



Escuela Técnica Superior de
Ingeniería Informática

Testing Report

Realizado por

Grupo: C1.027

Student 3: Manuel Jesús Niza Cobo

Para la asignatura de
Diseño y Pruebas II

Dirigido por
Patricia Jiménez Aguirre

En el departamento de
Lenguajes y Sistemas Informáticos

Convocatoria de junio, curso 2024/25

Índice general

1	Introducción	1
2	Testing funcional	2
2.1.	Pruebas Flight Assignment	2
2.1.1.	Create	2
2.1.2.	List	2
2.1.3.	Show	3
2.1.4.	Delete	4
2.1.5.	Update	4
2.1.6.	Publish	5
2.2.	Pruebas Activity log	5
2.2.1.	Create	5
2.2.2.	List	6
2.2.3.	Show	6
2.2.4.	Delete	7
2.2.5.	Update	7
3	Testing de rendimiento	9
3.1.	Muestra sobre los resultados	9
3.1.1.	Dispositivo 1 (Portatil)	9
3.1.2.	Dispositivo 1 (PC)	12
3.2.	Rendimiento del software con VisualVM	14
3.3.	Rendimiento del dispositivo	17
4	Cobertura	20
5	Creación de índices	24
6	Conclusiones	26

1. Introducción

Este documento tiene como objetivo analizar y evaluar el funcionamiento y rendimiento de los requisitos desarrollados por el Student 3, concretamente los Requisitos 8 y 9. A través de distintas pruebas, tanto funcionales como de rendimiento, se busca garantizar que las funcionalidades implementadas cumplen con los criterios de calidad esperados, tanto en términos de comportamiento correcto como de eficiencia.

En primer lugar, se incluye una sección dedicada al testing funcional, donde se recogen los resultados obtenidos tras ejecutar distintas pruebas sobre las funcionalidades desarrolladas. Estas pruebas verifican que los requisitos se comportan como se espera ante diferentes escenarios de uso, comprobando tanto casos normales como casos límite.

En segundo lugar, se presenta una sección centrada en el testing de rendimiento, donde se estudia la eficiencia del software bajo determinadas condiciones. Para ello, se realizan estadísticas de tiempo de respuesta antes y después de la aplicación de mejoras como el uso de índices en las entidades. Asimismo, se emplean herramientas como VisualVM y el monitor de rendimiento del sistema para identificar cuellos de botella, consumo de recursos y oportunidades de optimización tanto a nivel de software.

La combinación de ambos enfoques permite obtener una visión completa sobre la calidad del software. Este análisis es clave para garantizar que el sistema desarrollado no solo cumple con los requisitos funcionales, sino que también está preparado para ser utilizado de forma fiable en otros dispositivos.

2. Testing funcional

2.1. Pruebas Flight Assignment

2.1.1. Create

Create.safe

- **Funcionalidad:** create.safe
- **Descripción:** Verificar que un flight crew member puede crear asignaciones de vuelo de forma correcta.
- **Resultado esperado:** El sistema deberá permitir crear asignaciones de vuelo válidas.
- **Pruebas:** Se contempla en el test los escenarios donde se intentan rellenar los campos con datos incorrectos y se consigue crear un flight assignment.

Create.hack

- **Funcionalidad:** create.hack
- **Descripción:** Verificar que un rol no flight crew member correspondiente o no autenticado no puede acceder a la creación de asignaciones de vuelo.
- **Resultado esperado:** El sistema debe de lanzar error si un rol no flight crew member o no autenticado intenta acceder a la URL.
- **Prueba:** Se contempla hackear la id del formulario de creación, además del campo leg. Se contempla ingresar desde otro rol.

2.1.2. List

List-Planned.safe

- **Funcionalidad:** list-planned.safe
- **Descripción:** Verificar que un flight crew member puede listar sus asignaciones de vuelo planificadas.
- **Resultado esperado:** El sistema deberá mostrar las asignaciones asociadas al flight crew member.
- **Prueba:** Se comprueba que se accede sin problemas al listado de asignaciones.

List-Planned.hack

- **Funcionalidad:** list-planned.hack

- **Descripción:** Verificar que un rol no flight crew member o no autenticado no puede listar asignaciones de vuelo.
- **Resultado esperado:** El sistema no debe mostrar las asignaciones de vuelo, usando un rol no flight crew member o no autenticado, dando un error.
- **Prueba:** Se intenta acceder a la lista de asignaciones de vuelo desde otro rol.

List-Completed.safe

- **Funcionalidad:** list-completed.safe
- **Descripción:** Verificar que un flight crew member puede listar sus asignaciones de vuelo completadas.
- **Resultado esperado:** El sistema deberá mostrar las asignaciones asociadas al flight crew member.
- **Prueba:** Se comprueba que se accede sin problemas al listado de asignaciones.

List-Completed.hack

- **Funcionalidad:** list.hack
- **Descripción:** Verificar que un rol no flight crew member o no autenticado no puede listar asignaciones de vuelo.
- **Resultado esperado:** El sistema no debe mostrar las asignaciones de vuelo, usando un rol no flight crew member o no autenticado, dando un error.
- **Prueba:** Se intenta acceder a la lista de asignaciones de vuelo desde otro rol.

2.1.3. Show

Show.safe

- **Funcionalidad:** show.safe
- **Descripción:** Verificar que un flight crew member puede ver toda la información de una asignación de vuelo.
- **Resultado esperado:** El sistema deberá mostrar la información de las asignaciones de vuelo asociadas al flight crew member.
- **Prueba:** Se intenta acceder a uno de las asignaciones ya creadas.

