Un dibujo con letras

El contenido generado por IA puede ser incorrecto.

1. **Resumen Ejecutivo**

En este informe presento las pruebas funcionales y de rendimiento que llevé a cabo sobre mi estudiante. El objetivo fue asegurar que todas las funcionalidades se comportaran según lo esperado y evaluar la rapidez con la que responde el sistema en condiciones normales.

Para las pruebas funcionales, organicé los casos de prueba por funcionalidad. Cada caso se centra en una función específica y fue esencial para verificar que la aplicación se comporta correctamente.

Para las pruebas de rendimiento, seguí la metodología indicada en la guía de la sesión: recopilé los tiempos de ejecución a partir de los archivos .trace y procesé los datos utilizando Excel. Generé gráficos y calculé intervalos de confianza del 95 % para evaluar si los tiempos de respuesta del sistema se mantenían dentro de los límites aceptables. Las pruebas se ejecutaron en dos configuraciones distintas: una utilizando la base de datos sin ningún índice adicional, y otra con los índices relevantes aplicados. Posteriormente, realicé una comparación estadística entre ambas configuraciones para determinar el impacto del uso de índices en el rendimiento. Finalmente, probé los tiempos de ejecución en otro pórtatil.

En resumen, este informe refleja las pruebas que he realizado sobre las funcionalidades clave de la aplicación, respaldadas por datos de rendimiento que ofrecen una comprensión sólida del comportamiento del sistema en condiciones reales.

1. **Tabla de revisiones**

| **Versión** | **Fecha** | **Descripción** |
| --- | --- | --- |
| **1.0** | **26/05/25** | **Versión inicial** |

1. **Introducción**

El presente informe de pruebas tiene como objetivo documentar de forma clara y estructurada el proceso seguido para la creación y ejecución de los test aplicados a nuestra aplicación, ACME-ANS. En particular, se detallan los análisis correspondientes a los requisitos 8 y 9 asignados al integrante Student 3 del grupo de trabajo. A lo largo del documento se describen tanto los procedimientos llevados a cabo como los resultados obtenidos durante las pruebas funcionales y de rendimiento.

Para la elaboración de este informe se ha seguido como guía el esquema propuesto en el documento anexo proporcionado durante el curso. Dicha estructura nos ha permitido organizar el contenido de forma coherente y comprensible, priorizando la claridad en la presentación de los datos y conclusiones.

Dentro del informe, destacan especialmente dos bloques fundamentales. En primer lugar, el apartado dedicado a las pruebas funcionales, donde se enumeran los casos de prueba implementados, se analiza el porcentaje de código cubierto mediante dichas pruebas y se ofrece una justificación razonada para aquellas secciones del código que no han podido ser cubiertas.

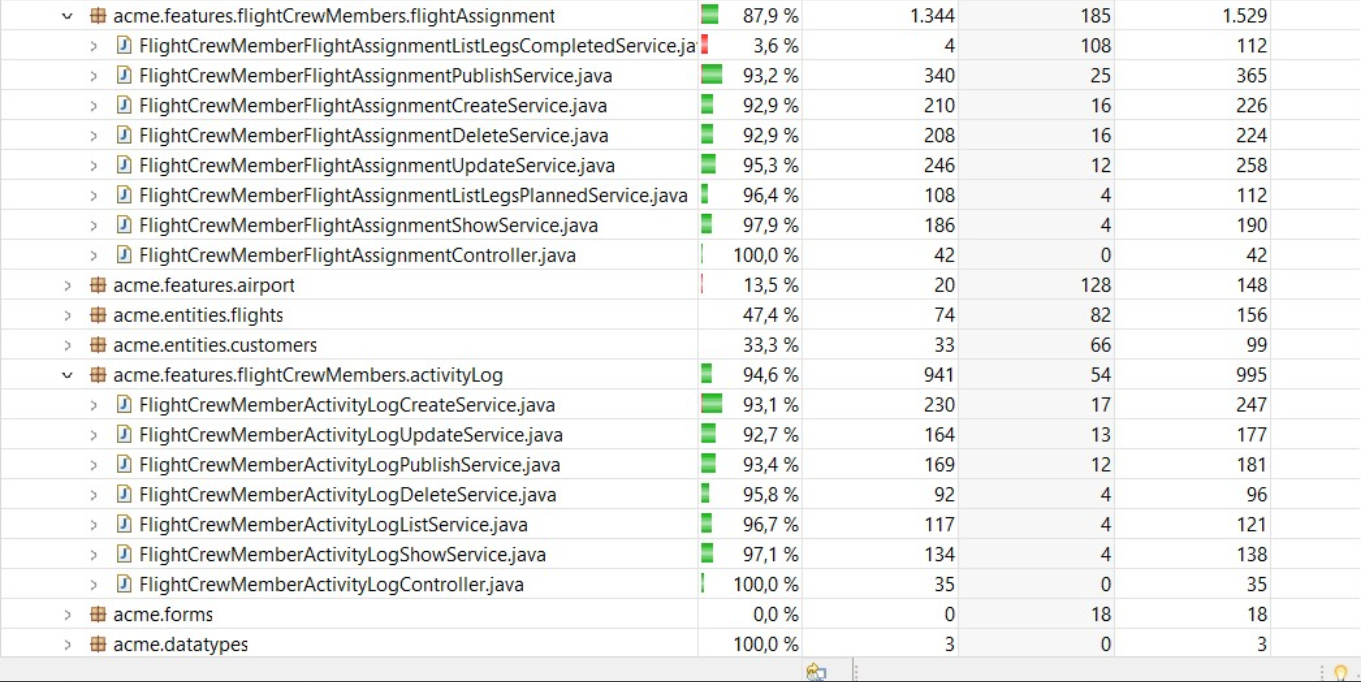
En segundo lugar, el capítulo de pruebas de rendimiento incluye un análisis comparativo del comportamiento de la aplicación en diferentes equipos, así como el impacto en el tiempo de ejecución tras la incorporación de índices en ciertas entidades de la base de datos. Esta modificación fue introducida con el objetivo de optimizar el rendimiento general de la aplicación, y los resultados obtenidos son analizados mediante gráficas e intervalos de confianza que permiten respaldar nuestras conclusiones de forma estadística.

