***TESTING REPORT C3***

INFORMACIÓN DE GRUPO

|  | |
| --- | --- |
|  | |
| | **Group:** **C3.056** | | | --- | --- | | **Repository:** https://github.com/Pablobi/Acme-ANS-C3 | | |  |  | | **Student #4**  **UVUS:**  pabberima  **Contact:** [pabberima@alum.us.es](mailto:pabberima@alum.us.es) |  | |  |  | | **Date:**  Sevilla, Agosto 14,10,2025 | | |  |
|  |  |
|  |  |
|  | |

TABLA DE CONTENIDO

Portada………………………………………………………………………..1

Resumen Ejecutivo……………………………………………….2

Tabla de Revisiones………………………………………….2

Introducción…………………………………………………………….2

Contenidos………………………………………………………………….3

Bibliografía…………………………………………………………..9

**RESUMEN EJECUTIVO**

Este informe representa de manera detallada el proceso de testing llevado a cabo para las features asociadas al Assistance Agent, describiendo las funcionalidades testeadas, los casos de prueba evaluados y los resultados obtenidos, funcionales y de rendimiento.

En el ámbito del testing funcional, para cada feature se dará un porcentaje en cobertura de código, así como los resultados de los casos de prueba utilizados.

En el ámbito de testing de rendimiento, se proporcionan análisis para cada feature que permiten comparar el rendimiento para cada una de estas.

TABLA DE REVISIONES

| **Revisión** | **Fecha** | **Descripción** |
| --- | --- | --- |
| 1 | 2025-10-14 | Creación del informe |

**INTRODUCCIÓN**

El objetivo de este Informe de Pruebas es, principalmente, comprobar que el código realizado en este proyecto es útil (es decir, se ejecuta tarde o temprano en alguna petición) y está optimizado.

Para las pruebas se han empleado numerosos casos de prueba mediante intentos positivos, negativos y de hacking, cuyos datos se han analizado posteriormente con ayuda de Microsoft Excel.

Se han utilizado el Tester Recorder que aporta el framework para grabar los datos en sus formularios correspondientes y comprobar así, si devuelve el valor esperado de error o aceptación, y el Tester Replayer para repetir los casos de prueba.

**CONTENIDOS**

**Metodología**

Como se ha mencionado anteriormente, para comprobar un servicio, se ha utilizado el Tester Recorder para grabar todos los datos introducidos: Para probar datos positivos el registro de llamadas se nombra como “Entidad-Servicio.safe”, para casos negativos el registro es llamado como “EntidadServicio.hack”.

Se han utilizado los siguientes criterios para los datos de entrada según su tipo:

* **String:** en casos positivos, hay valores en su rango mínimo y máximo de longitud, valores por el medio, cerca del final e inicial, para los casos negativos valores como null,caracteres raros, superar el máximo de longitud y cadenas que pueden ocasionar hacking.
* **Email:** en casos positivos, hay valores en su rango mínimo y máximo de longitud, y valores por el medio, cerca del final e inicial.

En casos negativos valores como null, emails no válidos, y superar el máximo de longitud.

* **Fechas:** en casos positivos, hay valores en pasado, Se prueban también los valores máximos, mínimos e intermedio para la restricción de past.

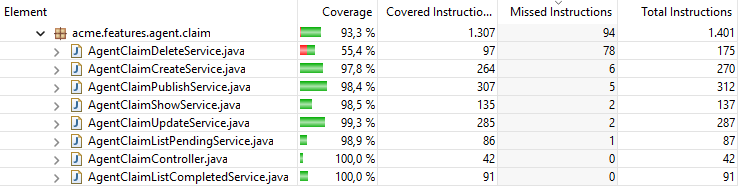
En casos negativos, hay fechas que superan la fecha actual, y fechas no escritas correctamente.

* **Score:** en casos positivos, hay valores entre 0.00, y 100.00 y valores integer

En casos negativos,valores negativos, valores por encima de 100.00, y valores que no son correctos para un Double, así como espacios en blanco.

**Testing Funcional**

**Feature #8:**



* **1.1.Show**:

1. Descripción: Para el caso positivo se ha accedido tanto a un claim publicado mediante un agent distinto al que lo creó, como a uno sin publicar mediante el mismo agente que lo creó.

Para el caso de hacking se intentó acceder mediante otro usuario con distinto rol, hacer una petición pero con el id de la request nulo y acceder a un claim no publicado mediante otro agent no está relacionado: en los tres casos da fallo de autorización.