Show.hack

- **Funcionalidad:** show.hack
- **Descripción:** Verificar que un rol no flight crew member o no autenticado no puede ver la información de las asignaciones de vuelo.
- **Resultado esperado:** El sistema no debe mostrar la información de asignaciones de vuelo, usando un rol no flight crew member o no autenticado, dando un error de pánico.

- **Prueba:** Se intenta acceder a una asignación de otro usuario con el mismo rol, se intenta acceder además desde otro rol a una asignación cualquiera. Se intenta hackear la id con asignaciones que no existen.

2.1.4. Delete

Delete.safe

- **Funcionalidad:** delete.safe
- **Descripción:** Verificar que un flight crew member puede borrar asignaciones de vuelo de forma correcta.
- **Resultado esperado:** El sistema deberá permitir eliminar asignaciones de vuelo que no estén publicadas.
- **Prueba:** Se intenta eliminar una asignación de vuelo que ya está creada.

Delete.hack

- **Funcionalidad:** delete.hack
- **Descripción:** Verificar que un rol no flight crew member o no autenticado no puede acceder a la eliminación de asignaciones de vuelo.
- **Resultado esperado:** El sistema debe de lanzar error de pánico si un rol no autorizado o no autenticado intenta acceder a la URL.
- **Prueba:** Se intenta eliminar una asignación de vuelo de otra persona con el mismo rol. Se intenta hackear la id para eliminar una que no existe. Se intenta eliminar una asignación desde otro rol.

Delete2.hack

- **Funcionalidad:** delete2.hack
- **Descripción:** Verificar que un rol no flight crew member o no autenticado no puede acceder a la eliminación de asignaciones de vuelo.
- **Resultado esperado:** El sistema debe de lanzar error de pánico si un rol no autorizado o no autenticado intenta acceder a la URL.
- **Prueba:** Se accede a la URL de GET para dar cobertura al unbind de delete.

2.1.5. Update

Update.safe

- **Funcionalidad:** update.safe
- **Descripción:** Verificar que un flight crew member puede actualizar asignaciones de vuelo válidas de forma correcta.
- **Resultado esperado:** El sistema deberá permitir actualizar asignaciones de vuelo válidas.

- **Prueba:** Se modifica alguno de los campos de una asignación que ya existe.

Update.hack

- **Funcionalidad:** update.hack
- **Descripción:** Verificar que un rol no flight crew member, no autenticado u otro flight crew member no puede acceder a la actualización de asignaciones de vuelo.
- **Resultado esperado:** El sistema debe de lanzar error de pánico si un rol no autorizado o no autenticado intenta acceder a la URL.
- **Prueba:** Se intenta hackear el campo leg, y la id de la asignación. Se intenta updatear la de otro usuario. Se intenta acceder al comando desde otro rol.

2.1.6. Publish

Publish.safe

- **Funcionalidad:** publish.safe
- **Descripción:** Verificar que un flight crew member puede publicar proyectos válidos de forma correcta.
- **Resultado esperado:** El sistema deberá permitir publicar asignaciones de vuelo válidas.
- **Prueba:** Se publica uno de las asignaciones ya creadas. Se contemplan todos los casos de uso por los que no se puede publicar una asignación.

Publish.hack

- **Funcionalidad:** publish.hack
- **Descripción:** Verificar que un rol no flight crew member, no autenticado u otro flight crew member no puede acceder a la publicación de asignaciones de vuelo.
- **Resultado esperado:** El sistema debe de lanzar error de pánico si un rol no autorizado o no autenticado intenta acceder a la URL.
- **Prueba:** Se intenta hackear el campo leg y la id. Se intenta publicar una asignación de otro usuario. Se intenta acceder al comando desde otro rol.

2.2. Pruebas Activity log

2.2.1. Create

Create.safe

- **Funcionalidad:** create.safe

- **Descripción:** Comprobar que un flight crew member puede crear asignación de vuelo de forma correcta.
- **Resultado esperado:** El sistema deberá permitir crear asignaciones de vuelo válidas.
- **Prueba:** Se crea un registro de actividad de una asignación ya publicada.

Create.hack

- **Funcionalidad:** create.hack
- **Descripción:** Comprobar que un rol no autorizado o no autenticado no puede acceder a la creación de asignaciones de vuelo.
- **Resultado esperado:** El sistema debe bloquear el acceso y lanzar un error cuando un rol no autorizado o no autenticado intenta crear una asignación.
- **Prueba:** Se intenta crear un registro de actividad con otro id diferente. Se intenta hackear el campo del assignment.

2.2.2. List

List.safe

- **Funcionalidad:** list.safe
- **Descripción:** Comprobar que un flight crew member puede listar sus asignaciones de vuelo.
- **Resultado esperado:** El sistema debe mostrar las asignaciones asociadas al flight crew member.
- **Prueba:** Se muestra el listado de registros de actividad ya existentes.

List.hack

- **Funcionalidad:** list.hack
- **Descripción:** Comprobar que un rol no autorizado o no autenticado no puede listar asignaciones.
- **Resultado esperado:** El sistema debe bloquear la solicitud y devolver un error.
- **Prueba:** Se intenta acceder al listado de otra asignación de otro usuario. Se intenta acceder desde otro rol. Se intenta hackear la id.

2.2.3. Show

Show.safe

- **Funcionalidad:** show.safe
- **Descripción:** Comprobar que un flight crew member puede ver la información de una asignación.

