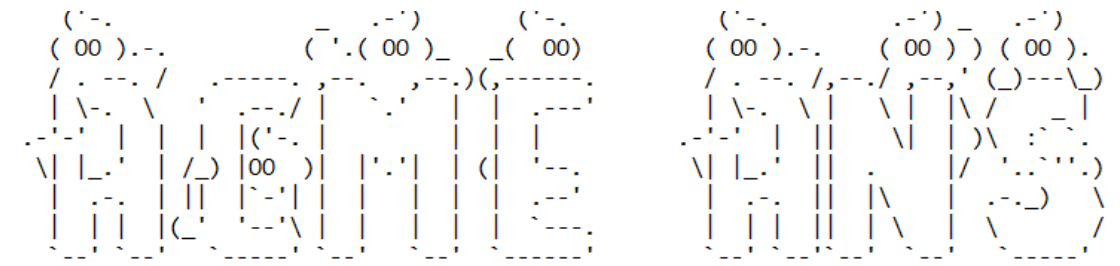
REPORTE DE TESTING Y MUTACIONES DEL CÓDIGO

Acme-ANS-D04



Repositorio: https://github.com/FranciscoFernandezN/Acme-ANS

Creado por el grupo C1.022, del G1

Participantes	
Nombres	Correos
Benito Merchán, Manuel Jesús	manbenmer1@alum.us.es

25 de Mayo de 2025

Índice

Resumen ejecutivo

Este es el proyecto del grupo C1.022 sobre Acme AirNav Solutions, S.A. (Acme ANS, S.A. abreviado), la cual es una compañía ficticia especializada en ayudar a aeropuertos a organizar y coordinar sus operaciones a partir de soluciones desarrolladas en software. La logística de los vuelos (la programación de los vuelos, la organización de reservas y de tripulación, etc.) se gestionan mediante el desarrollo de un WIS.

Con esto, para el correcto funcionamiento de la aplicación, se deberá escribir un reporte de testing donde se describan los valores de cobertura del código junto con los valores correspondientes al rendimiento de la aplicación, ambos, previos y posteriores a realizar mutaciones en el código.

Tabla de revisiones

Número	Fecha	Descripción
v1.0.0	25/05/2025	Versión inicial terminada del reporte

Introducción

En este reporte quedarán expuestos los resultados de los tests realizados durante el entregable D04 sobre las funcionalidades desarrolladas por el estudiante 4, Manuel Jesús Benito Merchán, sobre los requisitos 8 y 9.

Se reportarán 2 tipos de resultados, los relacionados con el testeo funcional y los relacionados con el testeo del rendimiento. A su vez también se reportarán las 5 mutaciones realizadas sobre el código y sus distintos efectos.

Este reporte está organizado de la siguiente forma:

1. **Resumen ejecutivo**: Introducción breve sobre el reporte.
2. **Tabla de revisiones**: Historial de revisiones del documento.
3. **Introducción**: Contextualización de la planificación y progreso además de su importancia.
4. **Funcional testing**: Resultados obtenidos de la fase de testing y posibles bugs encontrados.
5. **Performance testing**: Resultados obtenidos de analizar el rendimiento de los test en 2 ordenadores.
6. **Mutaciones en las claims**: Resultados de introducir bugs intencionales en el código.
7. **Conclusiones**: Resumen de los hallazgos y la importancia de este reporte.
8. **Bibliografía**: Fuentes consultadas durante la investigación.

Functional testing

- Resultados sobre el testeo en las claims:

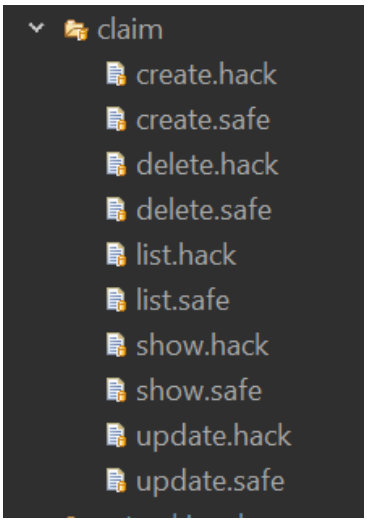
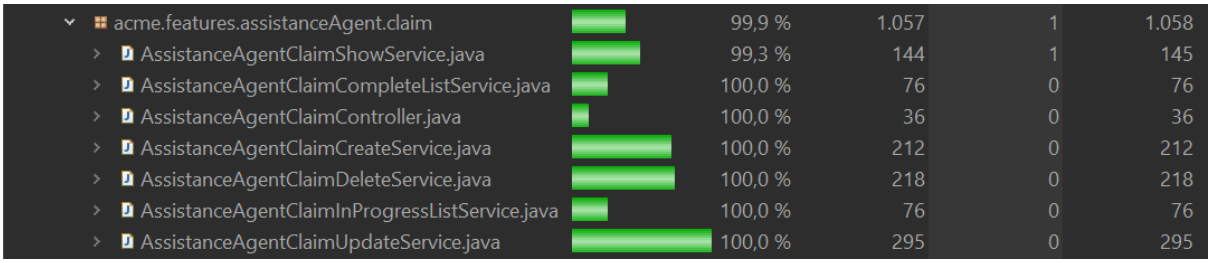


Figura 1: Archivos de testing para las claims

Se reportarion varios bugs durante la fase de testing. El más notorio fue al realizar las distintas formas de hacking, que permitian el cambio de valores en el f12 y su posterior actualización, lo que resultaba en un panic y una brecha de seguridad. También fueron detectadas validaciones innecesarias y otras necesarias.



acme.features.assistanceAgent.claim	99,9 %	1.057	1	1.058
AssistanceAgentClaimShowService.java	99,3 %	144	1	145
AssistanceAgentClaimCompleteListService.java	100,0 %	76	0	76
AssistanceAgentClaimController.java	100,0 %	36	0	36
AssistanceAgentClaimCreateService.java	100,0 %	212	0	212
AssistanceAgentClaimDeleteService.java	100,0 %	218	0	218
AssistanceAgentClaimInProgressListService.java	100,0 %	76	0	76
AssistanceAgentClaimUpdateService.java	100,0 %	295	0	295

Figura 2: Cobertura de los tests en las claims

Se ha logrado alcanzar una cobertura casi total de la feature con un 99.9%, que cubre todas las instrucciones a ejecutar pero hay alguna rama condicional que no se ejecuta debido a la posible falta de algunos datos de ejemplo más para el testeo.

```
@Override
public void authorise() {
    boolean status;
    int claimId;
    Claim claim;

    claimId = super.getRequest().getData("id", int.class);
    claim = this.aacr.findClaimById(claimId);
    status = super.getRequest().getPrincipal().hasRealmOfType(AssistanceAgent.class) && super.getRequest().getPrincipal().getRealmOfType(AssistanceAgent.class).getId() == claim.getAgent().getId() && claim != null;

    super.getResponse().setAuthorised(status);
}
```

Figura 3: Condición sin cubrir al completo

- Resultados sobre el testeo sobre los tracking logs:

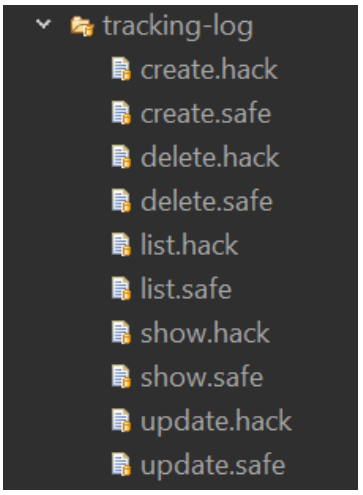


Figura 4: Archivos de testing para los tracking logs

Fueron detectados varios bugs durante la fase de testing. El más importante de ellos fue que se tenia la capacidad de publicar un log aunque su claim no estuviera pública. El resto fueron problemas de validaciones.

Se ha logrado alcanzar una cobertura casi total de la feature, no se ha alcanzado el 100% debido a ramas condicionales que no se ejecutan por la posible falta de algunos datos de ejemplo más para el testeo.