Finalmente, la estructura del documento se desarrolla en los siguientes capítulos: primero, se expone el listado de pruebas funcionales realizadas; luego, se presentan las pruebas de rendimiento junto con el análisis estadístico correspondiente; y por último, se incluyen las conclusiones derivadas del proceso de testing.

1. **Testing Funcional**

En esta sección del documento se pretende describir un listado con los casos de prueba realizados, agrupados por características, para las entidades FlightCrewMembers y ActivityLog. Para cada caso de prueba, se va a proporcionar una descripción y cuál es su eficacia para detectar errores.

Para probar las entidades obligatorias del Student 3, se han aplicado pruebas “.safe” y “.hack”. Para las primeras se ha probado que las entidades se listen, muestren y permitan un CRUD válido y funcional, permitiendo todo tipo de valores válidos y rechazando aquellos que no cumplen con la validación. Por otro lado, para las segundas, generalmente se han dividido en pruebas de “url” y en pruebas de valores, principalmente por dos motivos. El primero es que solía dar problemas hacer las dos pruebas juntas, y el segundo es que permitía ser mucho más específico en cada una de las pruebas. Del mismo modo, se ha buscado obtener la mayor variabilidad posible en los casos de prueba. En muchos casos se han realizado los casos de prueba con usuarios distintos para asegurar que todos los elementos se renderizan y funcionan correctamente tanto para usuarios nuevos como ya existentes. Por último, se muestra una captura con la cobertura total alcanzada. En lo que respecta a los paquetes globales:



* Casos de prueba por característica:
  + Para FlightAssignment

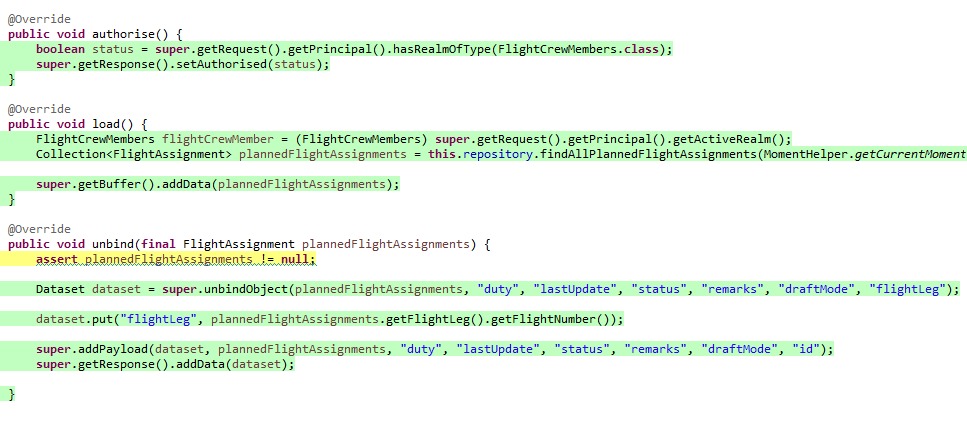
Comencemos especificando los casos de pruebas para las características “flight-crew-members/flight-assignment/list-planned”.

| Caso de prueba | Tipo | Descripción | Eficacia |
| --- | --- | --- | --- |
| list-planned.safe | Renderizado | Se prueba que el listado se muestre correctamente para asignaciones completadas y planeadas. |  |
| list-planned.hack | Hack | Se prueba que miembros con otro rol no puedan ver el listado | No se detectó ninguna incidencia ya que se realizaba una validación correcta en el “authorised”. |

La funcionalidad de listado de *legs* (tramos de vuelo) completadas ha sido desarrollada e implementada correctamente en el entorno de desarrollo (Eclipse). Esta funcionalidad está diseñada para identificar y mostrar aquellas *legs* cuya fecha corresponde a un momento anterior al actual, considerando que dichas *legs* ya han sido completadas.

