ACME AirNav Solutions

TESTING REPORT - Student 1

Grupo: C1.050
Miembros: Cristina Fernández Chica (criferchi@alum.us.es), Ángel Amo Sánchez (angamosan@alum.us.es), Candela Jazmín Gutiérrez González (cangutgon@alum.us.es), Marta Aguilar Morcillo (maragumor@alum.us.es) y Luis Emmanuel Chávez Malavé (luichamal@alum.us.es)
Repositorio: https://github.com/Cristinafernandezchica/Acme-ANS
Planning dashboard: https://github.com/users/Cristinafernandezchica/projects/1/views/1

Sevilla 24 mayo, 2025

TABLA DE CONTENIDOS

Resumen Ejecutivo

En este documento se presentan los diferentes procedimientos llevados a cabo a la hora de realizar el testing formal del proyecto llevado a cabo. Esto pasa por la generación de múltiples casos de prueba que traten las distintas posibilidades, siguiendo por el análisis de la funcionalidad del código en base a las mismas, y el análisis del rendimiento entre dos ordenadores (en mi caso no ha existido la posibilidad de realizarlo con dos ordenadores distintos, por lo que se ha añadido o restado aleatoriamente un 10% al tiempo medio de solicitud obtenido en mi ordenador) y en distintas condiciones, que para mi caso será la adición de índices en la base de datos.

Tabla de Revisiones

Número de revisión	Fecha	Descripción de revisión	Autor
1.0	24/05/2025	Primera versión del documento. Añadido el apartado completo de “Pruebas funcionales”	Cristina Fernández Chica
2.0	25/05/2025	Finalización del documento.	Cristina Fernández Chica

Introducción

En el informe se explicará cómo se ha realizado el testing funcional y el análisis del desempeño, indicando los posibles errores encontrados en el proceso. Eso se aplicará a los requisitos funcionales 8 y 9 del Student #1.

Para la redacción del mismo se seguirán los apartados indicados en los anexos proporcionados en la asignatura. Contará con dos apartados principales, las pruebas funcionales, donde indicaremos las pruebas realizadas por funcionalidad junto con la efectividad de las mismas, es decir, en qué medida nos han ayudado a detectar errores en el código, y las pruebas de rendimiento, con un intervalo de confianza del 95% al realizar pruebas antes y después de optimizar la búsqueda en base de datos con índices. Además, en este segundo apartado, se incluirán gráficos de tiempo empleado por funcionalidad. Por último se incluirá una conclusión en base a lo analizado en los apartados anteriores.

Pruebas

Pruebas funcionales

Para cada funcionalidad se han realizado pruebas positivas y negativas (archivos .safe) e intentos de hacking (archivos .hack). Se han seguido los pasos explicados en clase para la realización de las pruebas. Los nombres de los archivos en esencia son los mismos que se especifican en la tabla, algunos llevan añadido -mine por repetición de pruebas y posterior borrado de las inválidas.

manager/flight		
__.safe	**Descripción**	**Bugs**
list	Se listaron los vuelos del manager1, que tiene varios vuelos, del manager2, que tiene uno sin tramos (para probar el unbind), y del manager3 que no tiene vuelos asignados.	No se detectaron.
show	Se mostraron vuelos publicados y no publicados, con tramos y sin tramos.	No se detectaron.
create	Se probaron variaciones de datos válidos e inválidos en cada campo verificando validaciones hasta la creación de uno con todos los campos correctos.	No se detectaron.
update	Se probaron distintas variaciones de datos válidos e inválidos en cada campo verificando validaciones, probando a actualizar tanto vuelos con tramos como sin ningún tramo.	No se detectaron.
publish	Se probaron distintas variaciones de datos válidos e inválidos en cada campo para verificar validaciones. Se probó a publicar vuelos sin tramos, con varios tramos publicados o con uno solo. También, con algún tramo no publicado y con un tramo publicado pero que ya está en pasado.	No se detectaron.
delete	Se borraron vuelos con tramos y sin tramos.	No se detectaron.
all-data-test	Tras ejecutar el analizador se observaron algunos datos que faltaban por probar en pruebas anteriores y se hicieron en este test.	No se detectaron.
__.hack	**Descripción**	**Bugs**
list	Se intentó listar los vuelos de un manager en usuarios sin rol y con un rol incorrecto.	No se detectaron.
show	En manager1:	No se detectaron.