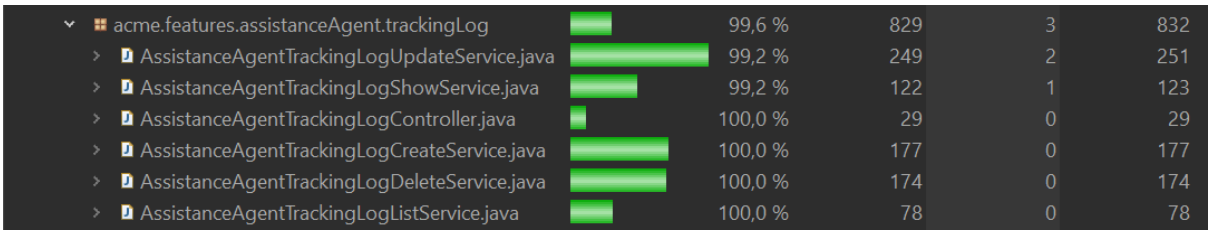


Figura 5: Cobertura de los tests en los tracking logs

Figura 6: Condición sin cubrir al completo en el show

```java
@Override
public void authorise() {
    boolean status;
    int trackingLogId;
    TrackingLog trackingLog;
    AssistanceAgent agent;

    agent = (AssistanceAgent) super.getRequest().getPrincipal().getRealmOfType(AssistanceAgent.class);

    trackingLogId = super.getRequest().getData("id", int.class);
    trackingLog = this.aatlr.findTrackingLogById(trackingLogId);

    status = super.getRequest().getPrincipal().hasRealmOfType(AssistanceAgent.class) && agent.getId() == trackingLog.getAgent().getId() && !trackingLog.getIsPublished() && trackingLog != null;
    if (status && super.getRequest().hasData("claim")) {
        List<Claim> claims = this.aatlr.findAllClaimsByAgentId(agent.getId());
        int claimId = super.getRequest().getData("claim", int.class);
        Claim claim = this.aatlr.findClaimById(claimId);
        status = claimId == 0 || claim != null && claims.contains(claim);
    }

    super.getResponse().setAuthorised(status);
}
```

Figura 7:Condiciones sin cubrir al completo en el update

Performance testing

- Resultados en ordenador 1 con índices:

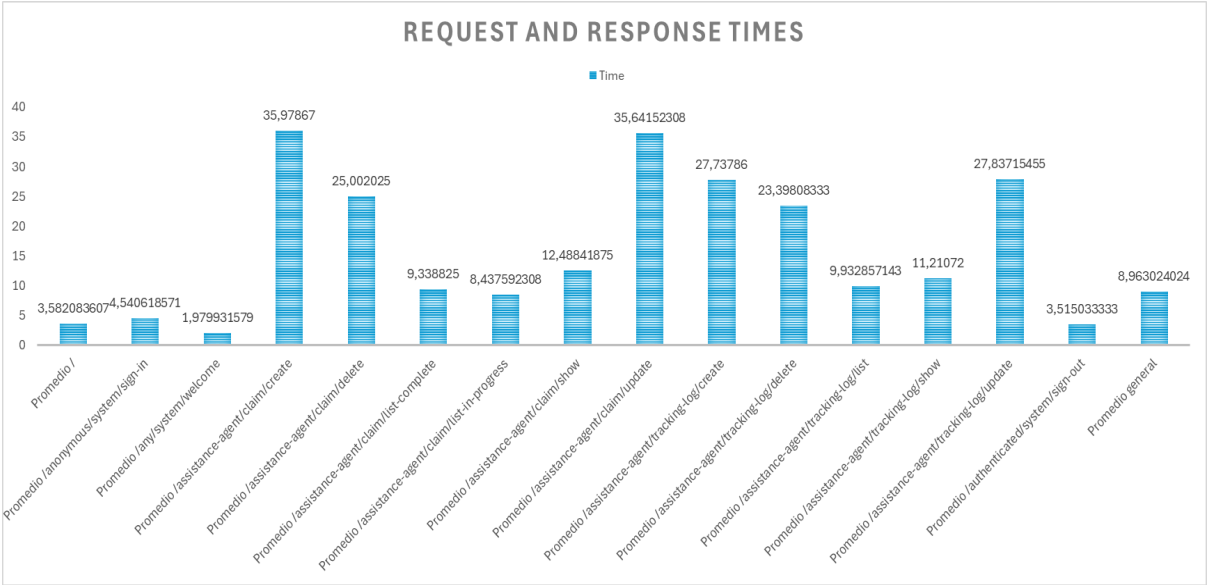


Figura 1: Tiempos para Ordenador 1 con índices

En esta tabla podemos observar los tiempos promedios de las distintas peticiones realizadas tanto en las claims como en los tracking logs con los índices calculados por el ordenador 1.

