Universidad de Sevilla

Escuela Técnica Superior de Ingeniería Informática

**Testing Report**

****

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

**Curso 2024 – 2025**

Grupo: C0.52

Repositorio: <https://github.com/BVP2455/Acme-ANS-D04>

Miembros:

Francisco Javier Martos Romero

Fecha: 26-05-2025

**Índice**

[Introducción 3](#_Toc199191526)

[Functional Testing 4](#_Toc199191527)

[Performance Testing 22](#_Toc199191528)

[Conclusiones 27](#_Toc199191613)

# Introducción

En este documento se procede a realizar el Testing Report del Student 4, en este encontraremos los resultados de haber testeado los requisitos pertinentes del student, en este caso las clases Claim y TrackingLogs de los AssistanceAgent. Cada clase se divide en los distintos métodos que se pueden hacer en dichas clases: List, Create, Show, Update, Publish y Delete. Cada uno de estos métodos tiene dos tests realizados: método.safe, donde se testea todos los casos de prueba de dicho método, y método.hack, donde se prueba el hacking correspondiente a dicho método. Ambos serán explicados y se comentará que se ha hecho para testear cada uno y se analizarán los resultados de dichos tests.

Functional Testing  
Claim

En esta captura encontramos los resultados de las pruebas realizados, mostrándonos la cobertura total del código al ejecutar la aplicación. Como podemos observar es bastante buena la cobertura del código (96.1%) sobre las instrucciones ejecutadas, la menor de ellas es de un 89.9% en el publish, en su propio apartado expondremos que le falta o que instrucciones no se están cubriendo en la prueba.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

**List Pending Service**

En este método, hemos verificado las distintas listas de claims pendientes accediendo a ellas a través del desplegable que aparece en la página de navegación una vez logeado como Assistance Agent.

En la prueba de hacking del list se verificó que la URL del endpoint de no pudiera ser utilizada por un rol distinto.

Al analizar la cobertura del código, observamos que se alcanza un 96% de cobertura de instrucciones, lo que indica que no todas las líneas han sido ejecutadas durante las pruebas. Algunas líneas no presentan cobertura, como es el caso de **super**.getResponse().setAuthorised(**false**); lo que sugiere que no se ha evaluado cuando el método no era GET.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

**List Resolved Service**

En este método, hemos verificado las distintas listas de claims completadas accediendo a ellas a través del desplegable que aparece en la página de navegación una vez logeado como Assistance Agent.

En la prueba de hacking del list se verificó que la URL del endpoint de no pudiera ser utilizada por un rol distinto.

Al analizar la cobertura del código, observamos que se alcanza un 96% de cobertura de instrucciones, lo que indica que no todas las líneas han sido ejecutadas durante las pruebas. Algunas líneas no presentan cobertura, como es el caso de **super**.getResponse().setAuthorised(**false**); lo que sugiere que no se ha evaluado cuando el método no era GET.

**Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.**

**Create Service**

En este método, hemos verificado los distintos casos accediendo al botón Crear que aparece al ingresar a listado de claim. Para cada atributo, fuimos probando los ejemplos proporcionados en el archivo Excel del Scrapbook (Sample-Data.xlsx), uno por uno, dejando los demás campos en blanco y creándolo, pulsando en el botón de Create, para que fuesen saltando las validaciones oportunas. Una vez probados todos los atributos, se publicó una claim correcta.

En la prueba de hacking del create, verificamos desde la herramienta "Inspeccionar" del navegador que no se pudieran asignar valores a las legs distintos a los legId existentes en la base de datos, probando con valores como 1 y -1. También comprobamos que el campo registrationMoment no pudiera ser modificado, ya que es de solo lectura (readOnly). Además, se verificó que la URL del endpoint de creación de esa claim no pudiera ser utilizada por otro agente o por un rol distinto.

Al analizar la cobertura del código, observamos que se alcanza un 100% de cobertura de instrucciones, lo que indica que todas las líneas han sido ejecutadas durante las pruebas. Sin embargo, algunas líneas presentan cobertura parcial, como es el caso de la condición if (legId == null || legId != 0), lo que sugiere que no se han evaluado todas las posibles ramas posibles.

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

**Update Service**

En este método, hemos verificado los distintos casos accediendo al formulario que aparece al ingresar a una claim específica. Para cada atributo, fuimos probando los ejemplos proporcionados en el archivo Excel del Scrapbook (Sample-Data.xlsx), uno por uno, dejando los demás campos en blanco y actualizándolo, dándole al botón de Update, para que fuesen saltando las validaciones oportunas. Una vez probados todos los atributos, se actualizó una claim correctamente.

