:

- Publicar un registro de seguimiento ya publicado (acción ilegal).
- Publicar un registro de seguimiento de otro agente de servicio (acción ilegal).
- Tratar de actualizar los componentes readOnly y Navigation mediante hacking. El resultado fue que se ignoraban los hackeos.

**Bugs:** el principal bug fue tener en cuenta la excepción, de nuevo. Antes, al intentar publicar un registro cuando el porcentaje era el mismo, daba error, ahora no falla SOLO cuando se desea publicar un registro con resolución al 100% (la excepción).

## Cobertura Tracking Log

A continuación, se muestra el porcentaje de cobertura de sentencias de todos los servicios de la entidad Tracking Log:

 acme.features.assistanceAgent.trackingLog	 93,8 %
>  AssistanceAgentTrackingLogUpdateService.java	 89,0 %
>  AssistanceAgentTrackingLogDeleteService.java	 65,3 %
>  AssistanceAgentTrackingLogCreateService.java	 96,3 %
>  AssistanceAgentTrackingLogPublishService.java	 97,9 %
>  AssistanceAgentTrackingLogCreateExceptionService.java	 99,3 %
>  AssistanceAgentTrackingLogListService.java	 98,9 %
>  AssistanceAgentTrackingLogShowService.java	 98,5 %
>  AssistanceAgentTrackingLogController.java	 100,0 %

**AssistanceAgentTrackingLogUpdateService: 89.0%**

**AssistanceAgentTrackingLogDeleteService: 65.3%**

**AssistanceAgentTrackingLogCreateService: 96.3%**

**AssistanceAgentTrackingLogPublishService: 97.9%**

**AssistanceAgentTrackingLogCreateExceptionService: 99.3%**

**AssistanceAgentTrackingLogListService: 98.9%**

**AssistanceAgentTrackingLogShowService: 98.5%**

Siendo la cobertura total del paquete `acme.features.AssistanceAgent.trackingLog` un 93.8%.

En términos generales, la cobertura de código de los servicios de la entidad Tracking Log es bastante satisfactoria. Las únicas líneas de código no cubiertas se deben a restricciones quizás demasiado estrictas en la autorización de los métodos, las cuales consideran casos que no pueden probarse sin el uso de herramientas externas, como Postman. Debido a esto, algunas ramas del código no se alcanzan, al igual que ciertas líneas encargadas de la internacionalización al español.

También una línea de código que encontramos en todos los métodos:

```
assert object != null
```

Esta línea de código no se llega a cubrir del todo en ningún método, pero está presente en todas las plantillas de los servicios proporcionados en la metodología de la asignatura y se consultó con el Consumere y no se consideró necesario cubrirla del todo.

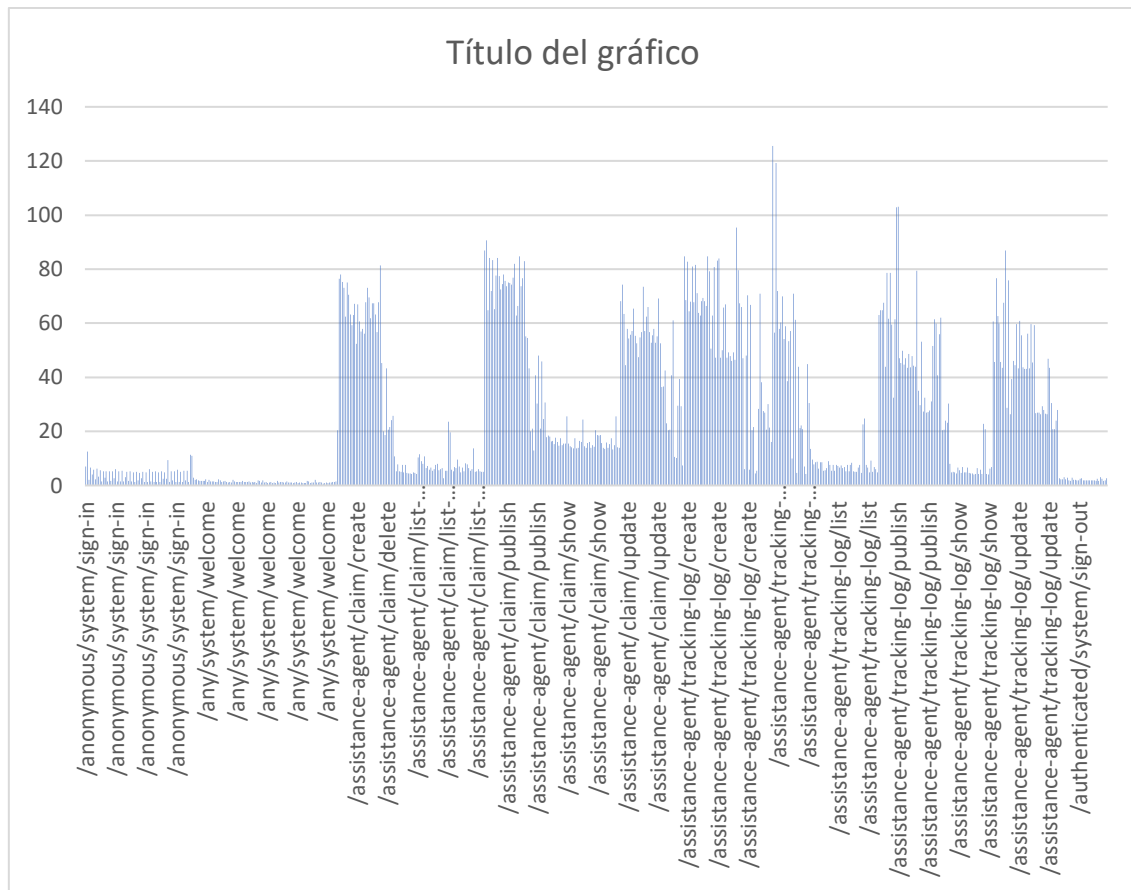
Caso especial: **AssistanceAgentTrackingLogDeleteService**. Todo el método `unbind` de este servicio no se llega a cubrir, al no poder realizar pruebas negativas que resulten en errores de lógica de negocio, como se explicó anteriormente. Esto se debe a que el método `unbind` se encargaría de reconstruir la vista en caso de que se produzca un error, pero al no poder producirse, no se llega a ejecutar.