Es notable que las peticiones que más han tardado son las relacionadas con la creación, actualización y eliminación de datos, esto se debe a la gran cantidad de datos que deben de manejar en las numerosas peticiones que se realizan. El listado y el resto de peticiones no deben trabajar mucho debido a que no contamos con la suficiente cantidad de datos como para que el sistema se tenga que esforzar en gran medida.

- Resultados en ordenador 1 sin índices:

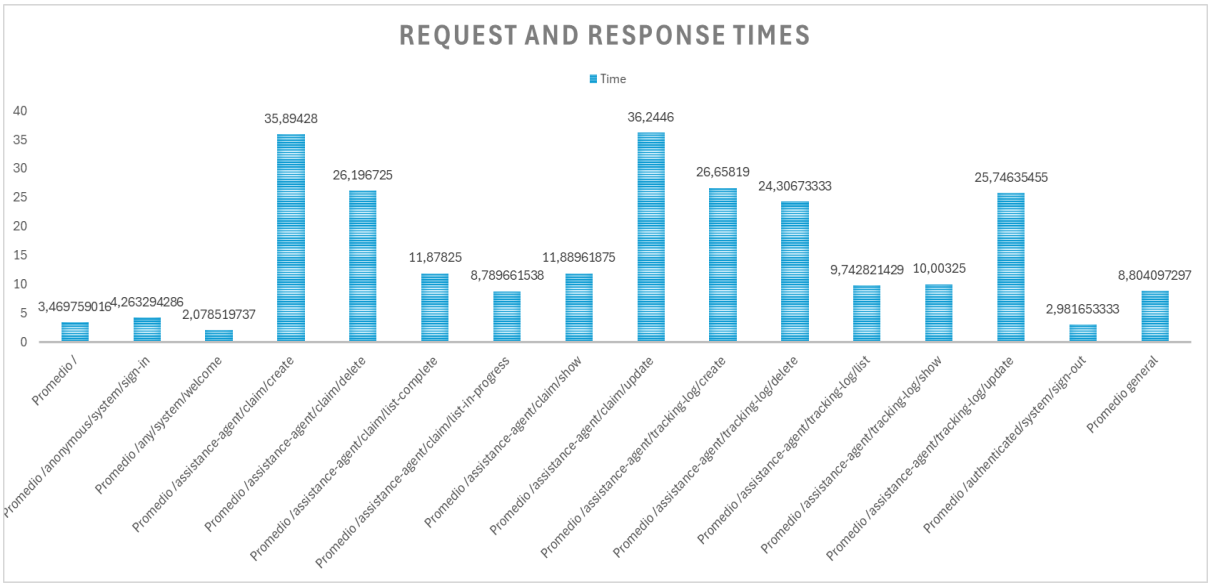


Figura 2: Tiempos para Ordenador 1 sin índices

Como se puede observar, los cambios entre usar o no índices no es muy clara, y esto es debido a que para que se refleje el rendimiento de los índices la cantidad de datos en la base de datos debe ser bastante más grande que los datos que actualmente contiene.

De hecho, en este caso, por diversos factores imposibles de controlar, el tiempo promedio en general ha sido más bajo que con el uso de índices.