	→ Vuelo con id=500 (manager/flight/show?id=500) → Vuelo con id= (manager/flight/show?id=) → Vuelo sin id (manager/flight/show) En manager distinto al 1, no autenticado y rol distinto a manager: → Todas las anteriores. → Vuelo que no le pertenece (prueba en manager2) (manager/flight/show?id=67)	
create & create-more	En manager1: → Intento de actualización con create (cambiando la url del botón). En archivo create-more. (manager/flight/create?id=67) En usuario sin autenticar o con rol distinto: → Meter la dirección de creación de flights (manager/flight/create)	En un principio el segundo hackeo mencionado podía realizarse.
update	En manager1: → Vuelo con id=500 (inexistente) → Vuelo con id= (vacío) → Entrar a la url de actualización justo al entrar sin hacerlo desde un show previo. → Intento actualizar vuelo por url (/manager/flight/update?id=67) Desde otro manager: → Intento actualización de un vuelo que no es suyo Desde otro rol y usuario no autenticado: → Todo lo anterior	En un inicio se podían meter IDs de vuelos no existentes.
publish	En manager1: → Vuelo con id=500 (inexistente) → Vuelo con id= (vacío) → Entrar a la url de publicación justo al entrar sin hacerlo desde un show previo. → Intento publicar un vuelo por url (/manager/flight/publicate?id=67) Desde otro manager: → Intento publicación de un vuelo que no es suyo Desde otro rol y usuario no autenticado: → Todo lo anterior	Se podían incluir IDs de vuelos no existentes.
delete	Rol distinto de manager y usuario sin autenticar: → Meter la url de delete sola, con id vacío, con id nulo y con id de un vuelo. Rol de manager: → Meter la url sola sin pasar por show, con id vacío, con id nulo, con id de un vuelo.	Se podían meter IDs de vuelos no existentes.