## 6.2 Testing de rendimiento

### 6.2.1 Análisis de rendimiento

Después de ejecutar las pruebas y recopilar los datos sobre las solicitudes realizadas, se realizó un análisis de los resultados obtenidos en términos de rendimiento de la aplicación, tomando como referencia el tiempo de respuesta de los servicios. Nos enfocamos en los servicios de las clases Claim y Tracking Log. A continuación, se presentan los datos promedio de cada solicitud; además, para mayor claridad, también se incluye un gráfico de barras.

Promedio /	3.145027869
Promedio /anonymous/system/sign-in	3.906197015
Promedio /any/system/welcome	1.446759783
Promedio /assistance-agent/claim/create	61.11605484
Promedio /assistance-agent/claim/delete	27.38955
Promedio /assistance-agent/claim/list-completed	6.94285625
Promedio /assistance-agent/claim/list-undergoing	7.484285714
Promedio /assistance-agent/claim/publish	64.91348333
Promedio /assistance-agent/claim/show	17.352422
Promedio /assistance-agent/claim/update	49.19936667
Promedio /assistance-agent/tracking-log/create	57.349058
Promedio /assistance-agent/tracking-log/delete	27.47913333
Promedio /assistance-agent/tracking-log/exceptional-case	47.235188
Promedio /assistance-agent/tracking-log/list	7.726716279
Promedio /assistance-agent/tracking-log/publish	49.05446667
Promedio /assistance-agent/tracking-log/show	6.441785185
Promedio /assistance-agent/tracking-log/update	44.15119524
Promedio /authenticated/system/sign-out	2.172912903
Promedio general	24.02839194



Como vemos en la gráfica, existen grandes diferencias entre los tiempos de respuesta de las peticiones de inicio de sesión o bienvenido, comparado con las funcionalidades de ambas entidades que estamos analizando. También es destacable que la funcionalidad de mayor tiempo y, por tanto, de mayor ineficiencia, es la de publicar de la entidad Claim, seguida de la de crear de la misma, siendo la primera unas seis veces mayor que, por ejemplo, la de listar reclamaciones completadas, siendo esta una de las más pequeñas. El promedio general es de 24.03 ms.

La identificación de los servicios más lentos sirvió como punto de partida para optimizar la aplicación. Se implementaron una serie de cambios en el código, específicamente añadiendo índices a las tablas de la base de datos de las entidades Claim y Tracking Log, con el objetivo de mejorar el rendimiento de las solicitudes. A continuación, se presenta un gráfico de barras con los resultados obtenidos después de la optimización y los nuevos promedios de tiempo de respuesta.

Promedio /	3.00205
Promedio /anonymous/system/sign-in	3.62256
Promedio /any/system/welcome	1.39763
Promedio /assistance-agent/claim/create	63.404
Promedio /assistance-agent/claim/delete	26.511
Promedio /assistance-agent/claim/list-completed	6.2375
Promedio /assistance-agent/claim/list-undergoing	7.12902
Promedio /assistance-agent/claim/publish	64.9086
Promedio /assistance-agent/claim/show	16.2042
Promedio /assistance-agent/claim/update	46.4171
Promedio /assistance-agent/tracking-log/create	54.9423
Promedio /assistance-agent/tracking-log/delete	27.4922
Promedio /assistance-agent/tracking-log/exceptional-case	48.6133
Promedio /assistance-agent/tracking-log/list	8.30083
Promedio /assistance-agent/tracking-log/publish	48.8955
Promedio /assistance-agent/tracking-log/show	6.05762
Promedio /assistance-agent/tracking-log/update	41.3845
Promedio /authenticated/system/sign-out	2.01404
Promedio general	23.5236