En la prueba de hacking del update, verificamos desde la herramienta "Inspeccionar" del navegador que no se pudieran asignar valores a las legs distintos a los legId existentes en la base de datos, probando con valores como 1 y -1. También comprobamos que el campo registrationMoment no pudiera ser modificado, ya que es de solo lectura (readOnly). Además, se verificó que la URL del endpoint de actualización de esa claim no pudiera ser utilizada por otro agente o por un rol distinto.

Al analizar la cobertura del código, observamos que se alcanza un 99.1% de cobertura de instrucciones, lo que indica que todas las líneas han sido ejecutadas durante las pruebas. Sin embargo, algunas líneas presentan cobertura parcial, como es el caso de la condición if (legId == null || legId != 0), lo que sugiere que no se han evaluado todas las posibles ramas posibles.

Texto

El contenido generado por IA puede ser incorrecto.

**Publish Service**

En este método, hemos verificado los distintos casos accediendo al formulario que aparece al ingresar a una claim específica. Para cada atributo, fuimos probando los ejemplos proporcionados en el archivo Excel del Scrapbook (Sample-Data.xlsx), uno por uno, dejando los demás campos en blanco y publicándolos, dándole al botón Publish, para que fuesen saltando las validaciones oportunas. Una vez probados todos los atributos, se publicó una claim correcta.

En la prueba de hacking del publish, verificamos desde la herramienta "Inspeccionar" del navegador que no se pudieran asignar valores a las legs distintos a los legId existentes en la base de datos, probando con valores como 1 y -1. También comprobamos que el campo registrationMoment no pudiera ser modificado, ya que es de solo lectura (readOnly). Además, se verificó que la URL del endpoint de publicación de esa claim no pudiera ser utilizada por otro agente o por un rol distinto.

Al observar la cobertura de código, detectamos que hay tres líneas dentro del método unbind a las que no se accede. Esto se debe a que, durante la prueba, no se editó ni envió una reclamación que no estuviera en modo borrador.

Texto

El contenido generado por IA puede ser incorrecto.

Imagen que contiene Escala de tiempo

El contenido generado por IA puede ser incorrecto.

**Delete Service**

En este método, hemos verificado que funcionase correctamente la funcionalidad del servicio Delete mediante el botón ubicado en la claim una vez que accedias a ella. Para ello hemos accedido a una Claim que no tenga TrackingLogs para poder realizar un borrado sin ningún inconveniente y a uno que si tuviera TrackingLogs para probar ese caso.

En la prueba de hacking del delete, se verificó que la URL del endpoint de borrado de esa claim no pudiera ser utilizada por otro agente o por un rol distinto.

Al analizar la cobertura del código, observamos que se alcanza un 98.4% de cobertura de instrucciones, lo que indica que todas las líneas han sido ejecutadas durante las pruebas. Sin embargo, algunas líneas presentan cobertura parcial, como es el caso de la condición AssistanceAgent assistanceAgent = claim == **null** ? **null** : claim.getAssistanceAgent(); lo que sugiere que no se han evaluado todas las posibles ramas posibles.

Texto

El contenido generado por IA puede ser incorrecto.

**Show Service**

En este método, hemos verificado los distintos casos accediendo al formulario que aparece al ingresar a una claim específica.

En la prueba de hacking del show se verificó que la URL del endpoint de show de esa claim no pudiera ser utilizada por otro agente o por un rol distinto.

Al analizar la cobertura del código, observamos que se alcanza un 94.3% de cobertura de instrucciones, lo que indica que no todas las líneas han sido ejecutadas durante las pruebas. Algunas líneas no presentan cobertura, como es el caso de **super**.getResponse().setAuthorised(**false**); lo que sugiere que no se ha evaluado cuando el claimId == null.

**Escala de tiempo

El contenido generado por IA puede ser incorrecto.**

Tracking Log

En esta captura encontramos los resultados de las pruebas realizados, mostrándonos la cobertura total del código al ejecutar la aplicación. Como podemos observar es bastante buena la cobertura del código (93.8%) sobre las instrucciones ejecutadas, la menor de ellas es de un 86.6% en el create, en su propio apartado expondremos que le falta o que instrucciones no se están cubriendo en la prueba.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

**List Service**

En este método, hemos verificado las distintas listas de trackingLogs accediendo a ellas a través de cualquier claim específica, pulsando el botón de Show TrackingLogs.