Para el correcto funcionamiento de esta característica, es necesario que el conjunto de datos utilizado incluya al menos una *leg* con una fecha en el pasado. En el archivo CSV proporcionado, no se han incluido *legs* con estas características, por lo que no se ha podido observar en ejecución la salida esperada de esta funcionalidad en esta fase.

No obstante, la lógica se encuentra implementada y lista para ser verificada una vez se disponga de datos históricos adecuados.

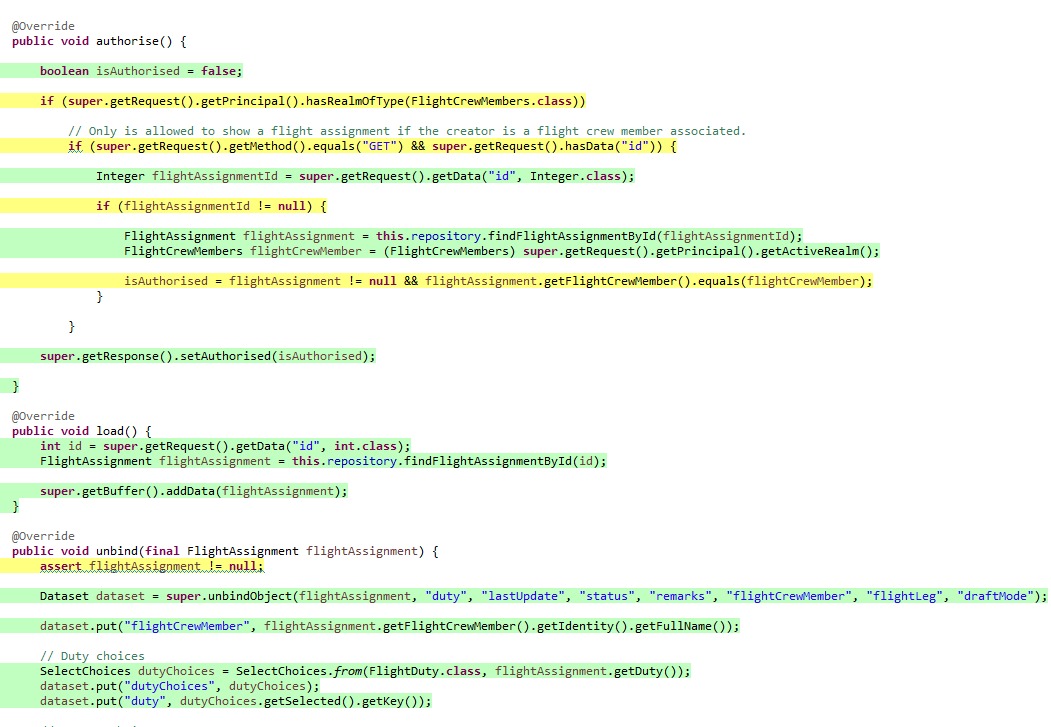


Todas las líneas de código han sido ejecutadas a excepción de:

-Assert plannedFlightAssignment!= null. Es una línea obligatoria que verifica que plannedFlightAssignment no sea nulo.

Pasemos a “flight-crew-members/flight-assignment/show?id=xx”:

| Caso de prueba | Tipo | Descripción | Eficacia |
| --- | --- | --- | --- |
| show.safe | Renderizado | Se prueba que una asignación se renderice correctamente sin ningún error. | No ha detectado ninguna incidencia ya que el código del servicio estaba correctamente implementado. |
| show.hack | Hack | Permite probar: -Miembros con otro rol no puedan ver los detalles de una asignación. -Miembros con mismo rol no puedan ver asignaciones que no le corresponden. | No se detectó ninguna incidencia ya que se realizaba una validación correcta en el “authorised”. |



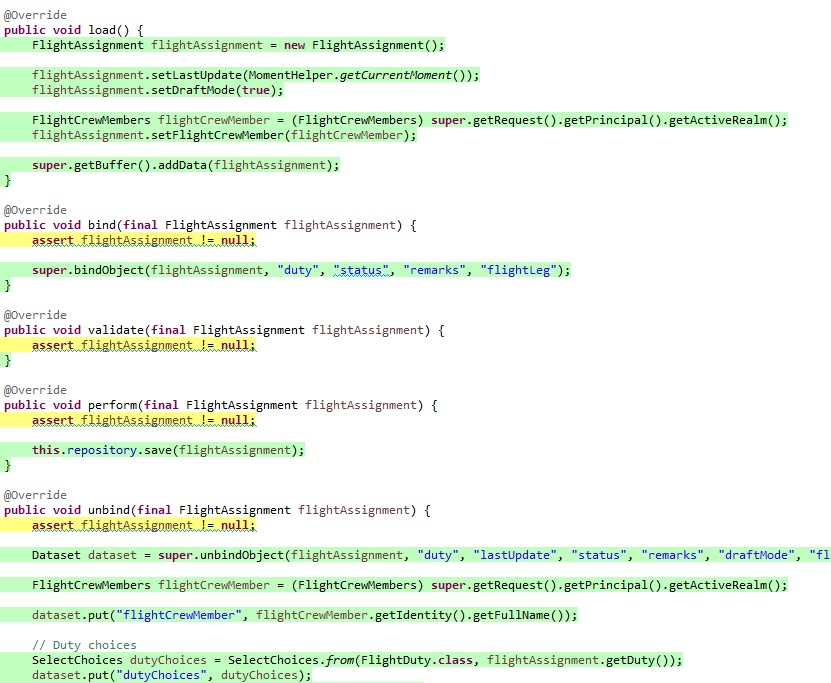
- La línea:

status **super**.getRequest().getPrincipal().hasRealm(customer) && isFlightAccessible;

Es una línea de condición que no se puede cumplir al completo ya que la condición 2 solo se puede verificar si vemos que el usuario efectivamente tiene el rol customer.

En lo que respecta a “flight-crew-members/flight-assignment/create” (GET y POST):

| Caso de prueba | Tipo | Descripción | Eficacia |
| --- | --- | --- | --- |
| create.safe | Casos positivos y negativos del formulario | Se prueba que todos los campos del formulario estén correctamente validados, rechazando los valores no admitidos por el modelo. | No se detectó ninguna incidencia ya que todos los campos estaban correctamente validados. |
| create.hack | Hack | Permite probar: -Miembros con otro rol no puedan acceder al formulario de creación - No se puedan asignar etapas no válidas a la asignación (borradores).  - No se pueda asignar otro miembro que no sea el “logueado” a la asignación.  - Atributos de selección no puedan ser modificados con valores ilegales . | No se detectó ninguna incidencia ya que se realizaba una validación correcta en el “authorised”. |



Todas las líneas de código han sido ejecutadas a excepción de:

- assert flightAssignment!= null. Es una línea obligatoria que verifica que flightAssignment no sea nulo.

En lo que respecta a “flight-crew-members/flight-assignment/update?assignmentId=${id}” (GET y POST):

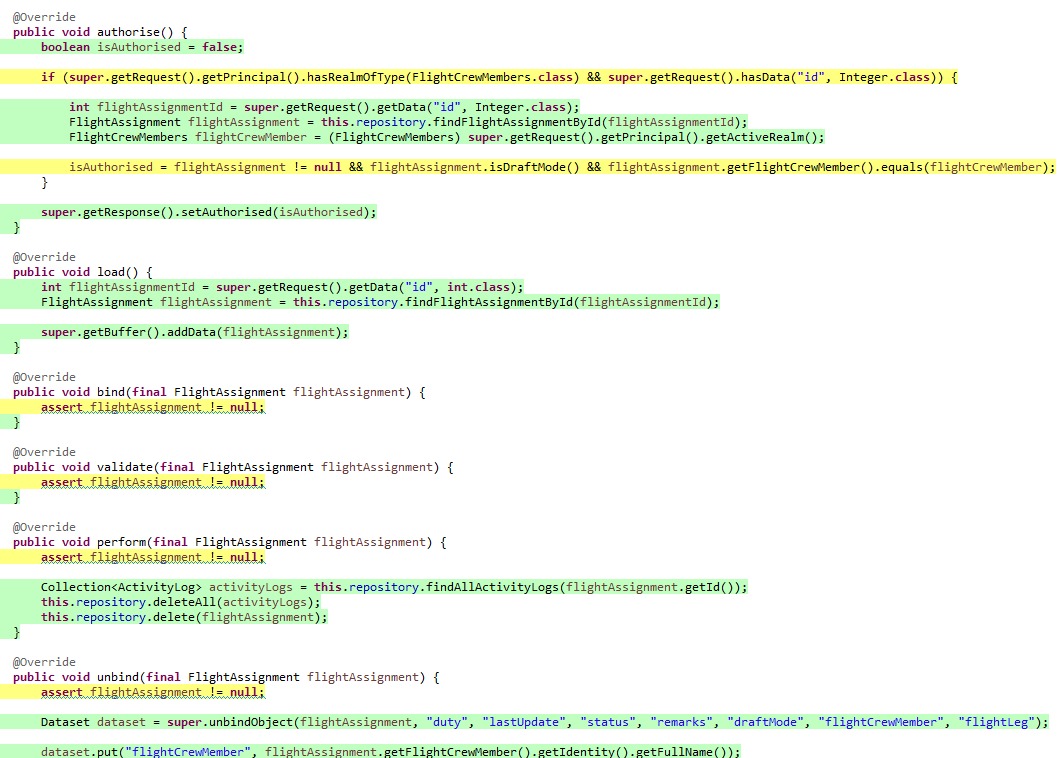
| Caso de prueba | Tipo | Descripción | Eficacia |
| --- | --- | --- | --- |
| update.safe | Casos positivos y negativos del formulario | Se prueba que todos los campos del formulario estén correctamente validados, rechazando los valores no admitidos por el modelo. | No se detectó ninguna incidencia ya que todos los campos estaban correctamente validados. |
| update.hack | Hack | Permite probar: -Miembros con otro rol no puedan acceder a los detalles de la asignación por medio del “update”. - Miembros del mismo rol no puedan acceder a los detalles de una asignación que no les pertenece por medio del “update”. - Asignaciones ya publicadas no puedan ser actualizadas. | No se detectó ninguna incidencia ya que se realizaba una validación correcta en el “authorised”. |