1. Cobertura: 98.5%;

.

* **1.2.Create**:

1. Descripción: Si se llena el formulario con datos correctos siendo agent, se crea el objeto sin problemas, Si se cambian los valores de los atributos readOnly “status” y “creationMoment” mediante hacking usando F12, el objeto se crea sin problemas, pero dichos valores se establecen automáticamente como el framework lo espera, sin aceptar los valores cambiados por el usuario

Si se cambian los valores de leg (atributo de navegación) o y “creationMoment”

Si se envía el formulario completamente vacío, da error not null en todo campo obligatorio. Si se crea un claim siendo agent con uno de los valores correctos y los demás incorrectos, se acepta el valor correcto y da error en el resto de valores incorrectos.

Si se hackea para poner un claim cuyo leg tiene una fecha de llegada superior a la de la creación del claim, un leg que no está publicado o un leg inválido, aparece un error de autorización. Lo mismo ocurre con claimType, si se hackea con un valor no válido, no se autoriza la creación

Si se crea un claim desde un rol que no es agent, da error de autorización

1. Cobertura: 97’8%

* **1.3.ListCompleted y ListPending**:

1. Descripción: Para el caso positivo se listaron los claims pending o completed pertenecientes a un agent concreto, siendo exitoso tanto en listCompleted como en listPending. En el caso de hacking se intentó listar los claims desde otro rol distinto al de agent: en ambos casos da error de autorización
2. Cobertura: 100% de cobertura en ambos casos.

* **1.4.Publish**:

1. Descripción: Si se llena el formulario con datos correctos siendo agent, se publica el objeto sin problemas, Si se cambian los valores de los atributos readOnly “status” y “creationMoment” mediante hacking usando F12, el objeto se publica sin problemas, pero dichos valores se establecen automáticamente como el framework lo espera, sin aceptar los valores cambiados por el usuario

Si se envía el formulario completamente vacío, da error not null en todo campo obligatorio. Si se publica un claim siendo agent con uno de los valores correctos y los demás incorrectos, se acepta el valor correcto y da error en el resto de valores incorrectos.

Si se hackea para poner un claim cuyo leg tiene una fecha de llegada superior a la de la creación del claim, un leg que no está publicado o un leg inválido, aparece un error de autorización. Lo mismo ocurre con claimType, si se hackea con un valor no válido, no se autoriza la publicación.

Si se intenta publicar un claim que no tiene tracking logs asociados, aparece un error de validación

Si se publica un claim desde un rol que no es agent, da error de autorización. Si se intenta publicar un claim ya publicado mediante su correspondiente agent, da error de autorización. Si otro agent intenta publicar una claim que no es suya, se produce también error de autorización.

1. Cobertura: 98.4%. En la siguiente línea falta por comprobar el caso de que trackingLog sea null.



* **1.5.Update**:

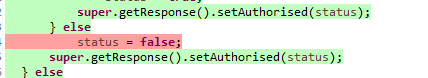
1. Descripción: Si se llena el formulario con datos correctos siendo agent, se actualiza el objeto sin problemas, Si se cambian los valores de los atributos readOnly “status” y “creationMoment” mediante hacking usando F12, el objeto se actualiza sin problemas, pero dichos valores se establecen automáticamente como el framework lo espera, sin aceptar los valores cambiados por el usuario

Si se envía el formulario completamente vacío, da error not null en todo campo obligatorio. Si se actualiza un claim siendo agent con uno de los valores correctos y los demás incorrectos, se acepta el valor correcto y da error en el resto de valores incorrectos.

Si se hackea para poner un claim cuyo leg tiene una fecha de llegada superior a la de la creación del claim, un leg que no está publicado o un leg inválido, aparece un error de autorización. Lo mismo ocurre con claimType, si se hackea con un valor no válido, no se autoriza la actualizacion

Si se actualiza un claim desde un rol que no es agent, da error de autorización. Si se actualiza un claim ya publicado mediante su correspondiente agent, da error de autorización. Si otro agente intenta actualizar una claim que no es suya, se produce también error de autorización.