En la prueba de hacking del list se verificó que la URL del endpoint de listado de los trackingLogs de esa claim no pudiera ser utilizada por otro agente o por un rol distinto.

Al analizar la cobertura del código, observamos que se alcanza un 91.7% de cobertura de instrucciones, lo que indica que no todas las líneas han sido ejecutadas durante las pruebas. Algunas líneas no presentan cobertura, como es el caso de **super**.getResponse().setAuthorised(**false**); lo que sugiere que no se ha evaluado cuando el método no era GET. Además, existe otra línea sin cubrir trackingLogs = **this**.repository.findAllTrackingLogs(assistanceAgentId);, a esta no se accede debido a que las pruebas evaluadas han sido sobre Claims que tenían TrackingLogs asociadas.

Escala de tiempo

El contenido generado por IA puede ser incorrecto.

**Create Service**

En este método, hemos verificado los distintos casos accediendo al botón Crear que aparece al entrar al listado de TrackingLogs. Para cada atributo, fuimos probando los ejemplos proporcionados en el archivo Excel del Scrapbook (Sample-Data.xlsx), uno por uno, dejando los demás campos en blanco y creándolo, pulsando sobre el botón Create, para que fuesen saltando las validaciones oportunas. Una vez probados todos los atributos, se creó un TrackingLog correcto.

En la prueba de hacking del create, se verificó que la URL del endpoint de creación de ese trackingLog no pudiera ser utilizada por otro agente o por un rol distinto.

Al analizar la cobertura del código, observamos que se alcanza un 86.6% de cobertura de instrucciones, lo que indica que no todas las líneas han sido ejecutadas durante las pruebas. Algunas líneas no presentan cobertura, como es el caso de las líneas 90 , 91 y 92, debido que al evaluar las pruebas no se cumplió el caso en el que ambos logs (el actual y el más reciente asociado al claim) tienen un 100% de resolución.

Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Escala de tiempo

El contenido generado por IA puede ser incorrecto.

**Update Service**

En este método, hemos verificado los distintos casos accediendo al formulario que aparece al ingresar a un trackingLog específico. Para cada atributo, fuimos probando los ejemplos proporcionados en el archivo Excel del Scrapbook (Sample-Data.xlsx), uno por uno, dejando los demás campos en blanco y actualizándolo, dándole al botón de Update, para que fuesen saltando las validaciones oportunas. Una vez probados todos los atributos, se actualizó un trackingLog correctamente.

En la prueba de hacking del update, se verificó que la URL del endpoint de actualización de esa claim no pudiera ser utilizada por otro agente o por un rol distinto.

Al analizar la cobertura del código, observamos que se alcanza un 96.3% de cobertura de instrucciones, lo que indica que todas las líneas han sido ejecutadas durante las pruebas. Sin embargo, algunas líneas presentan cobertura parcial, como es el caso de la condición **if** (trackingLogId != **null**), lo que sugiere que no se han evaluado todas las posibles ramas posibles.

Texto

El contenido generado por IA puede ser incorrecto.

Imagen que contiene Tabla

El contenido generado por IA puede ser incorrecto.

**Publish Service**

En este método, hemos verificado los distintos casos accediendo al formulario que aparece al ingresar a una claim específica. Para cada atributo, fuimos probando los ejemplos proporcionados en el archivo Excel del Scrapbook (Sample-Data.xlsx), uno por uno, dejando los demás campos en blanco y publicándolos, dándole al botón Publish, para que fuesen saltando las validaciones oportunas. Una vez probados todos los atributos, se publicó una claim correcta.

En la prueba de hacking del publish, verificamos desde la herramienta "Inspeccionar" del navegador que no se pudieran asignar valores a las legs distintos a los legId existentes en la base de datos, probando con valores como 1 y -1. También comprobamos que el campo registrationMoment no pudiera ser modificado, ya que es de solo lectura (readOnly). Además, se verificó que la URL del endpoint de publicación de esa claim no pudiera ser utilizada por otro agente o por un rol distinto.

Al analizar la cobertura del código, observamos que se alcanza un 97.2% de cobertura de instrucciones, lo que indica que no todas las líneas han sido ejecutadas durante las pruebas. Algunas líneas presentan cobertura parcial o sin cobertua, como es el caso de **super**.getResponse().setAuthorised(**false**); lo que sugiere que no se ha evaluado un método distinto al POST.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

**Delete Service**

En este método, hemos verificado que funcionase correctamente la funcionalidad del servicio Delete mediante el botón ubicado en el trackingLog una vez que accedias a él. Para ello hemos accedido a un TrackingLog para poder realizar un borrado.