Con índices	
Media	8,963024024
Error típico	0,712792152
Mediana	3,4674
Moda	#N/D
Desviación estándar	13,00723618
Varianza de la muestra	169,1881931
Curtosis	9,623130415
Coeficiente de asimetría	2,946570188
Rango	80,4072
Mínimo	1,1291
Máximo	81,5363
Suma	2984,687
Cuenta	333
Nivel de confianza(95,0%)	1,40215843

Sin índices	
Media	8,804097297
Error típico	0,7027129
Mediana	3,534
Moda	7,7455
Desviación estándar	12,82330709
Varianza de la muestra	164,4372048
Curtosis	9,853705679
Coeficiente de asimetría	2,986201001
Rango	76,4038
Mínimo	1,0413
Máximo	77,4451
Suma	2931,7644
Cuenta	333
Nivel de confianza(95,0%)	1,38233118

Prueba z para medias de dos muestras

	Con índices	*Sin índices*
Media	8,963024024	8,8040973
Varianza (conocida)	169,1881931	164,437205
Observaciones	333	333
Diferencia hipotética de la	0	
z	0,158777699	
P(Z<=z) una cola	0,43692201	
Valor crítico de z (una cola	1,644853627	
Valor crítico de z (dos cola	0,87384402	
Valor crítico de z (dos cola	1,959963985	

Figura 3: Valores estadísticos entre Ordenador 1 con y sin índices

La tabla nos aporta que a pesar de que los índices mejoran el rendimiento, si lo comparamos sin ellos parece que los tiempos mejoran en este caso.

Nota: Por razones que desconozco la moda del primer cálculo no aparece pero con la 2ª nos podemos hacer una idea de como es más o menos.

- Resultados del ordenador 2:

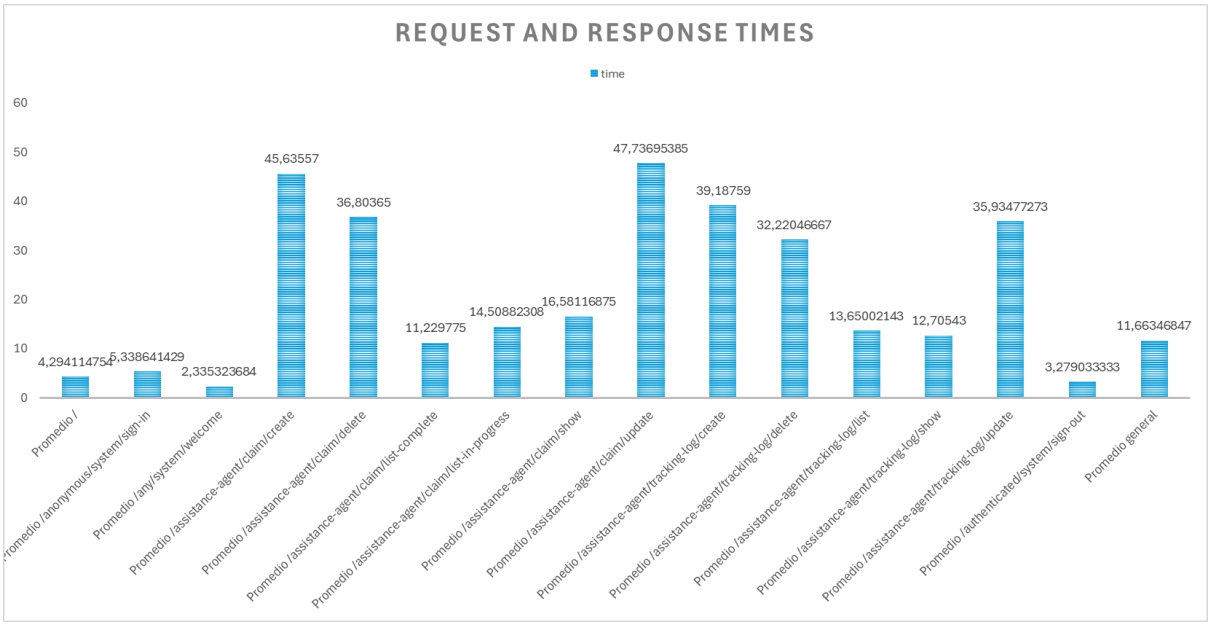


Figura 4: Tiempos para Ordenador 2 con índices

En esta tabla podemos observar los tiempos promedios de las distintas peticiones realizadas tanto en las claims como en los tracking logs con los índices calculados por el ordenador 2.