El código en rojo no autorizaria la publicación en caso de que se usase update como metodo get



1. Cobertura: 99.3%.

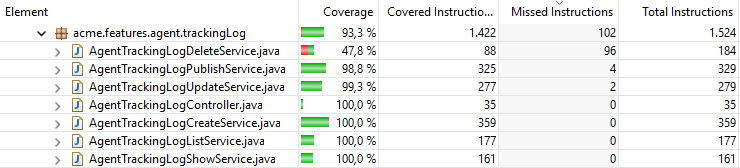
* **1.6.Delete**:

1. Descripción: En el caso positivo se borra un claim siendo su agent utilizando el botón de delete. Si el claim está publicado se intenta borrar aunque sea por parte de su agent, da un error de autorización Si se intenta borrar un claim por parte de otro agent al que no pertenecía, o desde otro rol distinto, da también un error de autorización

.

1. Cobertura: 57.7% Tiene tan poca cobertura debido a que la función unbind no es ejecutada en ningún momento.

**Feature #9:**



* **2.1.Show**:

1. Descripción: Para el caso positivo se mostró un trackingLog desde su propio agent. Se accedió tanto a un trackingLog publicada como a una sin publicar. Para el caso de hacking se intentó mostrar un trackingLog que pertenece a otro agent distinto que hizo la petición y se probó a acceder con otro usuario con distinto rol. En ambos casos da error de autorización
2. Cobertura: 100%

* **2.2.Create**:

1. Descripción: Si se llena el formulario con datos correctos siendo agent, se crea el objeto sin problemas, Si se cambian los valores de los atributos readOnly “status” y “creationMoment” mediante hacking usando F12, el objeto se crea sin problemas, pero dichos valores se establecen automáticamente como el framework lo espera, sin aceptar los valores cambiados por el usuario

Si se crea otra trackingLog de 100% sin que esté publicada la otra de 100%, da error de autorización. Una vez que la de 100% esté publicada ya se puede crear, pero si después de publicar esta última se intenta crear otra, da también fallo de autorización

Si se crea una trackingLog con 100% de resolución, y tiene su status “pending” o no tiene “resolution”, da un fallo de validación. Si tiene menos de 100% y tiene status “accepted” o “rejected”, también se provoca error en la validación

Si se envía el formulario completamente vacío, da error not null en todo campo obligatorio. Si se crea un trackingLog siendo agent con uno de los valores correctos y los demás incorrectos, se acepta el valor correcto y da error en el resto de valores incorrectos.

Si se crea un tracking log con un valor no válido para status, se muestra un error de autorización.

Si se crea un trackingLog desde un rol que no es agent, da error de autorización

1. Cobertura: 100%

* **2.3.List**:

1. Descripción: Al igual que en listar claims, se listaron los trackingLogs entrando desde un claim. En los casos negativos se intentó entrar a una trackingLog de un claim perteneciente a otro agent y desde otro usuario con otro rol.
2. Cobertura: 100%

* **2.4.Publish**:

1. Descripción: Si se llena el formulario con datos correctos siendo agent, se publica el objeto sin problemas, Si se cambian los valores de los atributos readOnly “status” y “percentage”(si es el trackingLog extra) “updateMoment” mediante hacking usando F12, el objeto se publica sin problemas, pero dichos valores se establecen automáticamente como el framework lo espera, sin aceptar los valores cambiados por el usuario

Si se intenta publicar un trackingLog sin que su claim esté publicado, da un fallo de autorización.

Si se envía el formulario completamente vacío, da error not null en todo campo obligatorio. Si se publica un trackingLog siendo agent con uno de los valores correctos y los demás incorrectos, se acepta el valor correcto y da error en el resto de valores incorrectos.

Si se intenta publicar un tracking cuyo porcentaje de resolución es menor a otro tracking log que esté publicado, da un error de validación. Si se intenta publicar un claim al 100% con un status que no sea accepted o rejected, y sin resolución, se produce también error de validación

Si se publica un tracking log con un valor no válido para status, se muestra un error de autorización.

Si se publica un trackingLog desde un rol que no es agent, da error de autorización. Si se intenta publicar un trackingLog ya publicado mediante su correspondiente agent, da error de autorización. Si otro agent intenta publicar un trackingLog que no es suyo, se produce también error de autorización.

1. Cobertura:98.8%

* **2.5.Update**:

1. Descripción: Descripción: Si se llena el formulario con datos correctos siendo agent, se actualiza el objeto sin problemas, Si se cambian los valores de los atributos readOnly “status” y “percentage”(si es el trackingLog extra con 100% de resolución) y “updateMoment” mediante hacking usando F12, el objeto se actualiza sin problemas, pero dichos valores se establecen automáticamente como el framework lo espera, sin aceptar los valores cambiados por el usuario

Si se envía el formulario completamente vacío, da error not null en todo campo obligatorio. Si se actualiza un claim siendo agent con uno de los valores correctos y los demás incorrectos, se acepta el valor correcto y da error en el resto de valores incorrectos.

