

# TESTING REPORT

**Acme-ANS-D04**

Celia Suárez Coronel, [celsuacor@alum.us.es](mailto:celsuacor@alum.us.es)

**C1.001**

<https://github.com/Albertoescobarsanchez/Acme-ANS-D04>

**Sevilla Mayo 26, 2025**

# Contenido

Resumen ejecutivo .....	3
Tabla de versionado .....	4
Introducción .....	5
Pruebas funcionales .....	6
Claim .....	6
List .....	6
Create .....	6
Update .....	7
Delete .....	7
Publish .....	7
Show .....	8
TrakingLog .....	9
List .....	9
Create .....	9
Update .....	10
Delete .....	10
Publish .....	10
Cobertura .....	12
Pruebas de rendimiento .....	13
Gráficas de pruebas .....	14
Conclusión .....	17
Bibliografía .....	18

## Resumen ejecutivo

Este informe presenta de manera detallada los resultados obtenidos tras la ejecución de pruebas funcionales, de validación lógica y de rendimiento sobre las funcionalidades correspondientes al rol “**AssistanceAgent**”, implementadas por el **Student #4** en el contexto del proyecto **Acme-ANS-D04**.

Las pruebas se han diseñado siguiendo una **metodología formal de testing**, con el objetivo de garantizar que el sistema responda correctamente ante escenarios tanto esperados como anómalos, incluyendo casos límite, entradas erróneas y accesos no autorizados. Estas pruebas se han realizado siguiendo el Excel dado por la asignatura llamado “Sample-Data” que nos permite testear valores válidos e incorrectos y sus límites, tanto superiores como inferiores, de cada campo.

En el plano funcional, se han verificado operaciones como la **creación, edición, publicación y eliminación de Claims** y de **registros de seguimiento (TrackingLogs)**, asegurando que las restricciones se cumplen rigurosamente.

Se han tomado en cuenta aspectos críticos como la **validación de campos obligatorios y restringidos**, el cumplimiento de la lógica de negocio (por ejemplo, la progresividad del porcentaje de resolución), y el comportamiento condicional basado en el estado de la entidad asociada (por ejemplo, la relación entre el modo borrador de una Claim y la posibilidad de publicar sus TrackingLogs).

En cuanto a las **pruebas de rendimiento**, se ha evaluado la eficiencia y capacidad de respuesta del sistema mediante múltiples ejecuciones. Los tiempos de respuesta obtenidos se han analizado estadísticamente, incluyendo el **cálculo de intervalos de confianza al 95%**, lo que permite estimar con precisión la variabilidad del sistema bajo carga controlada.

Este análisis exhaustivo no solo permite confirmar el correcto funcionamiento de las funcionalidades asignadas al rol “AssistanceAgent”, sino que también **contribuye a la garantía global de calidad, fiabilidad y robustez** del sistema, centrándose especialmente en las entidades **Claim** y **TrackingLog**, que desempeñan un papel esencial en la gestión de incidencias post-vuelo.

## Tabla de versionado

<b>Versión</b>	<b>Fecha</b>	<b>Descripción</b>
1.0	24/05/2025	Versión inicial
2.0	26/05/2025	Versión final

## Introducción

Este documento presenta las **pruebas funcionales, de cobertura y de rendimiento** realizadas sobre las funcionalidades implementadas por el **Student #4**, centradas en el rol **“AssistanceAgent”** y sus entidades **Claim** y **TrackingLog**.

Las pruebas funcionales verifican el correcto comportamiento del sistema ante situaciones válidas, errores y accesos no autorizados, así como el cumplimiento de las reglas de negocio específicas, como la progresividad del porcentaje de resolución o la coherencia entre el estado de las reclamaciones y sus registros de seguimiento.

Además, se incluyen **pruebas de rendimiento** ejecutadas en dos dispositivos distintos para evaluar la eficiencia del sistema. Los resultados se han analizado estadísticamente mediante el cálculo de **intervalos de confianza del 95%**, permitiendo valorar la estabilidad y capacidad de respuesta de la aplicación en distintos entornos.