En lo que respecta a “flight-crew-members/flight-assignment/publish” (GET y POST):

| Caso de prueba | Tipo | Descripción | Eficacia |
| --- | --- | --- | --- |
| publish.safe | Casos positivos y negativos del formulario | Se prueba que todos los campos del formulario estén correctamente validados, rechazando los valores no admitidos por el modelo. | No se detectó ninguna incidencia ya que todos los campos estaban correctamente validados. |
| publish.hack | Hack | Permite probar: -Miembros con otro rol no puedan acceder a los detalles de la asignación por medio del “publish”. - Miembros del mismo rol no puedan acceder a los detalles de una asignación que no les pertenece por medio del “publish”. - Asignaciones ya publicadas no puedan volver a publicarse. | No se detectó ninguna incidencia ya que se realizaba una validación correcta en el “authorised”. |

Por último, para “flight-crew-members/flight-assignment/delete” (GET y POST):

| Caso de prueba | Tipo | Descripción | Eficacia |
| --- | --- | --- | --- |
| delete.safe | Casos positivos y negativos del formulario | -Se prueba que una asignación se borre correctamente, incluso al meter valores diferentes de los mostrados, bien positivos o negativos. - Asignaciones con logs asociados se borren correctamente. | No se detectó ninguna incidencia ya que todos los campos estaban correctamente validados. |
| delete.hack | Hack | Permite probar: -Miembros con otro rol no puedan acceder a los detalles de la asignación por medio del “delete”. - Miembros del mismo rol no puedan acceder a los detalles de una asignación que no les pertenece por medio del “delete”. - Asignaciones ya publicadas no puedan ser eliminadas. | No detectó ninguna incidencia porque todo estaba bien validado e implementado. |



Se ejecutan todas las líneas al completo, menos aquella relativa a las condiciones del authorise. No es refactorizable ya que no podemos eliminar ninguna para asegurar el correcto cumplimiento, no se puede combinar al completo porque si no se cumple la primera condición no se puede probar la segunda.

* + Para ActivityLog

Comencemos especificando los casos de pruebas para las características “flight-crew-members/activity-log/list?masterId=xx”

| Caso de prueba | Tipo | Descripción | Eficacia |
| --- | --- | --- | --- |
| list.safe | Renderizado | Se prueba que el listado de incidentes se muestre correctamente para asignaciones completadas, ya que solo se pueden registrar incidentes cuando la etapa ha terminado | Permitió detectar una discrepancia en los requisitos, ya que hasta entonces se permitía listar logs para etapas que todavía no habían terminado |
| list.hack | Hack | Se prueba que: -Miembros con otro rol no puedan ver un listado de incidencias. -Miembros del mismo rol no puedan ver un listado de incidencias de una asignación que no les pertenece. - Miembros del rol correcto no puedan ver el listado de incidencias de asignaciones no publicadas o planeadas. | No se detectó ninguna incidencia ya que se realizaba una validación correcta en el “authorised”. |

Pasemos a “flight-crew-members/activity-log/show?id=xx”:

| Caso de prueba | Tipo | Descripción | Eficacia |
| --- | --- | --- | --- |
| show.safe | Renderizado | Se prueba que una incidencia se renderice correctamente sin ningún error. | No ha detectado ninguna incidencia ya que el código del servicio estaba correctamente implementado . |
| show.hack | Hack | Permite probar: -Miembros con otro rol no puedan ver los detalles de una incidencia. -Miembros con mismo rol no puedan ver incidencia de asignaciones que no le corresponden. | No se detectó ninguna incidencia ya que se realizaba una validación correcta en el “authorised”. . |

En lo que respecta a “flight-crew-members/activity-log/create?assignmentId=${assignmentId}” (GET y POST):

| Caso de prueba | Tipo | Descripción | Eficacia |
| --- | --- | --- | --- |
| create.safe | Casos positivos y negativos del formulario | Se prueba que todos los campos del formulario estén correctamente validados, rechazando los valores no admitidos por el modelo. En esta entidad es sencillo porque solo hay tres campos String modificables. | No se detectó ninguna incidencia ya que todos los campos estaban correctamente validados. |
| create.hack | Hack | Permite probar: -Miembros con otro rol no puedan acceder al formulario de creación de un incidente. - Miembros del mismo rol no puedan acceder al formulario de creación de un incidente asociado a una asignación que no les corresponde. - Miembros del rol correcto no puedan acceder al formulario de creación de un incidente asociado a una asignación en modo borrador o a una asignación cuya etapa no ha terminado | No detectó ninguna incidencia ya que el servicio estaba correctamente implementado. |