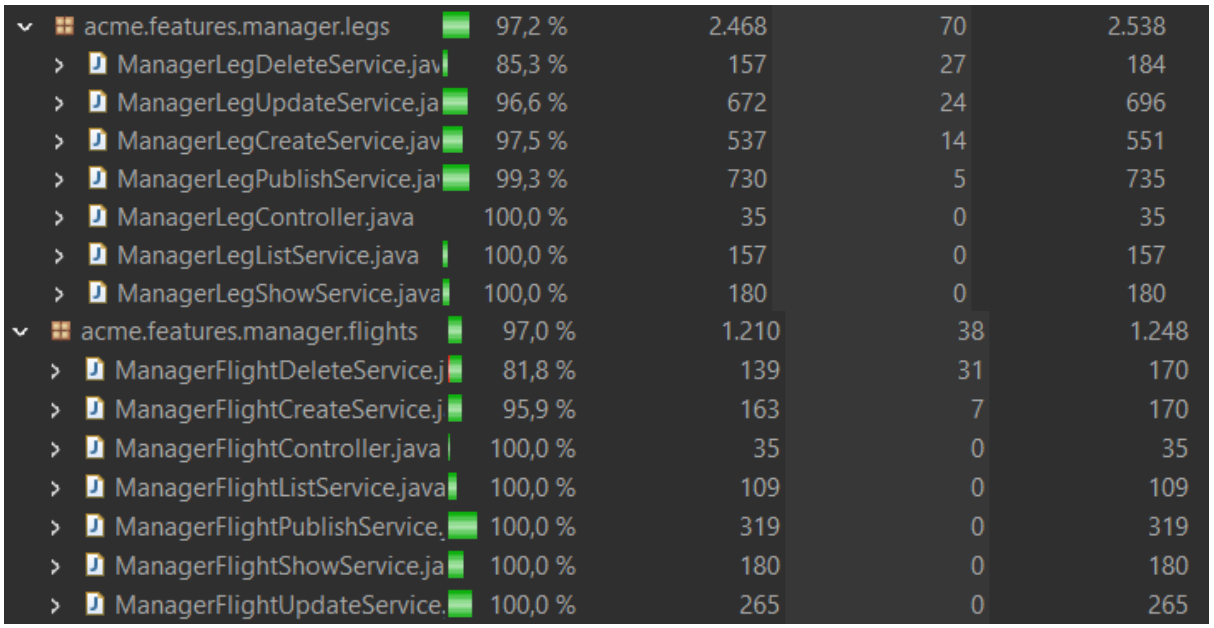
manager/leg		
__.safe	**Descripción**	**Bugs**
list	Se listaron los tramos de un vuelo con varios tramos, con un solo tramo y sin tramos. También, se listaron legs de vuelos publicados.	No se detectaron.
show	Se mostraron tramos no publicados y tramos publicados.	No se detectaron.
create	Se probaron variaciones de datos válidos e inválidos en cada campo verificando validaciones hasta la creación de un tramo con los datos correctos.	No se detectaron.
update	Se probaron distintas variaciones de datos válidos e inválidos en cada campo verificando las distintas validaciones.	No se detectaron.
publish	Se probaron distintas variaciones de datos válidos e inválidos en cada campo para verificar validaciones. Se probó a publicar vuelos sin tramos, con varios tramos publicados o con uno solo. También, con algún tramo no publicado y con un tramo publicado pero que ya está en pasado.	No se detectaron.
delete	Se borraron tramos no publicados.	No se detectaron.
all-data	Tras ejecutar el analizador se observaron algunos datos que faltaban por introducir en las pruebas anteriores y se hicieron en este test.	No se detectaron.
__.hack	**Descripción**	**Bugs**
list	En manager1: → Listado de tramos de un vuelo con id vacío, nulo y sin el id del vuelo en la url. Desde usuario distinto a manager1: → Listado de tramos de algún vuelo. (manager/leg/list?flightId=67) → Listado de tramos sin id de vuelo, de un vuelo inexistente o vacío. (manager/leg/list) (manager/leg/list?flightId=500) (manager/leg/list?flightId=) Desde otro manager, usuario no autenticado y con rol distinto: → Además de todo lo anterior, intento de listar los tramos de un vuelo que no les pertenece.	No se detectaron.
show	En manager1: → Intento de mostrar tramos sin id en la url, con id de un tramo inexistente y con id vacío.	No se detectaron.

	(manager/leg/show) (manager/leg/show?id=500) (manager/leg/show?id=) Rol manager distinto al 1, no autenticado y rol distinto a manager: → Además de todas las anteriores, intento de mostrar un tramo que no les pertenece.	
create	En manager1: → Dirección de creación con flightId vacío, de un vuelo inexistente o sin el flightId. (manager/leg/create?flightId=) (manager/leg/create?flightId=500) (manager/leg/create) → Poner un id inválida o vacía en los select de aeropuertos y aircraft. → Cambiar propiedad status (readonly fija). → Entrar en crear un tramo cuando el vuelo ya está publicado. → Intento de actualizar un tramo existente al crear el vuelo cambiando la url por el botón. (manager/leg/create?id=224). En usuario sin autenticar y con rol distinto: → Meter la dirección de creación de tramos (manager/leg/create?flightId=67). En otro manager: → Crear un tramo en un vuelo que no es suyo.	Se encontraron errores al poner IDs inválidas en los select. La última mencionada se podía realizar.
update	En manager1: → Tramos con id=500 (inexistente) → Tramo con id= (vacío) → Entrar a la url sin id (manager/leg/update) → Entrar a la url de actualización justo al entrar sin hacerlo desde un show previo. → Intento de actualizar un tramo por url (/manager/leg/update?id=226) → Poner un id inválida o vacía en los select de aeropuertos y aircraft. → Cambiar propiedad status cuando no está publicada, que es un readonly, y cuando está publicada que es un select. → Cuando un vuelo está publicado, intentar cambiar algún otro valor distinto a status, que son todos readonly. → Entrar en crear un tramo cuando el vuelo ya está publicado. Desde otro manager: → Actualización de un vuelo que no es suyo. Desde otro rol y usuario no autenticado: → Todas las mencionadas que implican ids vacíos, nulos, etc.	Se encontraron errores al poner IDs inválidas en los select.
publish	En manager1:	Se encontraron