## Pruebas funcionales

### Claim

#### List

##### Test Case: list.safe

**Descripción:** Este caso de prueba verificaba que un AssistanceAgent autorizado pudiera visualizar correctamente la lista de reclamaciones (Claims) que le pertenecen, tanto la lista de PendingClaims como la de CompletedClaims. Se comprobó que solo se mostraran las reclamaciones asociadas al agente autenticado, excluyendo cualquier reclamación ajena.

**Bugs detectados:** Ninguno

##### Test Case: list.hack

**Descripción:** Este caso de prueba verifica que ningún usuario sin el rol AssistanceAgent, o sin ser el propietario de las reclamaciones, pueda acceder a la lista de Claims. Se intentó acceder manualmente a la URL del listado por parte de otros roles y agentes no autorizados.

**Bugs detectados:** Ninguno

### Create

##### Test Case: create.safe

**Descripción:** En este caso de prueba se verifica que un AssistanceAgent pueda crear correctamente una nueva reclamación (Claim) con datos válidos. Se comprobó que los campos obligatorios (correo, descripción, tipo de reclamación, vuelo asociado) fueran requeridos, y que el sistema aceptara solo entradas válidas. Para ello, se han hecho varias validaciones con los datos de prueba del Excel "Sample-Data", para comprobar así todas las casuísticas de cada uno de los campos, probando sus límites y haciendo que fallen al insertar datos no permitidos.

**Bugs detectados:** Ninguno

##### Test Case: create.hack

**Descripción:** Se simuló el intento de crear una reclamación por parte de un usuario no autorizado (sin el rol de AssistanceAgent).

**Bugs detectados:** Ninguno

## Update

### Test Case: update.safe

**Descripción:** En este caso de prueba se valida que un AssistanceAgent pueda actualizar sus propias reclamaciones en modo borrador (draftMode = true) y que en caso contrario no se permita. Se verificó que los campos editables respondieran correctamente y que no se permitieran valores nulos ni inválidos.

**Bugs detectados:** Ninguno

### Test Case: update.hack

**Descripción:** Se intentó modificar una reclamación una reclamación de otro agente, y se forzaron valores no válidos (por ejemplo, modificar el indicador o el vuelo con valores alterados). También se probó que al no estar logueado o estarlo con otros roles no permitidos no pudieran acceder al formulario de actualización.

**Bugs detectados:** Ninguno

## Delete

### Test Case: delete.safe

**Descripción:** En este caso de prueba se valida que un AssistanceAgent pueda eliminar correctamente sus reclamaciones en modo borrador y que cuando estén publicadas no se permita. Se comprobó que el borrado se ejecutara correctamente y que no generara errores incluso si la reclamación tenía campos con valores alterados o incompletos.

**Bugs detectados:** Ninguno

### Test Case: delete.hack

**Descripción:** Se intentó eliminar una reclamación que no pertenecía al AssistanceAgent logueado, o acceder al endpoint de borrado desde otros roles o incluso sin estar logueado. También se verificó que las validaciones de seguridad impidieran este tipo de accesos ilegítimos.

**Bugs detectados:** Ninguno

## Publish

### Test Case: publish.safe

**Descripción:** Este caso de prueba verificaba que una reclamación pudiera ser publicada únicamente si estaba en modo borrador y todos sus campos requeridos estaban correctamente cumplimentados. Se validó que tras la publicación, los campos del formulario se volvieran de solo lectura y que la reclamación no pudiera volver a modificarse.

**Bugs detectados:** Ninguno

**Test Case: publish.hack**

**Descripción:** Se intentó publicar Claims desde otros roles no autorizados, publicar Claims pertenecientes a otros agentes y sin estar logueados. También se comprobó que no fuera posible alterar el modo de publicación a través de parámetros manipulados.

**Bugs detectados:** Ninguno

**Show****Test Case: show.safe**

**Descripción:** En este caso de prueba se verifica que un AssistanceAgent pueda visualizar los detalles de una Claim propia, tanto en modo borrador como publicada. Se comprobó que los datos se mostraran correctamente sin permitir edición alguna si la reclamación ya estaba publicada.

**Bugs detectados:** Ninguno

**Test Case: show.hack**

**Descripción:** Se intentó visualizar los detalles de una Claim desde otros roles no autorizados o desde otro AssistanceAgent distinto al propietario. También, se intentó visualizar una Claim sin estar logueado. Además, se verificó que no se pudiera forzar la visualización directa mediante modificación de parámetros.