- **Resultado esperado:** El sistema debe mostrar la información completa de la asignación asociada.
- **Prueba:** Se intenta acceder a un registro de actividad ya creada.

Show.hack

- **Funcionalidad:** show.hack
- **Descripción:** Comprobar que un rol no autorizado o no autenticado no puede ver asignaciones.
- **Resultado esperado:** El sistema debe bloquear el acceso y devolver error.
- **Prueba:** Se intenta acceder a un registro de actividad a través de la id. Se intenta acceder a través de otro rol.

2.2.4. Delete

Delete.safe

- **Funcionalidad:** delete.safe
- **Descripción:** Comprobar que un flight crew member puede eliminar asignaciones válidas.
- **Resultado esperado:** El sistema debe permitir eliminar asignaciones que cumplan criterios.
- **Prueba:** Se elimina un registro de actividad ya creado.

Delete.hack

- **Funcionalidad:** delete.hack
- **Descripción:** Comprobar que un rol no autorizado o no autenticado no puede eliminar asignaciones.
- **Resultado esperado:** El sistema debe bloquear la eliminación y devolver error.
- **Prueba:** Se intenta hackear un registro de actividad de otro usuario. Se intenta acceder al comando desde otro rol.

2.2.5. Update

Update.safe

- **Funcionalidad:** update.safe
- **Descripción:** Comprobar que un flight crew member puede actualizar asignaciones válidas.
- **Resultado esperado:** El sistema debe permitir actualizar asignaciones correctamente.
- **Prueba:** Se actualiza uno de los campos de un registro ya existente.

Update.hack

- **Funcionalidad:** update.hack
- **Descripción:** Comprobar que un rol no autorizado o no autenticado no puede actualizar asignaciones.
- **Resultado esperado:** El sistema debe bloquear la actualización y devolver error.
- **Prueba:** Se intenta hackear el registro de otro usuario. Se intenta acceder al comando desde otro rol.

Publish.safe

- **Funcionalidad:** publish.safe
- **Descripción:** Publicar un registro de actividad.
- **Resultado esperado:** El sistema debe permitir actualizar asignaciones correctamente.
- **Prueba:** Se publica un registro de actividad ya existente.

Publish.hack

- **Funcionalidad:** publish.hack
- **Descripción:** Comprobar que un rol o un usuario no está autorizado para publicar una asignación.
- **Resultado esperado:** El sistema debe bloquear la publicación y devolver error.
- **Prueba:** Se intenta publicar un registro de actividad de otro usuario. Se intenta realizar la acción desde otro rol.

3. Testing de rendimiento

Aquí presentamos los resultados obtenidos tras la ejecución de los tests funcionales, elaborados para los requisitos 8 y 9 del Student 3. El objetivo principal es evaluar el impacto de las pruebas sobre los tiempos de respuesta, evaluar la ejecución del software y el rendimiento del hardware en cada uno de los diferentes dispositivos. Para ello, se han realizado varias pruebas, una inicial sin índices y otra posterior con los índices optimizados aplicados, midiendo los tiempos de respuesta medios en dos dispositivos distintos: el portátil de trabajo y el PC personal.

3.1. Muestra sobre los resultados

Los resultados obtenidos mediante el análisis de la traza del log al ejecutar el launcher replay. Durante este proceso, se han registrado los tiempos de respuesta medios de las distintas features, permitiendo generar estadísticas.

3.1.1. Dispositivo 1 (Portatil)

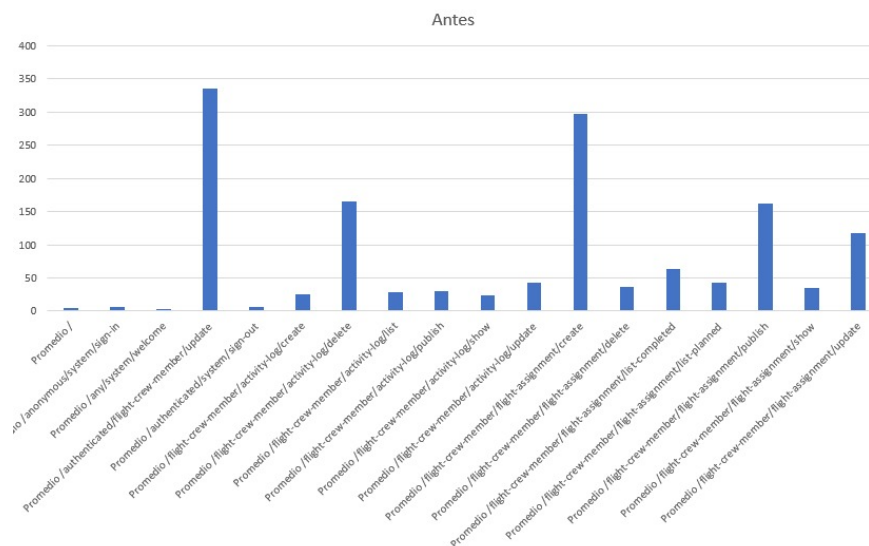


Figura 3.1: Gráfica de las pruebas en el dispositivo 1.