Estos resultados fueron obtenidos con un ordenador de similar potencia al ordenador 1, pero podemos ver que las proporciones se mantienen y las peticiones de tipo POST siguen resaltando por encima del resto..

Ordenador 1			*Ordenador 2*	
Media	8,963024024		Media	11,66346847
Error típico	0,712792152		Error típico	0,940068263
Mediana	3,4674		Mediana	3,7995
Desviación estándar	13,00723618		Desviación estándar	17,15463601
Varianza de la muestra	169,1881931		Varianza de la muestra	294,2815366
Curtosis	9,623130415		Curtosis	7,211499991
Coeficiente de asimetría	2,946570188		Coeficiente de asimetría	2,67969907
Rango	80,4072		Rango	91,95
Mínimo	1,1291		Mínimo	1,5242
Máximo	81,5363		Máximo	93,4742
Suma	2984,687		Suma	3883,935
Cuenta	333		Cuenta	333
Nivel de confianza(95,0%)	1,40215843		Nivel de confianza(95,0%)	1,849241235

Prueba z para medias de dos muestras

	Ordenador 1	*Ordenador 2*
Media	8,963024024	11,6634685
Varianza (conocida)	169,1881931	294,281537
Observaciones	333	333
Diferencia hipotética de las medias	0	
z	-2,289004658	
P(Z<=z) una cola	0,011039541	
Valor crítico de z (una cola)	1,644853627	
Valor crítico de z (dos colas)	0,022079082	
Valor crítico de z (dos colas)	1,959963985	

Figura 3: Valores estadísticos entre Ordenador 1 y Ordenador 2 con índices

Al haberse realizado la comparación con un ordenador ligeramente menos potente podemos ver que los tiempos han empeorado, pero esto es algo obvio debido a la diferencia de hardware, ya que el número de instruciones ejecutadas es el mismo.

Mutaciones en los aeropuertos

- Mutación 1, eliminar el authorise en AssistanceAgentClaimUpdateService:

```
@Override
public void authorise() {
    boolean status;
    int claimId;
    Claim claim;
    AssistanceAgent agent;

    agent = (AssistanceAgent) super.getRequest().getPrincipal().getRealmOfType(AssistanceAgent.class);

    claimId = super.getRequest().getData("id", int.class);
    claim = this.aacr.findClaimById(claimId);
    status = super.getRequest().getPrincipal().hasRealmOfType(AssistanceAgent.class) && agent.getId() =

    if (status && super.getRequest().hasData("leg")) {
        int legId = super.getRequest().getData("leg", int.class);
        Leg leg = this.aacr.findLegById(legId);
        List<Leg> legs = this.aacr.findAllLegsByAirlineId(agent.getAirline().getId());
        status = legId == 0 || leg != null && !leg.getIsDraftMode() || legs.contains(leg);
    }

    super.getResponse().setAuthorised(status);
}
```

Figura 1: Estado inicial del código

```
    if (status && super.getRequest().hasData("leg")) {
        int legId = super.getRequest().getData("leg", int.class);
        Leg leg = this.aacr.findLegById(legId);
        List<Leg> legs = this.aacr.findAllLegsByAirlineId(agent.getAirline().getId())
        status = legId == 0 || leg != null && !leg.getIsDraftMode() || legs.contains(
    }

    super.getResponse().setAuthorised(false);
}
```

Figura 2: Mutación 1

```
FAILED POST /assistance-agent/claim/update (request-id="bcf74da9-1f21-4c44-88ab-e906d7eae34e",
input="id=134&version=0&registrationMoment=2022%2F03%2F12+10%3A00&passengerEmail=blublublu%40gmail.com&description=El+piloto
+se+ha+puesto+a+hacer+maniobras+acrob%C3%A1ticas.+Si+no+hay+video+no+me+lo+creo
+chaval.&claimType=FLIGHT_ISSUES&indicator=REJECTED&leg=75&isPublished=true"): Expected 'status' to be '200', but got '500'.
```

Figura 3: Fallo detectado

El sistema no permite realizar ningún tipo de operación update ya que se le deniega el acceso con un 500 automáticamente.