**Bugs detectados:** Ninguno



## TrackingLog

### List

#### Test Case: list.safe

**Descripción:** En este caso de prueba se verifica que un AssistanceAgent autenticado pueda visualizar correctamente la lista de TrackingLogs asociados a una Claim de su propiedad. Se comprobó que la lista mostrara solo los registros pertenecientes a la reclamación seleccionada, en el orden correcto según el momento de creación.

**Bugs detectados:** Ninguno

#### Test Case: list.hack

**Descripción:** Se intentó acceder a la lista de TrackingLogs desde otros roles no autorizados o desde un AssistanceAgent que no fuera propietario de la Claim. Incluso sin estar logueado se intentó acceder a un listado de TrackingLogs, obteniendo así un error *Not Authorise*.

**Bugs detectados:** Ninguno

### Create

#### Test Case: create.safe

**Descripción:** En este caso de prueba se valida que un AssistanceAgent pueda crear un nuevo TrackingLog asociado a una Claim. Para comenzar, se comprobó que los atributos obligatorios (step, resolutionPercentage, indicator) fueran requeridos y que se introdujesen datos válidos para cada campo.

Se comprobó que el atributo resolution fuera obligatorio si resolutionPercentage =100, y opcional en caso contrario. También se verificó que el atributo indicator fuera ACCEPTED o REJECTED si resolutionPercentage era igual a 100, y PENDING si era menor. Además, se validó que el nuevo TrackingLog tuviera un resolutionPercentage mayor o igual al último existente.

**Bugs detectados:** Ninguno

#### Test Case: create.hack

**Descripción:** Se intentó crear un TrackingLog desde roles no autorizados o desde un AssistanceAgent que no fuera dueño de la Claim. También, se intentó acceder al formulario de creación sin estar logueado.

**Bugs detectados:** Ninguno

## Update

### Test Case: update.safe

**Descripción:** En este test se verifica que un TrackingLog existente pueda ser actualizado por su propietario solo si se encuentra en modo borrador (draftMode = true). Se comprobaron las restricciones de coherencia entre indicator y resolutionPercentage, la progresividad del atributo resolutionPercentage respecto al anterior y al siguiente registro, y la validación de que resolution fuera obligatorio solo si el porcentaje era 100.

**Bugs detectados:** Ninguno

### Test Case: update.hack

**Descripción:** Se intentó actualizar un TrackingLog ajeno al agente autenticado, desde otro rol y sin estar logueado.

**Bugs detectados:** Ninguno

## Delete

### Test Case: delete.safe

**Descripción:** En este test se verifica que un AssistanceAgent pueda eliminar correctamente un TrackingLog siempre que esté en modo borrador y sea de su propiedad. Se comprobó que al eliminarlo se respetaran todas las validaciones de integridad, sin errores relacionados con campos vacíos o relaciones rotas.

**Bugs detectados:** Ninguno

### Test Case: delete.hack

**Descripción:** Se intentó eliminar un TrackingLog que no pertenecía al agente, o realizar la operación desde un rol no autorizado. En todos los casos, el sistema denegó correctamente el acceso y protegió los datos.

**Bugs detectados:** Ninguno

## Publish

### Test Case: publish.safe

**Descripción:** En este caso se validó que un TrackingLog solo pudiera publicarse si la Claim asociada también estaba publicada. Se verificaron todas las restricciones antes de la publicación: resolution obligatorio si el porcentaje era 100, indicador adecuado, progresividad del porcentaje, y validación de que no se excedieran los dos registros completados.

**Bugs detectados:** Ninguno

**Test Case: publish.hack**

**Descripción:** Se intentó publicar un TrackingLog con otro agent, desde otro rol, o con violaciones a las reglas de negocio. El sistema impidió correctamente todos los intentos ilegítimos.

**Bugs detectados:** Ninguno

[Show](#)**Test Case: show.safe**

**Descripción:** Este test confirmó que un AssistanceAgent pudiera visualizar los detalles de un TrackingLog propio. Se verificó que, si el registro estaba publicado, los campos se mostraran en modo solo lectura y no permitieran modificaciones ni en el formulario ni mediante manipulación del navegador.