En la prueba de hacking del delete, se verificó que la URL del endpoint de borrado de ese trackingLog no pudiera ser utilizada por otro agente o por un rol distinto.

Al analizar la cobertura del código, observamos que se alcanza un 96.4% de cobertura de instrucciones, lo que indica que todas las líneas han sido ejecutadas durante las pruebas Sin embargo, algunas líneas presentan cobertura parcial, como es el caso de la condición **if** (trackingLogId != **null**), lo que sugiere que no se han evaluado todas las posibles ramas posibles.

Texto

El contenido generado por IA puede ser incorrecto.

**Show Service**

En este método, hemos verificado los distintos casos accediendo al formulario que aparece al ingresar a una trackingLog específico.

En la prueba de hacking del show se verificó que la URL del endpoint de show de ese trackingLog no pudiera ser utilizada por otro agente o por un rol distinto.

Al analizar la cobertura del código, observamos que se alcanza un 95.4% de cobertura de instrucciones, lo que indica que no todas las líneas han sido ejecutadas durante las pruebas. Algunas líneas no presentan cobertura, como es el caso de **super**.getResponse().setAuthorised(**false**); lo que sugiere que no se ha evaluado cuando el método era distinto a GET.

Texto

El contenido generado por IA puede ser incorrecto.

# Performance Testing

En este apartado se analizan los tiempos de respuesta del sistema al ejecutar las pruebas funcionale. El objetivo es evaluar el rendimiento de la aplicación y comparar la eficiencia de ambos entornos de ejecución. Para ello, se han recogido los tiempos de servicio de cada petición y se han representado en gráficas comparativas.

Además, se ha calculado un intervalo de confianza del 95% para los tiempos medios de respuesta, lo que permite estimar con alta fiabilidad el comportamiento esperado del sistema en condiciones similares. Finalmente, se ha realizado un contraste de hipótesis con un 95% de confianza para determinar si las diferencias observadas en el rendimiento entre ambos ordenadores son estadísticamente significativas y, en consecuencia, identificar cuál de los dos presenta un mejor desempeño.

Performance Testing Before Indices

En esta tabla y gráfico podemos observar los tiempos promedio de las distintas peticiones previamente a aplicarle los índices.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | request-path | time | | Promedio / | 3,32489274 | | Promedio /administrator/airport/create | 23,9832984 | | Promedio /administrator/airport/list | 8,276525 | | Promedio /administrator/airport/show | 5,0901 | | Promedio /administrator/airport/update | 18,2369254 | | Promedio /anonymous/system/sign-in | 6,27394894 | | Promedio /any/system/welcome | 2,04605951 | | Promedio /assistance-agent/claim/create | 43,3167946 | | Promedio /assistance-agent/claim/delete | 35,84832 | | Promedio /assistance-agent/claim/list | 77,55806 | | Promedio /assistance-agent/claim/pending | 266,171131 | | Promedio /assistance-agent/claim/publish | 37,826 | | Promedio /assistance-agent/claim/show | 26,9254386 | | Promedio /assistance-agent/claim/update | 39,1315912 | | Promedio /assistance-agent/tracking-log/create | 32,7562 | | Promedio /assistance-agent/tracking-log/delete | 24,66685 | | Promedio /assistance-agent/tracking-log/list | 13,55701 | | Promedio /assistance-agent/tracking-log/publish | 16,4539872 | | Promedio /assistance-agent/tracking-log/show | 10,4551765 | | Promedio /assistance-agent/tracking-log/update | 15,2080619 | | Promedio /authenticated/system/sign-out | 3,51565641 | | Promedio general | 30,2099902 | |  |

El intervalo de confianza del 95%:

|  |  |  |
| --- | --- | --- |
| *Before Indices* | |  |
| Media | 33,90042773 |  |
| Error típico | 2,224049601 |  |
| Mediana | 14,49335 |  |
| Moda | 2,1015 |  |
| Desviación estándar | 67,60525485 |  |
| Varianza de la muestra | 4570,470484 |  |
| Curtosis | 10,48241033 |  |
| Coeficiente de asimetría | 3,37142131 |  |
| Rango | 419,4851 |  |
| Mínimo | 1,2753 |  |
| Máximo | 420,7604 |  |
| Suma | 31323,99523 |  |
| Cuenta | 924 |  |
| Nivel de confianza(95,0%) | 4,364780689 |  |
| Interval (ms) | 29,53564705 | 38,2652084 |
| Interval (s) | 0,029535647 | 0,03826521 |