### 6.2.2 Intervalo de confianza

Se analizó el intervalo de confianza de los tiempos de respuesta de las peticiones de la aplicación, con el objetivo de comprobar si la aplicación cumple con el requisito del rendimiento establecido: que el tiempo de respuesta de las peticiones no supere el segundo de media. Para ello se realizó un análisis de los datos obtenidos en los tests, y se calculó el intervalo de confianza de los tiempos de respuesta de las peticiones, con un nivel de confianza del 95%. La herramienta utilizada para el análisis ha sido Excel, que proporciona un complemento llamado Herramientas para el análisis. A continuación, se muestra el resultado obtenido:

Before	After						
63.8502	66.618						
3.0842	3.0457						
4.5595	4.4216						
2.2445	2.6685	Media	24.0283919			Media	23.5235721
3.702	3.424	Error típico	1.01878502			Error típico	1.00634926
2.0079	2.0272	Mediana	7.8658			Mediana	7.6307
6.0245	5.9897	Moda	2.7247			Moda	1.8464
1.5107	1.4475	Desviación es	27.0889555			Desviación es	26.7582951
3.2764	3.9674	Varianza de la	733.811508			Varianza de la	716.006359
1.683	1.4523	Curtosis	-0.21673959			Curtosis	-0.08992486
3.1079	3.0412	Coefficiente de	1.01785325			Coefficiente de	1.05799098
1.3082	2.6779	Rango	124.694			Rango	110.7016
3.0871	3.231	Mínimo	0.7745			Mínimo	0.766
1.2231	1.2553	Máximo	125.4685			Máximo	111.4676
2.9229	2.8008	Suma	16988.0731			Suma	16631.1655
1.2092	1.7706	Cuenta	707			Cuenta	707
3.6543	2.8546	Nivel de confia	2.00021101			Nivel de confia	1.97579551
1.2603	1.2137	Interval (ms)	22.0281809	26.0286029		Interval (ms)	21.5477766
2.5237	3.5638	Interval (s)	0.02202818	0.0260286		Interval (s)	0.02154778
1.3932	1.1428						0.02549937
3.4475	2.5581						
1.7836	1.8645						
3.3405	2.4858						
1.2231	1.2553						

### 6.2.3 Hypothesis contrast

Prueba z para medias de dos muestras		
	Before	After
Media	24.02839194	23.52357214
Varianza (conocida)	733.811508	716.006359
Observaciones	707	707
Diferencia hipotética de las medias	0	
z	0.352524549	
P(Z<=z) una cola	0.362222456	
Valor crítico de z (una cola)	1.644853627	
Valor crítico de z (dos colas)	0.724444911	
Valor crítico de z (dos colas)	1.959963985	

Como podemos observar, el valor crítico de  $z$  (dos colas) es 0.724.... Para saber si los cambios han sido significativos, hemos de comparar el valor  $z$  con  $\alpha$ , que en nuestro caso es 0,05. En este caso, nuestro  $z$  se encuentra por encima del intervalo  $[0.00, \alpha)$ , por lo que podemos afirmar que, aunque los cambios han conseguido rebajar el promedio general de las peticiones, tal y como muestran las gráficas y los datos, globalmente no se han podido conseguir mejoras significativas.

## 7. Conclusiones

A lo largo del documento, se ha realizado un análisis exhaustivo de los servicios de las clases Claim y TrackingLog, que incluyó pruebas funcionales y de rendimiento para verificar el correcto funcionamiento de los servicios y el rendimiento de la aplicación. En los tests funcionales, se llevaron a cabo pruebas tanto positivas como negativas, confirmando que los servicios funcionan adecuadamente y no presentan errores. En los tests de rendimiento, se analizó el tiempo de respuesta de las solicitudes y se comprobó que la aplicación cumple con los requisitos de rendimiento establecidos.

Además, se efectuó una refactorización de las entidades Claim y TrackingLog, añadiendo índices a las tablas de la base de datos para tratar de mejorar el rendimiento de la aplicación. Posteriormente, se realizó un contraste de hipótesis que confirmó que los cambios implementados no fueron significativos y, por tanto, no mejoraron el rendimiento de la aplicación. Por tanto, se aprende también que es esencial realizar otras tareas como la refactorización del código o el cambio de hardware para tratar de mejorar el rendimiento de las aplicaciones.

En resumen, he adquirido valiosas lecciones sobre el testing de aplicaciones web utilizando el Acme-Framework, que ha facilitado enormemente este proceso. También he aprendido a utilizar Excel para un análisis más profundo de los datos obtenidos en las pruebas, algo que no había hecho anteriormente.

## 8. Bibliografía

- 08 Annexes