- Mutación 2, cambiar condición de un validador en AssistanceAgentClaimUpdateService:

```
@Override
public void validate(final Claim claim) {
    if ((claim.getIndicator() == ClaimState.ACCEPTED || claim.getIndicator() == ClaimState.REJECTED) && !claim.getIsPublished())
        super.state(false, "isPublished", "assistance-agent.claim.create.should-be-publised");
    if (claim.getIndicator() == ClaimState.IN_PROGRESS && claim.getIsPublished())
        super.state(false, "indicator", "assistance-agent.claim.create.cant-be-published");
}
```

Figura 4: Estado inicial del código

```
@Override
public void validate(final Claim claim) {
    if (!(claim.getIndicator() == ClaimState.ACCEPTED || claim.getIndicator() == ClaimState.REJECTED) && !claim.getIsPublished())
        super.state(false, "isPublished", "assistance-agent.claim.create.should-be-publised");
    if (claim.getIndicator() == ClaimState.IN_PROGRESS && claim.getIsPublished())
        super.state(false, "indicator", "assistance-agent.claim.create.cant-be-published");
}
```

Figura 5: Mutación 2

```
ERROR Exception: org.springframework.orm.jpa.vendor.HibernateJpaDialect.convertHibernateAccessException
(HibernateJpaDialect.java:315): Object of class [acme.entities.claims.Claim] with identifier [133]: optimistic locking
failed; nested exception is org.hibernate.StaleObjectStateException: Row was updated or deleted by another transaction (or
unsaved-value mapping was incorrect) : [acme.entities.claims.Claim#133]
org.hibernate.event.internal.DefaultMergeEventListener.entityIsDetached(DefaultMergeEventListener.java:341): Row was updated
or deleted by another transaction (or unsaved-value mapping was incorrect) : [acme.entities.claims.Claim#133]
FAILED POST /assistance-agent/claim/update (request-id="8b05a726-ae8b-4519-ae49-8815e9daff7a",
input="id=133&version=1&registrationMoment=2022%2F03%2F11+10%3A00&passengerEmail=blebleble%40gmail.com&description=El
```

Figura 6: Fallo detectado

El fallo nos esta diciendo que se están realizando operaciones con valores incorrectos y que no cumplen con lo establecido.

- Mutación 3, eliminar una validación en AssistanceAgentClaimDeleteService:

```
@Override
public void validate(final Claim claim) {
    int claimId;
    claimId = super.getRequest().getData("id", int.class);
    List<TrackingLog> tl = this.aacr.findTrackingLogsByClaimId(claimId);
    super.state(tl.isEmpty(), "*", "assistance-agent.claim.delete.has-log");
}
```

Figura 7: Estado inicial del código

```
@Override
public void validate(final Claim claim) {
}
```

Figura 8: Mutación 3

```
WARN  Error: 1451-23000: Cannot delete or update a parent row: a foreign key constraint fails (`acme-ans-d04-
test`.`tracking_log`, CONSTRAINT `FK7foj49c7eaukl4c0dc2foe6mx` FOREIGN KEY (`claim_id`) REFERENCES `claim` (`id`))
WARN  SQL Error: 1451, SQLState: 23000
ERROR (conn=16) Cannot delete or update a parent row: a foreign key constraint fails (`acme-ans-d04-test`.`tracking_log`,
CONSTRAINT `FK7foj49c7eaukl4c0dc2foe6mx` FOREIGN KEY (`claim_id`) REFERENCES `claim` (`id`))
ERROR Exception: org.springframework.orm.jpa.vendor.HibernateJpaDialect.convertHibernateAccessException
(HibernateJpaDialect.java:276): could not execute statement; SQL [n/a]; constraint [null]; nested exception is
org.hibernate.exception.ConstraintViolationException: could not execute statement
```

Figura 9: Fallo detectado

- Mutación 4, cambiar los límites de los strings en los tracking logs:

```java
@Mandatory
@ValidString(max = 50)
@Automapped
private String              step;

@Mandatory
@ValidScore
@Automapped
private Double              resolutionPercentage;

@Mandatory
@ValidString
@Automapped
private String              resolution;
```