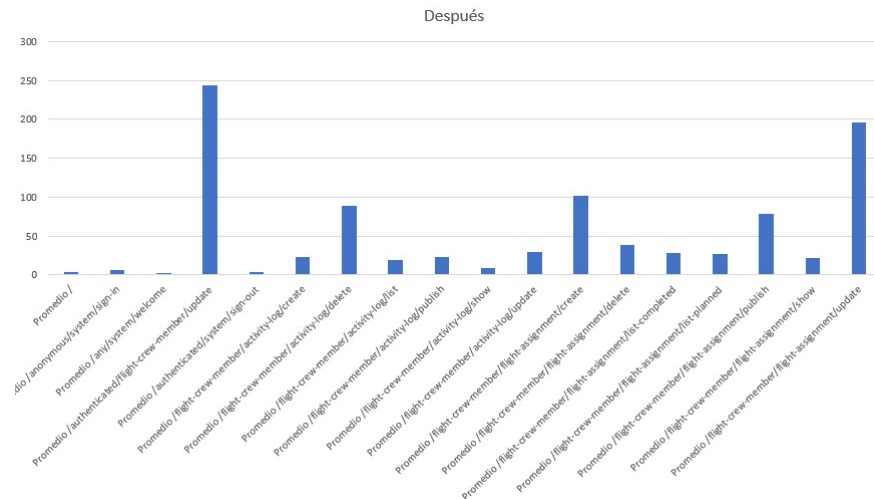


Figura 3.2: Gráfica de las pruebas en el dispositivo 1.

	Antes		Después	
Media	29.3305205		Media	23.0747917
Error típico	3.32676327		Error típico	3.19315307
Mediana	7.6881		Mediana	5.027099
Moda	#N/D		Moda	1.8713
Desviación es	94.7397697		Desviación es	90.9348104
Varianza de la	8975.62397		Varianza de la	8269.13974
Curtosis	58.2945816		Curtosis	75.2161943
Coefficiente de	7.21926414		Coefficiente de	8.36111462
Rango	1050.2571		Rango	1049.742
Mínimo	1.430701		Mínimo	1.3457
Máximo	1051.6878		Máximo	1051.0877
Suma	23787.0521		Suma	18713.6561
Cuenta	811		Cuenta	811
Nivel de confi	6.53009371		Nivel de confi	6.26783066
Intervalo(ms)	22.8004268		Intervalo(ms)	16.8069611
	35.8606142			29.3426224
Intervalo(s)	0.02280043		Intervalo(s)	0.01680696
	0.03586061			0.02934262

Figura 3.3: Estadísticas de las pruebas en el dispositivo 1.

	Prueba z para medias de dos muestras		
		<i>Before</i>	<i>After</i>
Media	29.3305205	23.0747917	
Varianza (conocida)	8975.62397	8269.13974	
Observaciones	811	811	
Diferencia hipotética	0		
z	1.35662468		
P(Z<=z) una cola	0.08745025		
Valor crítico de z (una	1.64485363		
Valor crítico de z (dos	0.1749005		
Valor crítico de z (dos	1.95996398		

Figura 3.4: Estadísticas de las pruebas en el dispositivo 1.

A partir de los datos obtenidos, los tiempos de respuesta medios mostraron una reducción, pasando de 29,33 ms antes de la optimización a 23,07 ms después de la implementación de índices. Sin embargo, dado que las desviaciones estándar en ambos casos son superiores a 90 ms, la diferencia no resultó estadísticamente significativa al 95 % de confianza.

El valor estadístico p - value de dos colas es 0,1749 lo cual refuerza la afirmación de que no hay evidencia suficiente para afirmar que los tiempos de respuesta mejoraron de forma significativa, ya que este valor se encuentra por encima del umbral de alfa de 0,05 en el dispositivo 1.

Los intervalos de confianza al 95 % se superponen:

- Antes del cambio: [22,80 ms, 35,86 ms]
- Después del cambio: [16,81 ms, 29,34 ms]

3.1.2. Dispositivo 1 (PC)

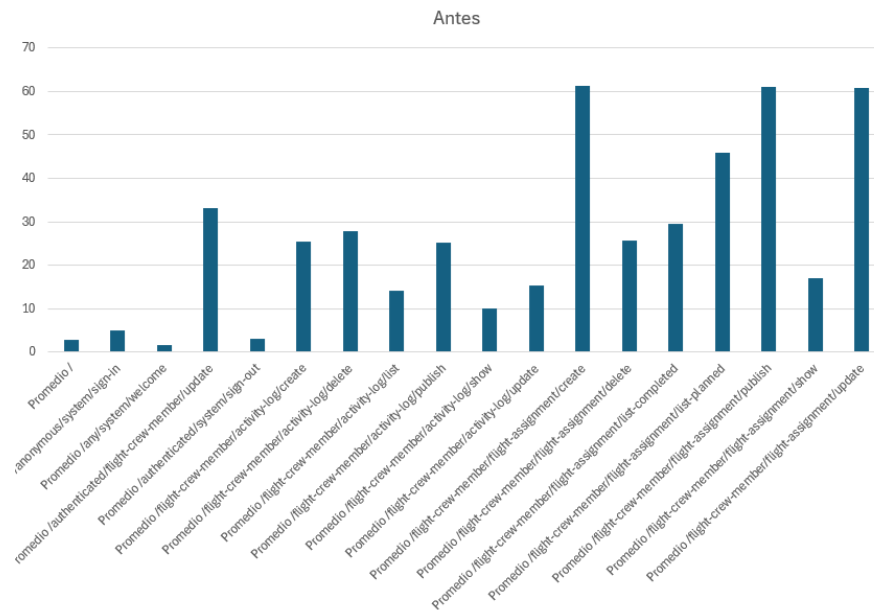


Figura 3.5: Gráfica de las pruebas en el dispositivo 2.