	→ Tramo con id=500 (inexistente) → Tramo con id= (vacío) → Entrar a la url de publicación justo al entrar sin hacerlo desde un show previo. → Intento publicar un tramo por url (/manager/flight/publicate?id=226) → Poner un id inválida o vacía en los select de aeropuertos y aircraft. → Cambiar propiedad status al publicar, esta es un readonly. → Intentar publicar un tramo ya publicado, Desde otro manager: → Intento publicación de un vuelo que no es suyo Desde otro rol y usuario no autenticado: → Todo lo mencionado para el manager1 sobre cambios de id en la url.	errores al poner IDs inválidas en los select.
delete	En manager: → Entrar a la url de borrado justo al entrar sin hacerlo desde un show previo. → Poner id de leg no existente. → Poner id vacía. Rol distinto de manager y usuario sin autenticar: → Meter la url de delete sola, con id vacío, con id nulo y con id de un tramo cualquiera.	Al poner de un tramo no existente no funcionaba correctamente.

Recubrimiento del código

Las pruebas se pueden considerar lo suficientemente buenas pues la cobertura tanto de las funcionalidades de flight como de leg, rondan el 97%. Por tanto, podemos decir que el código se ha probado en profundidad intentando buscar todos los casos posibles en cada una de las funcionalidades.



Análisis de desempeño

En primer lugar, el requisito de rendimiento que se va a tener en cuenta para este análisis es que la aplicación debe responder a las peticiones en menos de 1.5 segundos de media.

A continuación, vamos a analizar las estadísticas básicas sobre el rendimiento de la aplicación antes y después de optimizar los índices, utilizando también sus respectivas gráficas de tiempo por funcionalidad. Para ello, se ha realizado un contraste de hipótesis mediante Z-test que demuestra que los cambios si tienen impacto estadístico.

	Before	*After*
Media	10,8099975	10,62311066
Varianza (conocida)	277,3094765	245,1847813
Observaciones	960	960
Diferencia hipotética de las medias	0	
z	0,253322395	
P(Z<=z) una cola	0,400009546	
Valor crítico de z (una cola)	1,644853627	
Valor crítico de z (dos colas)	0,800019092	
Valor crítico de z (dos colas)	1,959963985	

Podemos observar que el p-valor obtenido al realizar la prueba Z (dos colas) es de 0.85599118. Puesto que hemos considerado un nivel de confianza del 95%, el nivel de significancia es 0.05. Dado que el p-valor no es menor que la significancia, lo que quiere decir que tenemos una hipótesis nula, pues no existe una diferencia significativa en los tiempos de ejecución antes y después de la adición de los índices. En las siguientes tablas, se observa los tiempos antes y después de aplicar los cambios (índices):

BEFORE	
Media	10,8099975
Error típico	0,537460734
Mediana	4,9254
Moda	1,2188
Desviación estándar	16,65261178
Varianza de la muestra	277,3094791
Curtosis	11,91176468
Coeficiente de asimetría	2,968071753
Rango	166,0065
Mínimo	0,716
Máximo	166,7225
Suma	10377,5976
Cuenta	960
Nivel de confianza(95,0%)	1,054734846