**Bugs detectados:** Ninguno

**Test Case: show.hack**























**Descripción:** Se intentó visualizar TrackingLogs ajenos o acceder desde otros roles. También se probó alterar manualmente la URL con IDs no válidos o no autorizados. El sistema detectó y bloqueó todos estos intentos correctamente.

**Bugs detectados:** Ninguno




















## Cobertura

En este apartado se presenta el análisis del porcentaje de cobertura alcanzado a través de la ejecución de los distintos tests previamente descritos, aplicados específicamente a las entidades “Claim” y “TrackingLog”. Este indicador de cobertura permite evaluar que el código asociado a dichas entidades ha sido verificado mediante pruebas automatizadas. De este modo, se garantiza una mayor fiabilidad y robustez del sistema, al minimizar la posibilidad de errores no detectados durante el desarrollo. La cobertura obtenida asegura que tanto las funcionalidades principales como los posibles escenarios alternativos han sido debidamente considerados en el proceso de prueba.

### 1. Claim

▼  acme.features.assistanceAgent.claim		98,7 %	1.123	15	1.138
>  AssistanceAgentClaimDeleteService.java		96,3 %	181	7	188
>  AssistanceAgentClaimUpdateService.java		96,4 %	190	7	197
>  AssistanceAgentClaimCreateService.java		99,6 %	224	1	225
>  AssistanceAgentClaimCompletedListService.java		100,0 %	74	0	74
>  AssistanceAgentClaimController.java		100,0 %	42	0	42
>  AssistanceAgentClaimPendingListService.java		100,0 %	70	0	70
>  AssistanceAgentClaimPublishService.java		100,0 %	191	0	191
>  AssistanceAgentClaimShowService.java		100,0 %	151	0	151
▼  acme.entities.claim		100,0 %	104	0	104
>  Claim.java		100,0 %	26	0	26
>  ClaimType.java		100,0 %	44	0	44
>  Indicator.java		100,0 %	34	0	34

### 1. TrackingLog

▼  acme.features.assistanceAgent.trackingLog		98,2 %	988	18	1.006
>  AssistanceAgentTrackingLogUpdateService.java		95,6 %	195	9	204
>  AssistanceAgentTrackingLogDeleteService.java		95,3 %	141	7	148
>  AssistanceAgentTrackingLogCreateService.java		99,5 %	195	1	196
>  AssistanceAgentTrackingLogListService.java		99,1 %	112	1	113
>  AssistanceAgentTrackingLogController.java		100,0 %	35	0	35
>  AssistanceAgentTrackingLogPublishService.java		100,0 %	174	0	174
>  AssistanceAgentTrackingLogShowService.java		100,0 %	136	0	136
▼  acme.entities.trackingLog		100,0 %	3	0	3
>  TrackingLog.java		100,0 %	3	0	3