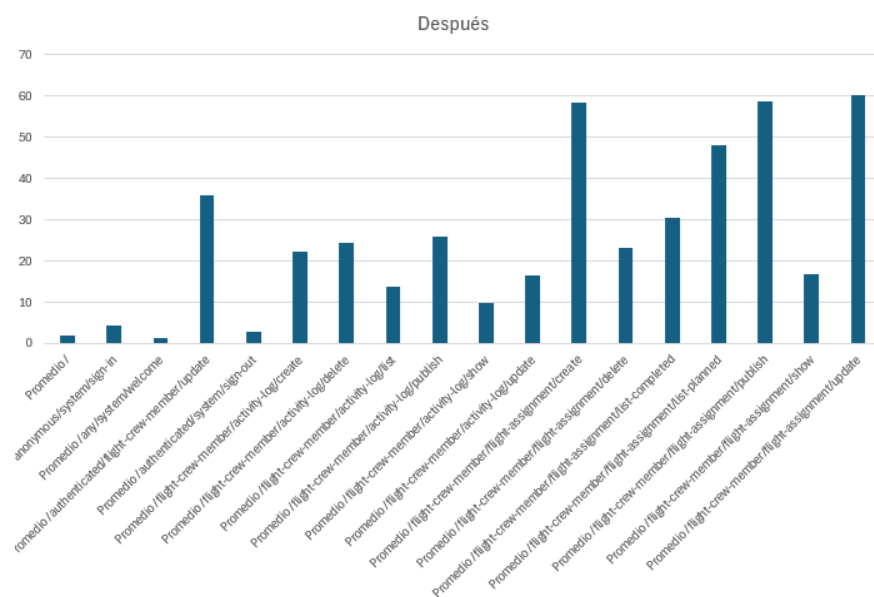


Figura 3.6: Gráfica de las pruebas en el dispositivo 2.

<i>Antes</i>		<i>Después</i>	
Media	13.4180317	Media	13.0284958
Error típico	0.71035506	Error típico	0.70321138
Mediana	4.6504	Mediana	3.5485
Moda	1.2834	Moda	0.951001
Desviación es	20.2295352	Desviación es	20.0260971
Varianza de la	409.234094	Varianza de la	401.044564
Curtosis	9.23285164	Curtosis	7.11101957
Coeficiente de	2.81970033	Coeficiente de	2.58184215
Rango	130.4515	Rango	113.244
Mínimo	1.051	Mínimo	0.7907
Máximo	131.5025	Máximo	114.0347
Suma	10882.0237	Suma	10566.1101
Cuenta	811	Cuenta	811
Nivel de confi	1.39435383	Nivel de confi	1.38033152
Intervalo(ms)		Intervalo(ms)	
Intervalo(s)		Intervalo(s)	

Figura 3.7: Estadísticas de las pruebas en el dispositivo 2.

Prueba z para medias de dos muestras			
	<i>Before</i>	<i>After</i>	
Media	13.4180317	13.0284958	
Varianza (con	409.234094	401.044564	
Observacione	811	811	
Diferencia hip	0		
z	0.38970921		
P(Z<=z) una co	0.34837579		
Valor crítico d	1.64485363		
Valor crítico d	0.69675158		
Valor crítico d	1.95996398		

Figura 3.8: Estadísticas de las pruebas en el dispositivo 2.

En cambio en el dispositivo 2 se puede ver que los tiempos de respuesta medios mostraron una reducción drástica respecto a los del dispositivo 1. Respecto al caso inicial y el caso final con índices pasando de 13,41 ms antes de la optimización a 13,08 ms después de la implementación de índices. Sin embargo, dado que las desviaciones estándar en ambos casos son superiores a 400 ms, la diferencia no resultó estadísticamente significativa al 95 % de confianza.

El valor estadístico p - value de dos colas es 0,696 lo cual confirma que casi no existió mejora desde la versión inicial a la final con índices, ya que este valor se encuentra por encima del umbral de alfa de 0,05 en el dispositivo 2.

Los intervalos de confianza al 95 % se superponen:

- Antes del cambio: [12,02 ms, 14,81 ms]
- Después del cambio: [11,64 ms, 14,4 ms]

Las conclusiones entonces que podemos sacar es que en ninguno de los dos dispositivos existe una diferencia sustancial de mejora de la versión anterior a la posterior con índices. En el dispositivo 1 si es algo más notable pero sigue estando por encima del umbral de alfa de 0,05. Respecto a los tiempos de respuesta, el dispositivo 2 muestra una mejora que afecta a los intervalos de confianza.

3.2. Rendimiento del software con VisualVM

Para realizar esta prueba se ha utilizado VisualVM para identificar los métodos y clases que consumen más recursos durante la ejecución, permitiendo detectar posibles cuellos de botella que puedan ser optimizados mediante refactorización o mejoras de índices.