Interval (ms)	9,755262654	11,86473235
Interval(s)	0,009755263	0,011864732

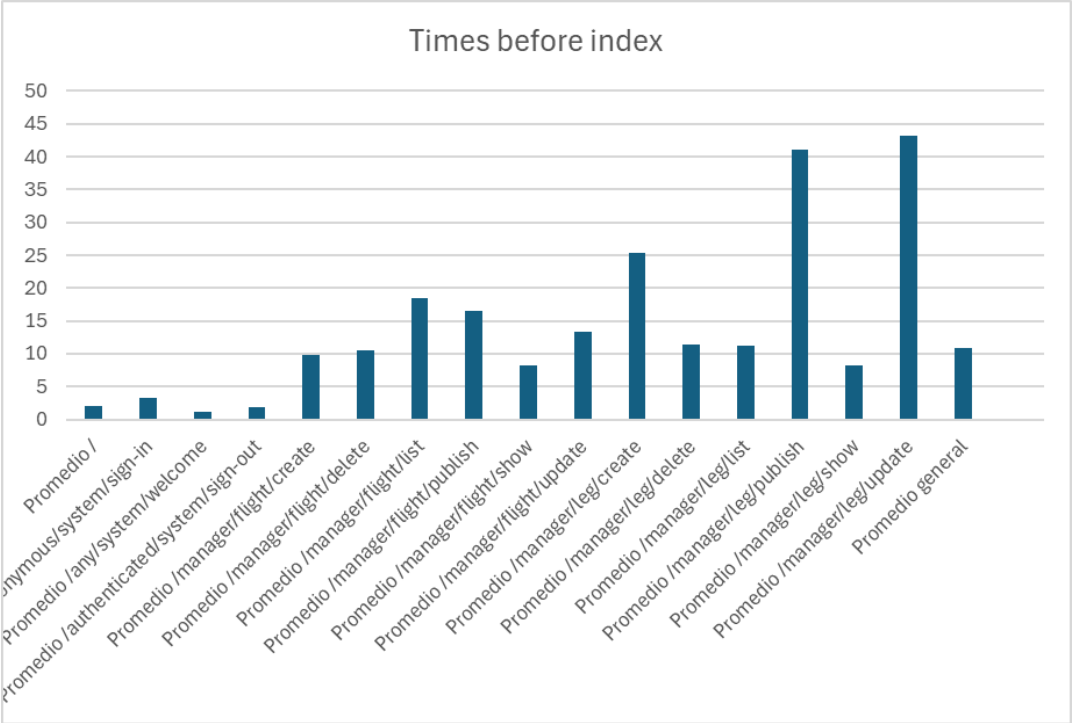
AFTER	
Media	10,62311066
Error típico	0,505371956
Mediana	4,939101
Moda	1,2479
Desviación estándar	15,65837735
Varianza de la muestra	245,1847813
Curtosis	4,91895662
Coeficiente de asimetría	2,396549029
Rango	74,9366
Mínimo	0,6847
Máximo	75,6213
Suma	10198,18623
Cuenta	960
Nivel de confianza(95,0%)	0,99176252