En lo que respecta a “flight-crew-members/activity-log/update” (GET y POST):

| Caso de prueba | Tipo | Descripción | Eficacia |
| --- | --- | --- | --- |
| update.safe | Casos positivos y negativos del formulario | Se prueba que todos los campos del formulario estén correctamente validados, rechazando los valores no admitidos por el modelo (principalmente la longitud al ser los campos String). | No se detectó ninguna incidencia ya que todos los campos estaban correctamente validados. |
| update.hack | Hack | Permite probar: -Miembros con otro rol no puedan ver los detalles de una incidencia mediante el “update”. -Miembros con mismo rol no puedan ver incidencia de asignaciones que no le corresponden mediante el “update”. - Miembros del rol correcto no puedan tratar de actualizar incidencias ya publicadas. | No detectó ninguna incidencia porque todo estaba bien validado e implementado. |

En lo que respecta a “flight-crew-members/ activity-log /publish” (GET y POST):

| Caso de prueba | Tipo | Descripción | Eficacia |
| --- | --- | --- | --- |
| publish.safe | Casos positivos y negativos del formulario | Se prueba que todos los campos del formulario estén correctamente validados, rechazando los valores no admitidos por el modelo (principalmente la longitud al ser los campos String). | No se detectó ninguna incidencia ya que todos los campos estaban correctamente validados. |
| publish.hack | Hack | Permite probar: -Miembros con otro rol no puedan ver los detalles de una incidencia mediante el “publish”. -Miembros con mismo rol no puedan ver incidencia de asignaciones que no le corresponden mediante el “publish.” - Miembros del rol correcto no puedan tratar de publicar incidencias ya publicadas. | No detectó ninguna incidencia porque todo estaba bien validado e implementado. |

Por último, para “flight-crew-members/activity-log /delete” (GET y POST):

| Caso de prueba | Tipo | Descripción | Eficacia |
| --- | --- | --- | --- |
| delete.safe | Casos positivos y negativos del formulario | Se prueba que una incidencia se borre correctamente bien al incluir valores correctos o incorrectos. | No se detectó ninguna incidencia. |
| delete.hack | Hack | Permite probar: -Miembros con otro rol no puedan acceder a los detalles de la incidencia por medio del “delete”. - Miembros del mismo rol no puedan acceder a los detalles de una incidencia que no les pertenece por medio del “delete”. - Incidencias ya publicadas no puedan ser borradas. | No detectó ninguna incidencia porque todo estaba bien validado e implementado. |

* Conclusiones finales

En definitiva, se han desarrollado un conjunto de pruebas que cubre la gran mayoría de líneas de código de las entidades del Student 3. Aunque haya métodos de validación que se pueden optimizar, se ha cumplido el objetivo de obtener un sistema robusto y seguro ante la gran mayoría de escenarios que se dan en el mundo real.

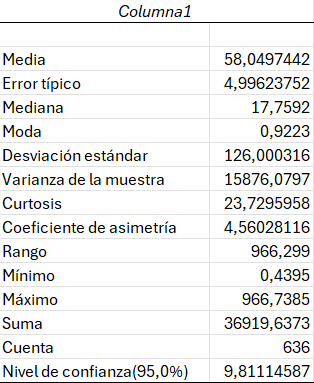
1. **Testing de Rendimiento**

Para el desarrollo de esta sección se va a emplear un equipo con un procesador AMD Ryzen 7 3750H with Radeon Vega Mobile Gfx con 16,0 GB de RAM

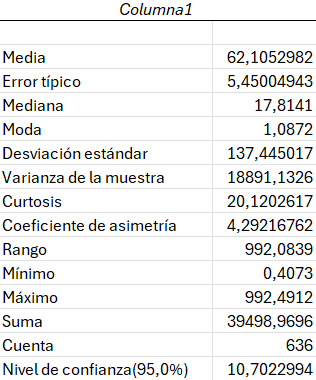
Para este estudio se va a emplear los casos de prueba que he ejecutado individualmente como Student 3, empleando la versión 25.5.0 del proyecto y del framework. En el primer lanzamiento, se emplearán índices en las entidades y en el segundo lanzamiento, no se emplearán, para ver si disminuye el rendimiento.

* Recogida de datos

Para la obtención de datos, tanto antes como después de implementar índices, se han seguido los pasos explicados en la teoría, analizando la información del “.trace” generado por la aplicación y filtrándola para obtener un fichero limpio del que poder obtener información útil como gráficos.