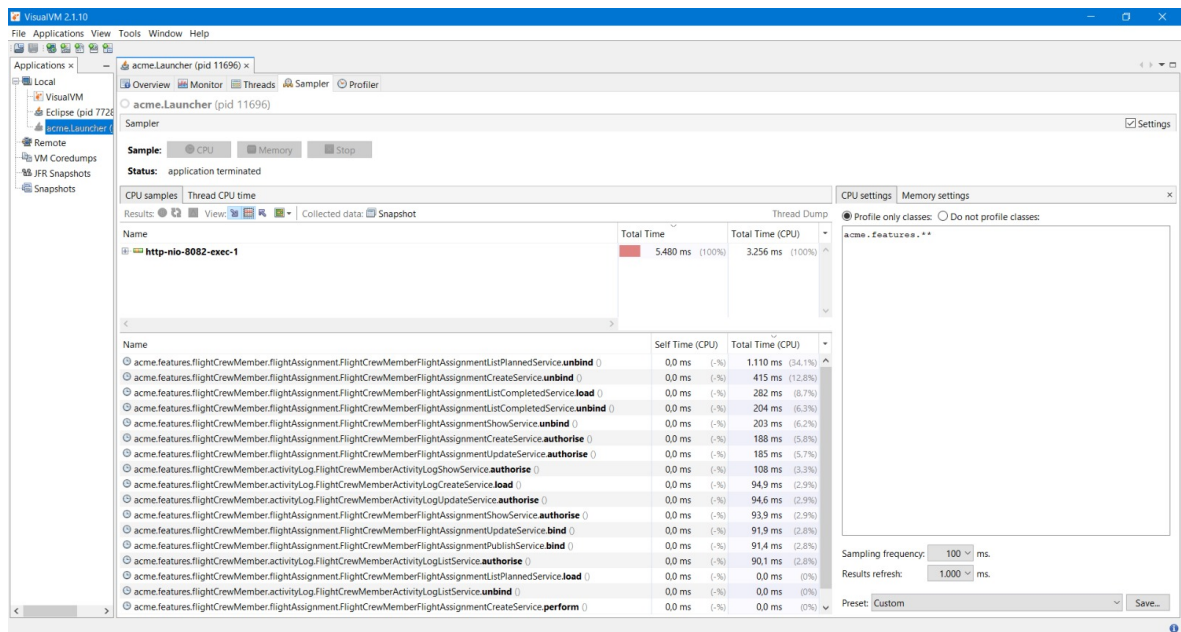


Figura 3.9: Rendimiento de la ejecución dispositivo 1.

Name	Self Time (CPU)	Total Time (CPU)
acme.features.flightCrewMember.flightAssignment.FlightCrewMemberFlightAssignmentListPlannedService.unbind ()	0,0 ms (-%)	1.110 ms (34,1%)
acme.features.flightCrewMember.flightAssignment.FlightCrewMemberFlightAssignmentCreateService.unbind ()	0,0 ms (-%)	415 ms (12,8%)
acme.features.flightCrewMember.flightAssignment.FlightCrewMemberFlightAssignmentListCompletedService.load ()	0,0 ms (-%)	282 ms (8,7%)
acme.features.flightCrewMember.flightAssignment.FlightCrewMemberFlightAssignmentListCompletedService.unbind ()	0,0 ms (-%)	204 ms (6,3%)
acme.features.flightCrewMember.flightAssignment.FlightCrewMemberFlightAssignmentShowService.unbind ()	0,0 ms (-%)	203 ms (6,2%)
acme.features.flightCrewMember.flightAssignment.FlightCrewMemberFlightAssignmentCreateService.authorise ()	0,0 ms (-%)	188 ms (5,8%)
acme.features.flightCrewMember.flightAssignment.FlightCrewMemberFlightAssignmentUpdateService.authorise ()	0,0 ms (-%)	185 ms (5,7%)
acme.features.flightCrewMember.activityLog.FlightCrewMemberActivityLogShowService.authorise ()	0,0 ms (-%)	108 ms (3,3%)
acme.features.flightCrewMember.activityLog.FlightCrewMemberActivityLogCreateService.load ()	0,0 ms (-%)	94,9 ms (2,9%)
acme.features.flightCrewMember.activityLog.FlightCrewMemberActivityLogUpdateService.authorise ()	0,0 ms (-%)	94,6 ms (2,9%)
acme.features.flightCrewMember.flightAssignment.FlightCrewMemberFlightAssignmentShowService.authorise ()	0,0 ms (-%)	93,9 ms (2,9%)
acme.features.flightCrewMember.flightAssignment.FlightCrewMemberFlightAssignmentUpdateService.bind ()	0,0 ms (-%)	91,9 ms (2,8%)
acme.features.flightCrewMember.flightAssignment.FlightCrewMemberFlightAssignmentPublishService.bind ()	0,0 ms (-%)	91,4 ms (2,8%)
acme.features.flightCrewMember.activityLog.FlightCrewMemberActivityLogListService.authorise ()	0,0 ms (-%)	90,1 ms (2,8%)
acme.features.flightCrewMember.flightAssignment.FlightCrewMemberFlightAssignmentListPlannedService.load ()	0,0 ms (-%)	0,0 ms (0%)
acme.features.flightCrewMember.activityLog.FlightCrewMemberActivityLogListService.unbind ()	0,0 ms (-%)	0,0 ms (0%)
acme.features.flightCrewMember.flightAssignment.FlightCrewMemberFlightAssignmentCreateService.perform ()	0,0 ms (-%)	0,0 ms (0%)
acme.features.flightCrewMember.flightAssignment.FlightCrewMemberFlightAssignmentCreateService.bind ()	0,0 ms (-%)	0,0 ms (0%)

Figura 3.10: Rendimiento de la ejecución dispositivo 1.