Performance Testing After Indices

En esta tabla y gráfico podemos observar los tiempos promedio de las distintas peticiones después de aplicarle los índices.

|  |  |
| --- | --- |
| request-path | time |
| Promedio / | 2,94982097 |
| Promedio /administrator/airport/create | 20,3440081 |
| Promedio /administrator/airport/list | 9,368325 |
| Promedio /administrator/airport/show | 4,8561 |
| Promedio /administrator/airport/update | 16,0097507 |
| Promedio /anonymous/system/sign-in | 5,98436667 |
| Promedio /any/system/welcome | 1,89179755 |
| Promedio /assistance-agent/claim/create | 44,1426649 |
| Promedio /assistance-agent/claim/delete | 38,33592 |
| Promedio /assistance-agent/claim/list | 85,45416 |
| Promedio /assistance-agent/claim/pending | 265,637274 |
| Promedio /assistance-agent/claim/publish | 37,3079828 |
| Promedio /assistance-agent/claim/show | 24,3946544 |
| Promedio /assistance-agent/claim/update | 33,3095794 |
| Promedio /assistance-agent/tracking-log/create | 29,1440481 |
| Promedio /assistance-agent/tracking-log/delete | 28,68005 |
| Promedio /assistance-agent/tracking-log/list | 12,8294867 |
| Promedio /assistance-agent/tracking-log/publish | 17,1939213 |
| Promedio /assistance-agent/tracking-log/show | 10,0181353 |
| Promedio /assistance-agent/tracking-log/update | 16,13545 |
| Promedio /authenticated/system/sign-out | 3,0648641 |
| Promedio general | 29,2897486 |

El intervalo de confianza del 95%:

|  |  |  |
| --- | --- | --- |
| *After Indices* | |  |
| Media | 32,92407615 |  |
| Error típico | 2,222325055 |  |
| Mediana | 14,53035 |  |
| Moda | 2,2002 |  |
| Desviación estándar | 67,55283319 |  |
| Varianza de la muestra | 4563,385272 |  |
| Curtosis | 11,02727835 |  |
| Coeficiente de asimetría | 3,421488839 |  |
| Rango | 495,3963 |  |
| Mínimo | 1,242 |  |
| Máximo | 496,6383 |  |
| Suma | 30421,84636 |  |
| Cuenta | 924 |  |
| Nivel de confianza(95,0%) | 4,361396203 |  |
| Interval (ms) | 28,56267994 | 37,2854723 |
| Interval (s) | 0,02856268 | 0,03728547 |

Contraste de hipótesis para el 95%

Al llevar a cabo el contraste de hipótesis mediante la prueba Z, utilizando los valores obtenidos en ambos entornos (antes y después de aplicar los datos), se han obtenido los siguientes resultados:

|  |  |  |
| --- | --- | --- |
| Prueba z para medias de dos muestras | |  |
|  | *Before* | *After* |
| Media | 30,2826424 | 29,3774203 |
| Varianza (conocida) | 4570,47048 | 4563,38527 |
| Observaciones | 1049 | 1049 |
| Diferencia hipotética de las medias | 0 |  |
| z | 0,30677208 |  |
| P(Z<=z) una cola | 0,37950843 |  |
| Valor crítico de z (una cola) | 1,64485363 |  |
| Valor crítico de z (dos colas) | 0,75901686 |  |
| Valor crítico de z (dos colas) | 1,95996398 |  |

Dado un nivel de significancia α = 0.05, el valor obtenido para la probabilidad bilateral P(Z ≤ z) ha sido de 0,7590. Este valor es considerablemente superior al umbral de significancia, por lo que no se puede rechazar la hipótesis nula. En consecuencia, se concluye que la creación y aplicación de los índices en el código no produce una mejora estadísticamente significativa en el rendimiento del sistema.

Este resultado puede explicarse por el hecho de que el volumen de datos en la base de datos es relativamente reducido, lo que hace que el uso de índices no tenga un impacto notable en los tiempos de respuesta.

# Conclusiones

Se han medido los tiempos de respuesta del sistema antes y después de la integración de índices en la base de datos, con el objetivo de evaluar su impacto en el rendimiento. Tras analizar los resultados y aplicar una prueba estadística con un nivel de confianza del 95%, se ha determinado que no existe una mejora significativa en el rendimiento tras la introducción de dichos índices. Esto se debe principalmente a que el volumen de datos manejado por el sistema es reducido, por lo que los índices no ofrecen una ventaja apreciable en los tiempos de acceso o ejecución de las operaciones.