Datos estadísticos obtenidos a partir de la primera traza mejorada



Datos estadísticos obtenidos a partir de la segunda traza no mejorada



Datos estadísticos obtenidos a partir de la primera traza mejorada



Datos estadísticos obtenidos a partir de la segunda traza no mejorada

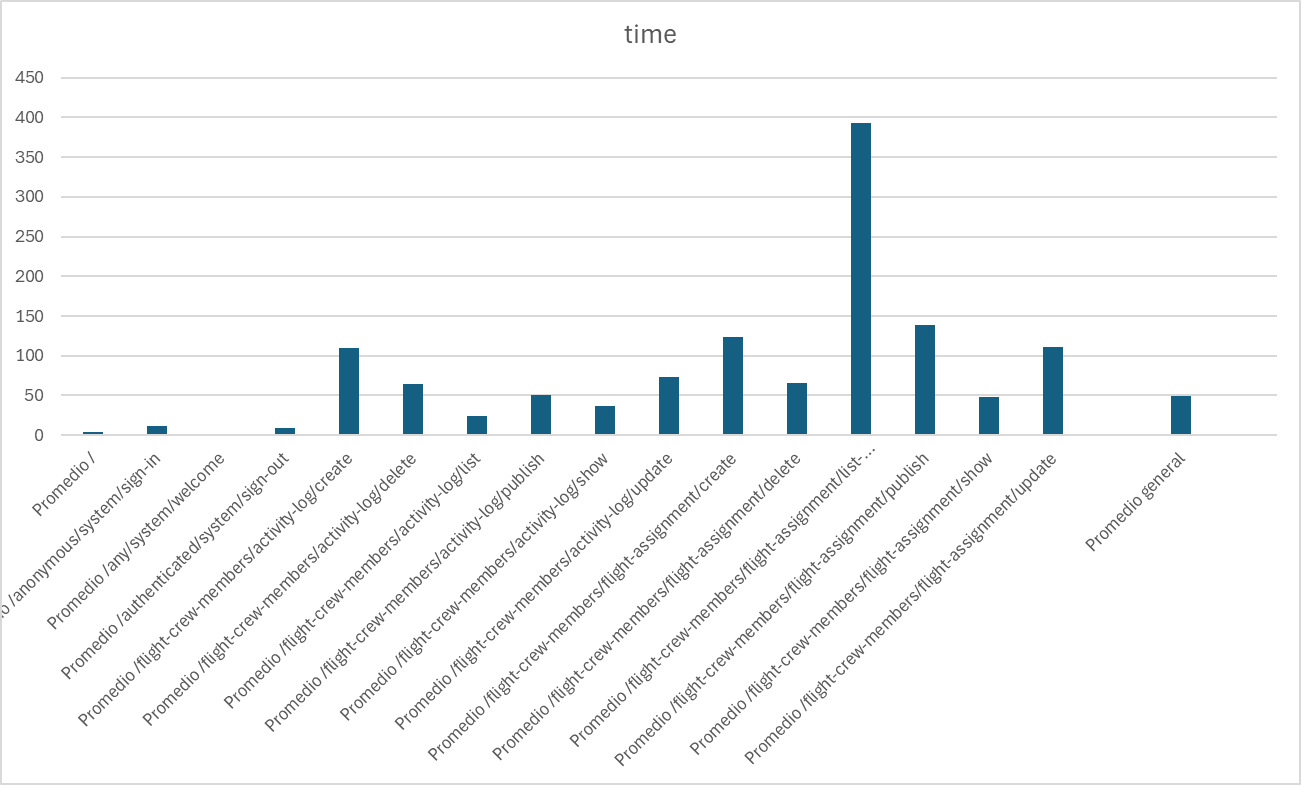


Gráfico obtenido a partir de la primera traza mejorada

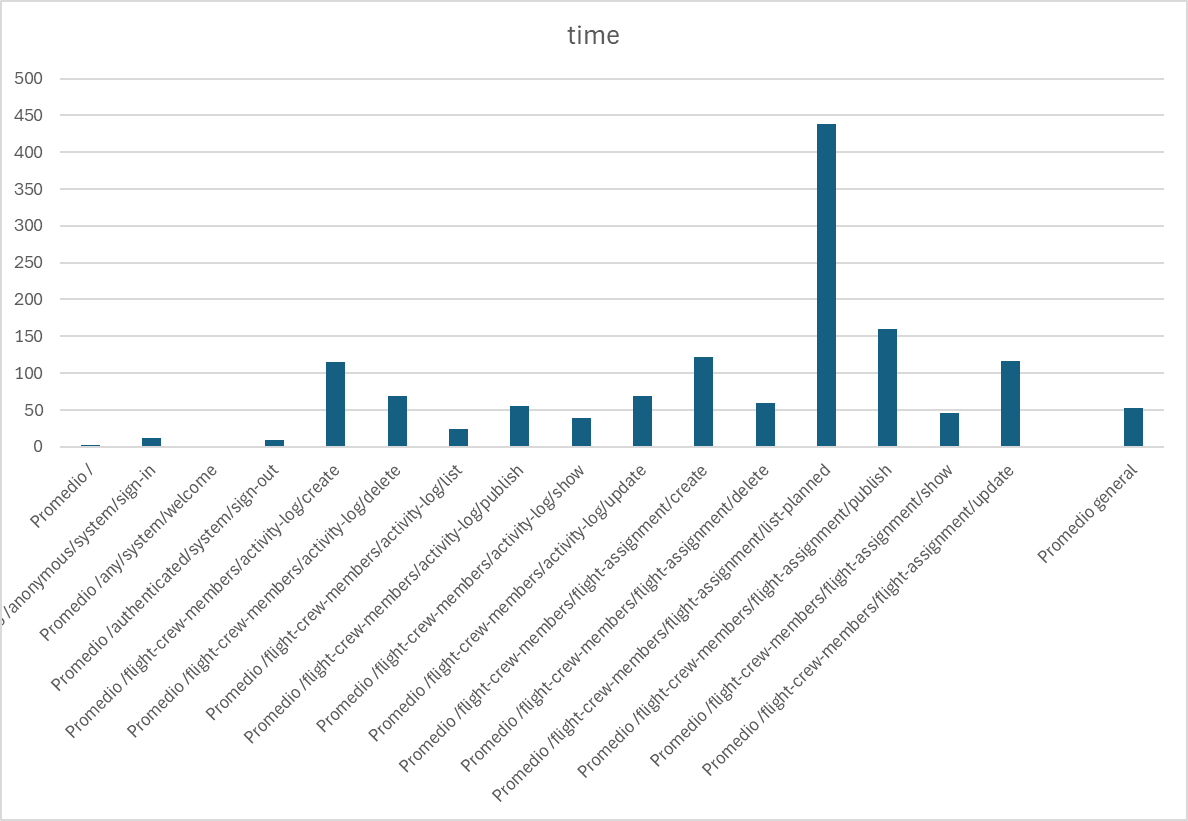
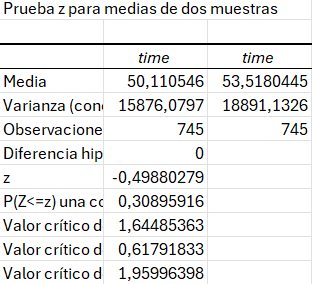


Gráfico obtenido a partir de la segunda traza no mejorada

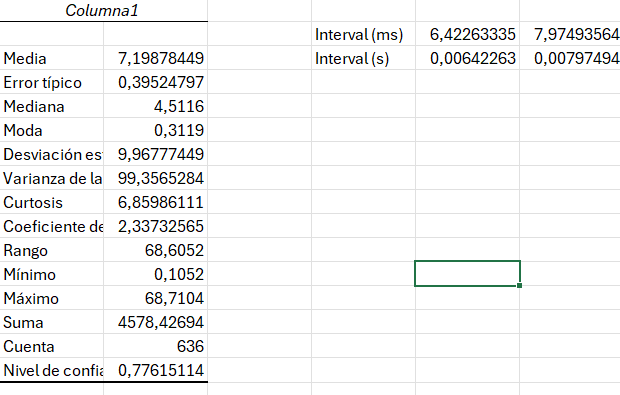
Como podemos observar, las trazas **“antes”** y **“después”** de la incorporación de índices presentan medias en torno a **58 ms** y **62 ms** respectivamente, con desviaciones estándar de **126 ms** y **137 ms**. Aunque hay un ligero incremento tanto en la media como en la dispersión tras el cambio, estas diferencias no son suficientemente grandes como para considerarse significativas en términos prácticos.

Por tanto, podemos concluir que, aunque la implementación de índices genera una ligera variación en las medidas estadísticas, **no tiene un impacto relevante en el tiempo de ejecución**.

Para finalizar con las pruebas de rendimiento, vamos a concluir este informe con una decisión firme sobre los resultados de las dos trazas generadas. Tras comparar el tiempo de las peticiones realizadas en ambas, hemos obtenido los siguientes datos:



Aunque se observa una leve diferencia entre las medias de ejecución antes y después de implementar índices (50,11 ms vs. 53,52 ms), dicha diferencia **no es estadísticamente significativa**. La prueba Z confirma que no hay evidencia suficiente para afirmar que los índices introducen una mejora o deterioro relevante en el rendimiento.



Por último, se ha realizado otro testing de rendimiento en otro dispositivo, que cuenta con este procesador: 12th Gen Intel(R) Core(TM) i7-12650H 2.70 GHz

El nuevo equipo tiene una media casi 8 veces menor que los tiempos anteriores Esto sugiere una mejora de rendimiento muy importante, no relacionada con los índices, sino con el hardware del segundo portátil.

Esto indica que el hardware es el factor determinante del rendimiento en este caso, mucho más que la inclusión de índices u optimizaciones a nivel de base de datos.