Interval (ms)	9,631348136	11,61487318
Interval (s)	0,009631348	0,011614873

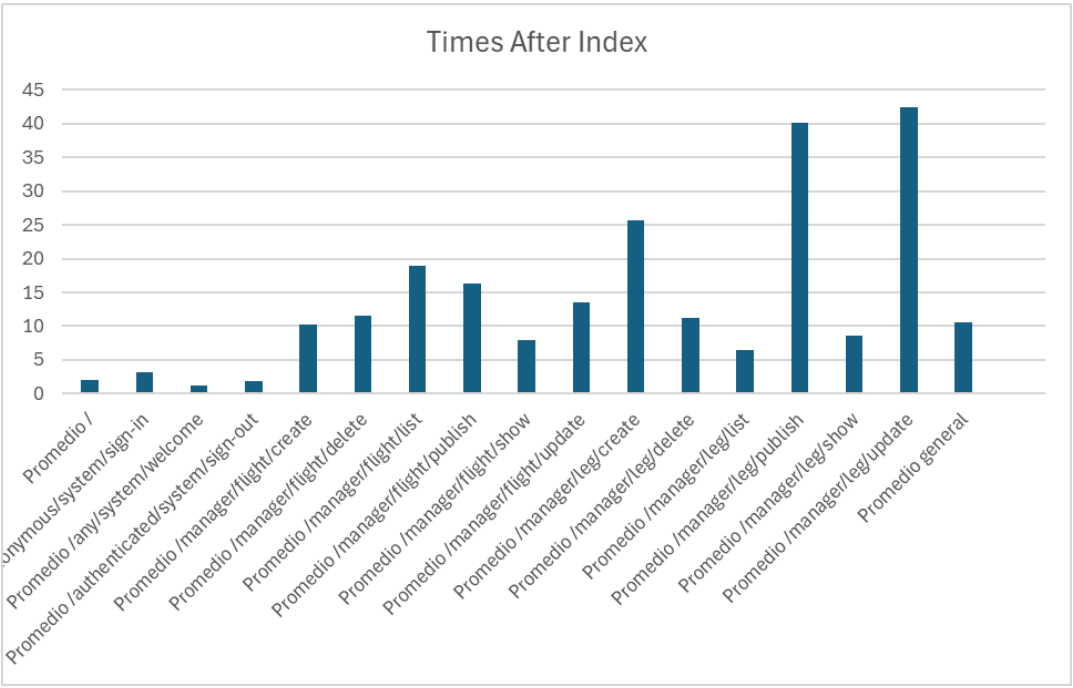
Fijándonos en el intervalo de confianza, antes de realizar los cambios añadiendo los índices, podíamos responder en un tiempo inferior o igual a 11.86473235 ms. Mientras que, tras la adición de los índices optimizando la búsqueda en base de datos, la cota superior pasa a ser 11.61487318 ms, reduciéndose en 0.24985917. Este resultado no tiene significancia, como lo podemos corroborar con el p-valor (0.800019092), que es bastante mayor al nivel de significancia habitual (0.05).

Tras esto, vamos a observar como quedan los gráficos de cada uno, para ver cuales son los MIR que no se han conseguido reducir a pesar de múltiples intentos:

Antes



Después



Se puede comprobar, que los MIR más claros provienen de /manager/leg/update y manager/leg/publish, y aunque update se ha reducido un poco, en general se han quedado al mismo nivel. La única diferencia realmente notable entre el antes y el después es en manager/leg/list, que se ha visto reducida en unos 5 ms. Esto se debe a que es un servicio que hace bastante uso de peticiones a repositorios y, por tanto, de índices en caso de tenerlos.

Visto que los MIR eran bastante notables y que trás la adición de los índices no se consiguió una mejora considerable en ellos, se procedió a analizar la ejecución de pruebas con VisualVM. Los resultados no fueron demasiado concluyentes, pues los peores tiempos de ejecución se generaban en métodos bind, los cuales no tienen código más allá de la habitual línea para cargar los atributos necesarios.