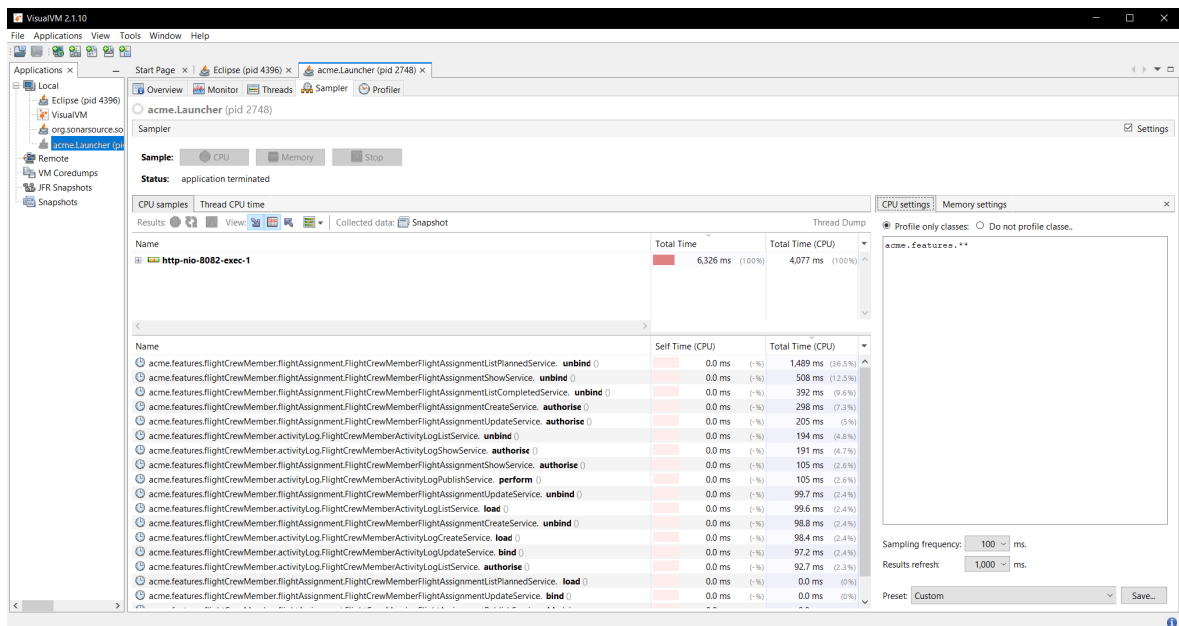


Figura 3.11: Rendimiento de la ejecución dispositivo 2.

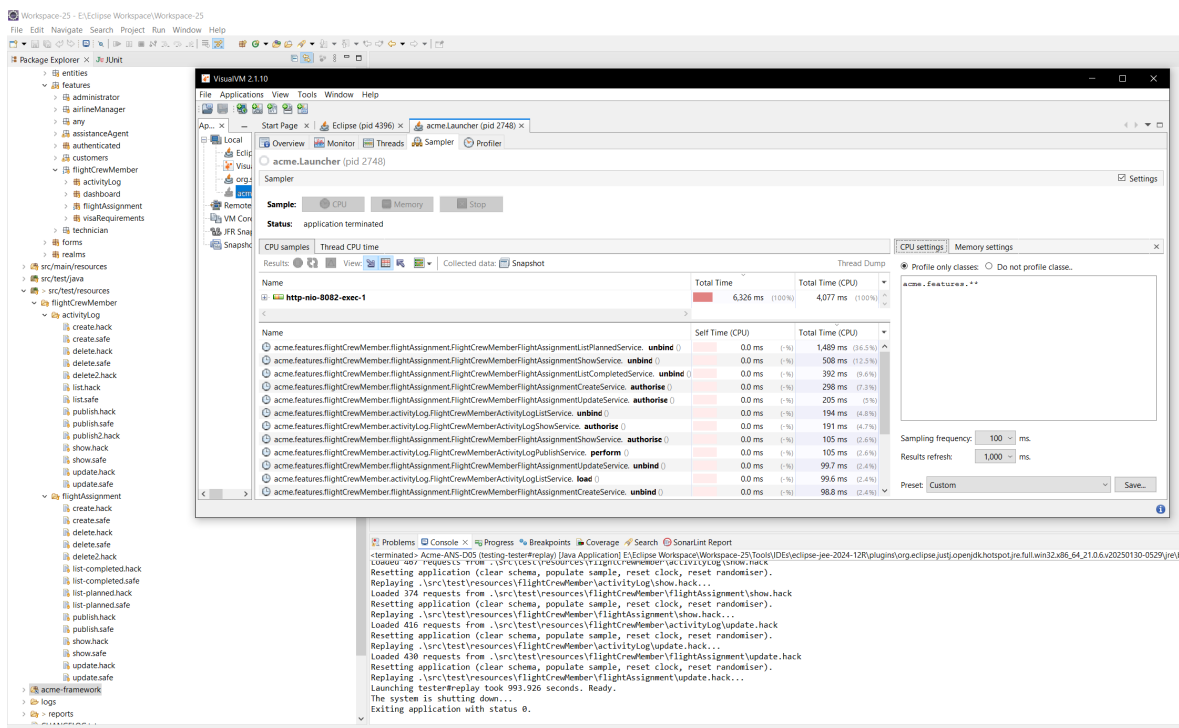


Figura 3.12: Rendimiento de la ejecución dispositivo 2.

Se puede observar un análisis detallado del consumo de recursos a nivel de clases y métodos durante la ejecución del proyecto. Permite visualizar qué partes del código suponen una mayor carga de trabajo, lo que facilita la identificación de MIR. Estos puntos críticos del sistema entre los que se encuentran el unbind de ListPlanned (Flight Assignment), unbind de Create (Flight Assignment) y load/unbind de ListComplete (Flight Assignment) como los que más afectan

negativamente. Estos métodos se pueden mejorar a través de refactorizaciones o mejoras en las consultas a la base de datos gracias a índices, todo ello para optimizar el rendimiento de la aplicación.