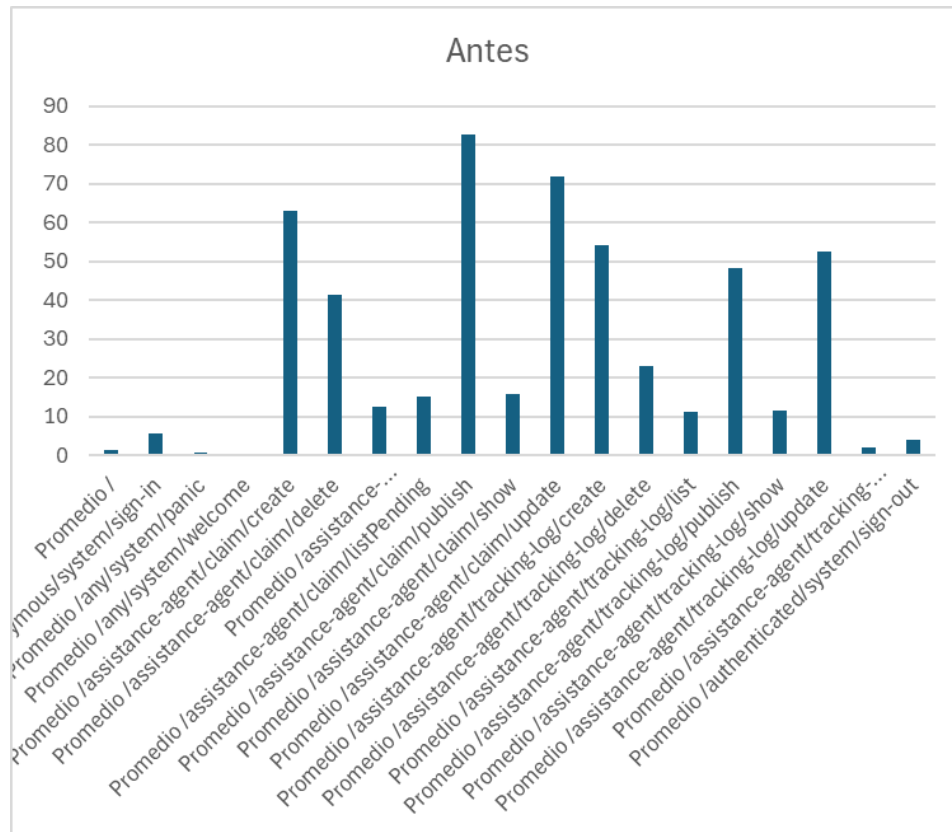
Tal y como se puede observar, se logró alcanzar satisfactoriamente el umbral mínimo de cobertura establecido como requisito previo para la ejecución de las pruebas de rendimiento (>95%). Este hito resulta fundamental, ya que asegura que las funcionalidades críticas del sistema han sido validadas de forma adecuada mediante pruebas automatizadas, proporcionando así una base sólida y confiable sobre la cual llevar a cabo los análisis de rendimiento que se describirán en los siguientes apartados.

## Pruebas de rendimiento

Con el objetivo de evaluar el comportamiento del sistema bajo condiciones controladas, se llevaron a cabo pruebas de rendimiento basadas en la información recopilada durante la ejecución de las pruebas funcionales previamente descritas. Dichas pruebas se realizaron en dos dispositivos con características de hardware diferentes: un **Lenovo IdeaPad Slim 5i Gen 11 15.6" Intel Core i5-1135G7 2.4GHz / 16GB RAM / 512GB SSD**, con especificaciones superiores, y un **HP Pavilion x360 Convertible Intel Core i5-1035G1 1.00GHz / 8GB RAM**, de gama más modesta. En ambos entornos, se midieron los tiempos de respuesta obtenidos a partir de la reproducción (replay) de los casos de prueba, y se calcularon los intervalos de confianza al 95% para dichos tiempos. A partir de esta información, se procedió a realizar un contraste de hipótesis mediante un **Z-Test para dos muestras independientes**, con el fin de determinar si las diferencias observadas en el rendimiento entre ambos dispositivos eran estadísticamente significativas. Los resultados de este análisis se detallan a continuación:

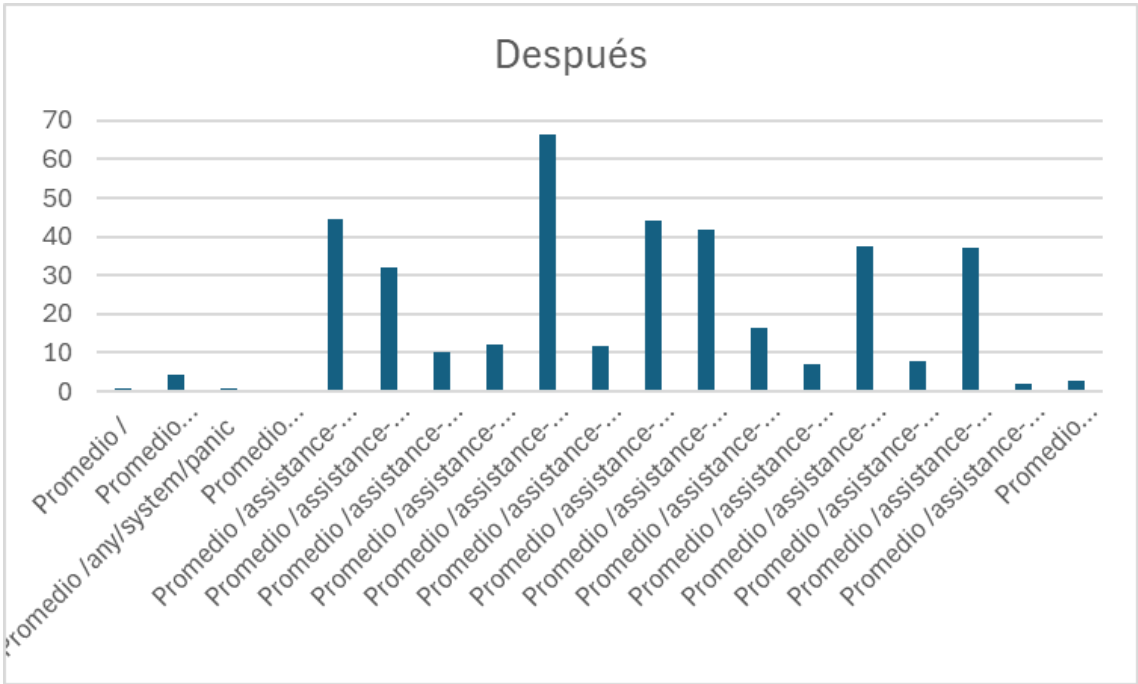
## Gráficas de pruebas

### Antes



Antes		
Media		18,064584
Error típico		0,9999904
Mediana		8,2384
Moda		0,327
Desviación estándar		30,016374
Varianza de la muestra		900,98269
Curtosis		8,4744153
Coefficiente de asimetría		2,6775855
Rango		226,1924
Mínimo		0,2112
Máximo		226,4036
Suma		16276,19
Cuenta		901
Nivel de confianza(95,0%)		1,9625845
Interval (ms)	16,10199965	20,027169
Interval (s)	0,016102	0,0200272

## Después



	<i>Después</i>	
	Media	13,221469
	Error típico	0,743939
	Mediana	6,2314
	Moda	0,3736
	Desviación estándar	22,330564
	Varianza de la muestra	498,65409
	Curtosis	10,112747
	Coefficiente de asimetría	2,8498501
	Rango	165,9524
	Mínimo	0,1725
	Máximo	166,1249
	Suma	11912,544
	Cuenta	901
	Nivel de confianza(95,0%)	1,4600571
Interval (ms)	11,76141197	14,681526
Interval (s)	0,011761412	0,0146815

## Z-Test

Prueba z para medias de dos muestras		
	80,5323	56,1491
Media	17,9951756	13,1737717
Varianza (conocida)	900,98269	498,65409
Observaciones	900	900
Diferencia hipotética de las medias	0	
z	3,86622474	
P(Z<=z) una cola	5,5267E-05	
Valor crítico de z (una cola)	1,64485363	
Valor crítico de z (dos colas)	0,00011053	
Valor crítico de z (dos colas)	1,95996398	

Como se puede apreciar, se ha obtenido un valor de 0.00011053 para un nivel de significancia  $\alpha = 0.95$ . Por lo cual, se puede deducir que los cambios realizados en la optimización del código y el uso de índices provocaron una mejora sustancial en el rendimiento, descartando la opción de que esta mejora fuera producida por azar.



## Conclusión

Durante el proceso de pruebas se ha puesto de manifiesto la importancia de definir con precisión los casos de prueba asociados a cada funcionalidad requerida por el sistema. Este enfoque ha permitido validar de forma rigurosa el comportamiento de las operaciones asignadas al rol “AssistanceAgent”, facilitando además la detección temprana de errores lógicos y el aseguramiento del cumplimiento de las restricciones de negocio en torno a las entidades Claim y TrackingLog.

Asimismo, el proceso de testing ha demostrado ser una herramienta esencial para reproducir y resolver de manera eficiente los fallos detectados, al disponer de escenarios estructurados y documentados que favorecen el diagnóstico preciso.

Por otro lado, el análisis de rendimiento ha sido fundamental para revisar los requisitos no funcionales del sistema. Ha permitido identificar posibles cuellos de botella y valorar la necesidad de realizar mejoras en el diseño o refactorización del código, con el objetivo de garantizar la estabilidad, eficiencia y escalabilidad de la aplicación en diferentes contextos de uso.

## Bibliografía

Blanco intencionalmente.