Figura 10: Estado inicial del código

```java
@Mandatory
@ValidString(max = 30)
@Automapped
private String              step;

@Mandatory
@ValidScore
@Automapped
private Double              resolutionPercentage;

@Mandatory
@ValidString(max = 30)
@Automapped
private String              resolution;
```

Figura 11: Mutación 4

```
Loaded 278 requests from .\src\test\resources\assistance-agent\tracking-log\create.safe
Resetting application (clear schema, populate sample, reset clock, reset randomiser).
ERROR Validating 'tracking-log-01' ... FAILED. 'resolution': Length must be between 0 and 30.
ERROR Validating 'tracking-log-02' ... FAILED. 'step': Length must be between 0 and 30.
ERROR Validating 'tracking-log-03' ... FAILED. 'resolution': Length must be between 0 and 30.
ERROR Validating 'tracking-log-04' ... FAILED. 'step': Length must be between 0 and 30. 'resolution': Length must be betwee
0 and 30.
ERROR Validating 'tracking-log-06' ... FAILED. 'resolution': Length must be between 0 and 30. 'step': Length must be betwee
0 and 30.
```

Figura 12: Fallo detectado

- Mutación 5, eliminación de campos en el bind en AssistanceAgentTrackingLogCreateService:

```java
@Override
public void bind(final TrackingLog trackingLog) {
    super.bindObject(trackingLog, "step", "resolutionPercentage", "resolution", "claim", "isPublished");
}
```

Figura 13: Estado inicial del código

```java
@Override
public void bind(final TrackingLog trackingLog) {
    super.bindObject(trackingLog);
}
```

Figura 14: Mutación 5

```
FAILED POST /assistance-agent/tracking-log/create (request-id="a283390a-dd2d-44ec-8ad2-33a1168728f5",
input="id=0&version=0&lastUpdateMoment=2025%2F01%2F01+00%3A00&step=kagmsng&resolutionPercentage=60.0&resolution=fafgegsg&cla
im=133&isPublished=true"): Expected 'payload' to be '{claim=[{"key":"0","label":"----","selected":false,"sealed":true},
{"key":"132","label":"blablabla@gmail.com","selected":false,"sealed":true},
{"key":"133","label":"blebleble@gmail.com","selected":true,"sealed":true},
{"key":"134","label":"blublublu@gmail.com","selected":false,"sealed":true},
{"key":"138","label":"rassclat@genda.com","selected":false,"sealed":true}], id=0, isPublished=true, isPublished$error=This
tracking-log can't be published because his claim isn't published., lastUpdateMoment=2025/01/01 00:00, resolution=fafgegsg,
resolutionPercentage=60.00, service=160, step=kagmsng, version=0}', but got '{claim=
[{"key":"0","label":"----","selected":true,"sealed":true},
{"key":"132","label":"blablabla@gmail.com","selected":false,"sealed":true},
{"key":"133","label":"blebleble@gmail.com","selected":false,"sealed":true},
{"key":"134","label":"blublublu@gmail.com","selected":false,"sealed":true},
{"key":"138","label":"rassclat@genda.com","selected":false,"sealed":true}], claim$error=May not be null., id=0,
isPublished=false, lastUpdateMoment=2025/01/01 00:00, resolution=fafgegsg, resolution$error=May not be null.,
resolutionPercentage=60.0, resolutionPercentage$error=May not be null., service=160, step=kagmsng, step$error=May not be
null., version=0}'.
```

Figura 15: Fallo detectado

El fallo ha resultado en que no se otenian los datos de la base de datos y por tanto no se pueden hacer la operaciones con normalidad.

Conclusiones

Como conclusión de este reporte decir que gracias a los tests se han encontrado diversas vulnerabilidades en el código y ciertas validaciones que no estaban haciendo lo correcto, con lo cual la fase de testing ha sido todo un éxito y ha realizado su función de manera plena. También se ha podido observar qué puntos de las operaciones realizadas por el sistema son los más costosos en cuanto a tiempo y recursos gracias al análisis de los datos.

Bibliografía

Intencionalmente en blanco