Name	Self Time (CPU)	Total Time (CPU)
acme.features.manager.legs.ManagerLegPublishService.**bind** ()	0,0 ms (- %)	1.075 ms (18,9 %)
acme.features.manager.legs.ManagerLegUpdateService.**bind** ()	0,0 ms (- %)	905 ms (15,9 %)
acme.features.manager.legs.ManagerLegCreateService.**bind** ()	0,0 ms (- %)	694 ms (12,2 %)

Además, todos los tiempos Self Time (CPU) salían a 0.0 ms, lo que indicaba que no era el método en sí, si no lo que se llamaba dentro. Tras esto se procedió a intentar refactorizar los métodos para reducir llamadas y comprobaciones, además se revisaron de nuevo todos los repositorios utilizados, comprobando que no faltasen índices por añadir a las entidades implicadas. Después de realizar una refactorización del código (en adición a los índices), siguiendo los datos proporcionados por VisualVM, el resultado de los test no mejoraba en absoluto, es por ello que se optó por hacer esta comparación solo con la adición de índices. Se pueden observar estas refactorizaciones en la rama t/184

Para terminar con este análisis, vamos a ver una comparativa entre dos ordenadores distintos, comprobando de esta manera la relevancia del hardware a la hora de responder peticiones. La comparativa es únicamente tras la optimización de los índices.
El PC_1 es el mismo utilizado para el reporte, un HP Victus 16-d1001ns con procesador Intel Core i7 12700H y tarjeta gráfica dedicada (GeForce RTX 3050 Ti), y para el PC_2 se ha utilizado el portátil de uno de mis compañeros del grupo, con un Intel Core i7, sin tarjeta gráfica dedicada.

PC_1	
Media	10,62311066
Error típico	0,505371956
Mediana	4,939101
Moda	1,2479
Desviación estándar	15,65837735
Varianza de la muestra	245,1847813
Curtosis	4,91895662
Coeficiente de asimetría	2,396549029
Rango	74,9366
Mínimo	0,6847
Máximo	75,6213
Suma	10198,18623
Cuenta	960
Nivel de confianza(95,0%)	0,99176252

Interval (ms)	9,631348136	11,61487318
Interval (s)	0,009631348	0,011614873
Interval(ms)	19,62001114	24,10188722
Interval(s)	0,019620011	0,024101887

PC_2	
Media	21,86094918
Error típico	1,141913732
Mediana	7,4088005
Moda	1,7404
Desviación estándar	35,38090294
Varianza de la muestra	1251,808293
Curtosis	5,930536395
Coeficiente de asimetría	2,553175638
Rango	189,315301
Mínimo	1,189799
Máximo	190,5051
Suma	20986,51122
Cuenta	960
Nivel de confianza(95,0%)	2,24093804

Se puede observar que, para el PC_2, el intervalo de confianza tiene una cota superior de 24.10188722 ms, lo que es 2.075 veces mayor que la cota superior del PC_1. Esto deja en evidencia que el PC_1 tiene un desempeño mejor, ya que garantiza tiempos de respuestas considerablemente menores a los del PC_2.

Conclusión

En este informe se demuestra que el proceso de testing formal llevado a cabo en el proyecto ha sido completo y con el objetivo de encontrar todos los errores posibles para así corregirlos. Las pruebas realizadas indican que se han probado múltiples escenarios posibles para los requisitos 8 y 9 del Student 1, pudiendo de esta manera reducir la posibilidad de fallo del sistema al mínimo.

Por otra parte, el análisis de rendimiento ha descubierto, mediante un Z-test con significancia estadística del 95%, que las optimizaciones aplicadas para una mayor rapidez en la búsqueda en base de datos, no han sido suficientes para mejorar el tiempo de respuesta del sistema, lo que nos ha llevado a identificar ciertos problemas con algunos métodos, que a pesar de ser revisados exhaustivamente e incluso refactorizados, no se ha tenido éxito en la mejora del rendimiento.

Si lo miramos en conjunto, podemos decir que es un sistema con una calidad funcional bastante robusta, en base a los test realizados, pero con un rendimiento que puede llegar a dar ciertos problemas en algunos dispositivos con sistemas menos avanzados.

De esto sacamos que debemos revisar la eficiencia desde un inicio a la hora de implementar un proyecto, pues a la hora de querer refactorizar puede dar bastantantes problemas y resultar en un sistema demasiado lento.

Bibliografía

Material proporcionado por el profesorado a través de la Enseñanza Virtual.