Si se intenta actualizae un claim cuyo leg tiene una fecha de llegada superior a la de la creación del claim, o bien el vuelo asociado a dicho leg del claim no está publicado, dan errores en el campo “leg”.

Si se actualiza un tracking log con un valor no válido para status, se muestra un error de autorización.

Si se actualiza un claim desde un rol que no es agent, da error de autorización. Si se actualiza un claim ya publicado mediante su correspondiente agent, da error de autorización. Si otro agente intenta actualizar una claim que no es suya, se produce también error de autorización.

1. Cobertura: 99.3%

* **2.6.Delete**:

1. Descripción: En el caso positivo se borró un trackingLog mediante el botón en su formulario. En el caso de hacking se intentó borrar un trackingLog ya publicado, se intentó borrar un tracking Log de otro agent distinto y se intentó borrar un trackingLog con otro roll. En los tres casos da fallo de autorización
2. Cobertura: 47.1%

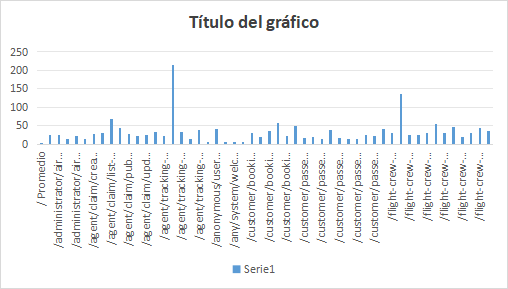
**Testing de Rendimiento**

En este segundo apartado veremos métricas respecto al rendimiento de la aplicación, siempre relacionado a las features asociadas a Agent, antes y después de añadir índices a las tablas de la base de datos. Se proporciona un intervalo de confianza del 95% para el tiempo que tarda el proyecto en atender las solicitudes para dos versiones: primero sin índices y luego con ello. Además, un contraste de hipótesis con un 95% de confianza determina cuál sería el caso más eficiente.

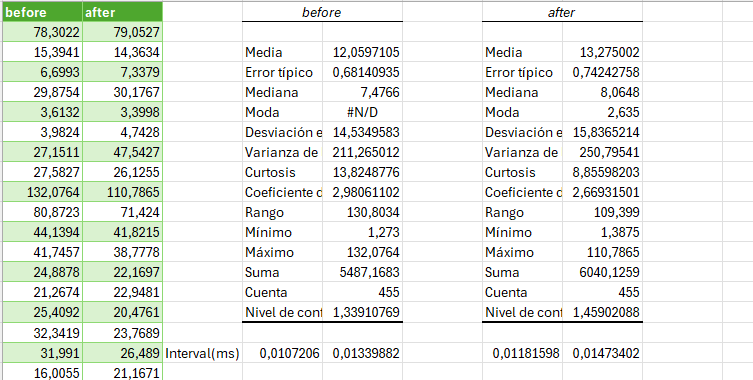
Estos índices se añaden en base a las consultas que se hacen desde los repositorios y los atributos que se utilizan de cada entidad.

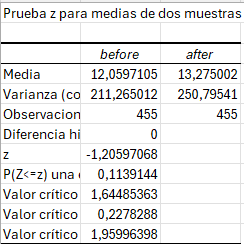
Los siguientes tests han sido realizados con un sistema que consta con 8GB de memoria RAM y un Intel I7-10510U como procesador.

Primero, se ejecutan todos los tests con una configuración normal( es decir sin utilizar índices) de la versión sin índices y luego la versión con índices. La tabla representa los distintos tiempos de ejecución para las distintas llamadas realizadas en los tests agrupadas por ruta. Tras utilizar los índices explicados en las diapositivas de clase para mejorar el rendimiento de las llamadas, los resultados quedarían de la siguiente forma:

****

Dado el valor critico de z (dos colas) = 0,2278788 < 1, según la teoría vista en clase las pruebas, no se rechaza la hipótesis nula. No hay evidencia suficiente para poder decir que haya una diferencia estadísticamente significativa entre usar los índices y no usarlos.

****

****

**BIBLIOGRAFÍA**

Intencionadamente en blanco.