3.3. Rendimiento del dispositivo

Finalmente, se monitorizó el rendimiento general del sistema, en concreto del hardware, durante la ejecución de las pruebas, lo cual proporciona una visión más completa del rendimiento y la carga del sistema sobre los dos dispositivos.

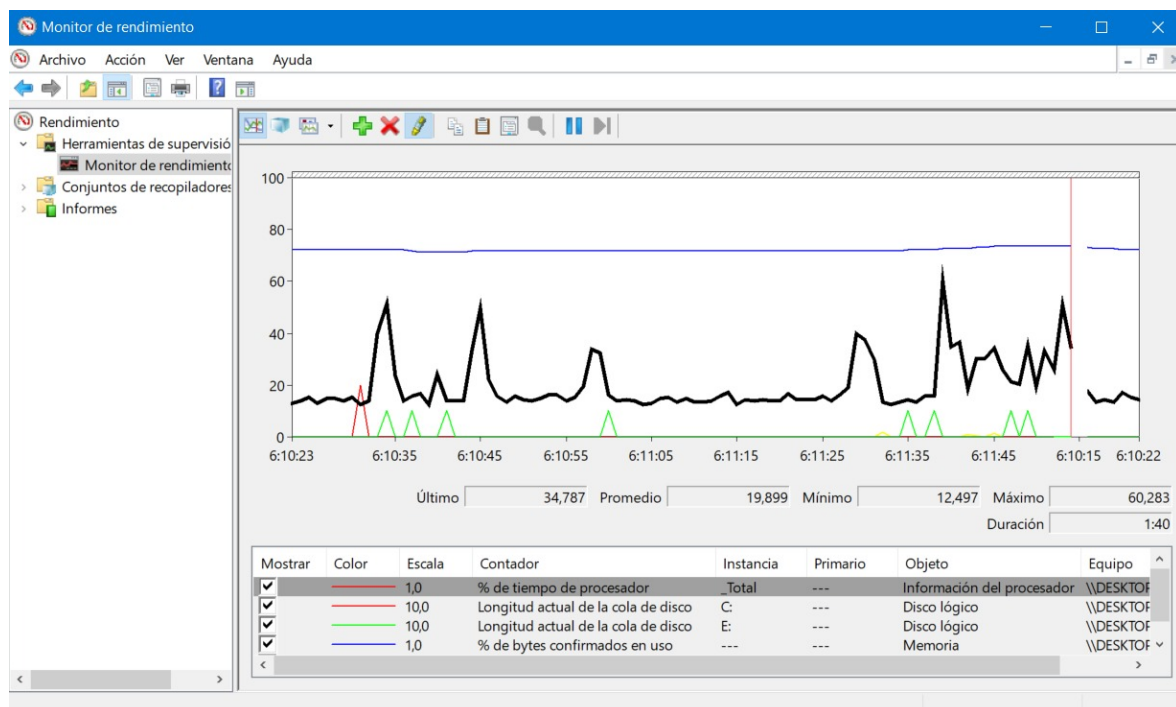


Figura 3.13: Rendimiento del hardware dispositivo 1.

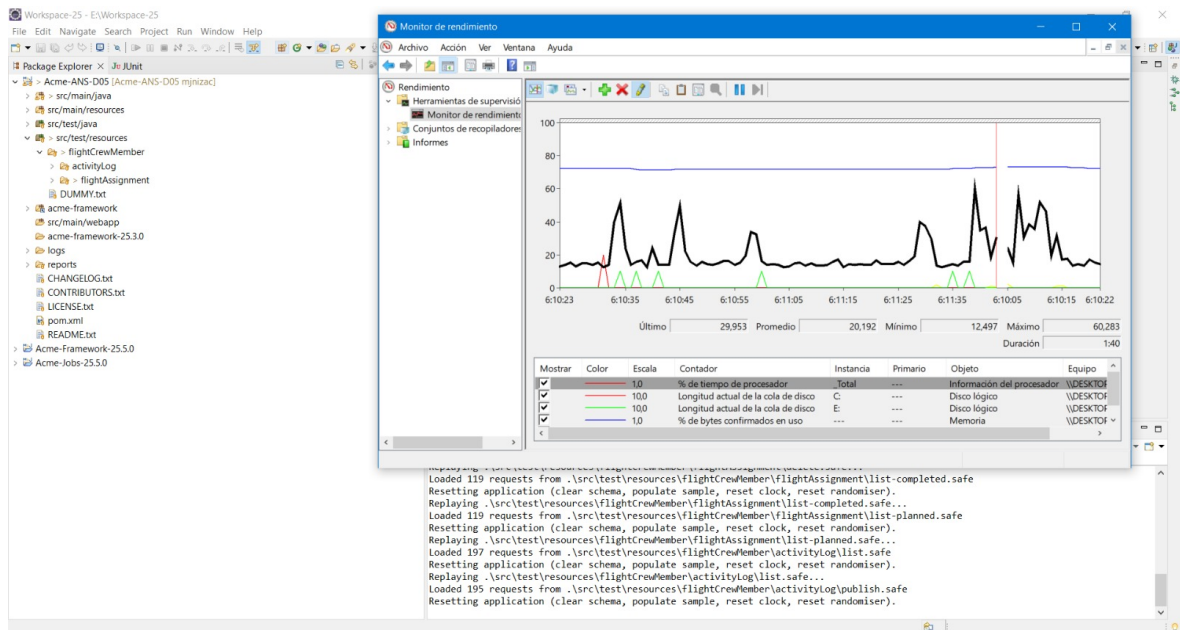


Figura 3.14: Rendimiento del hardware dispositivo 1.

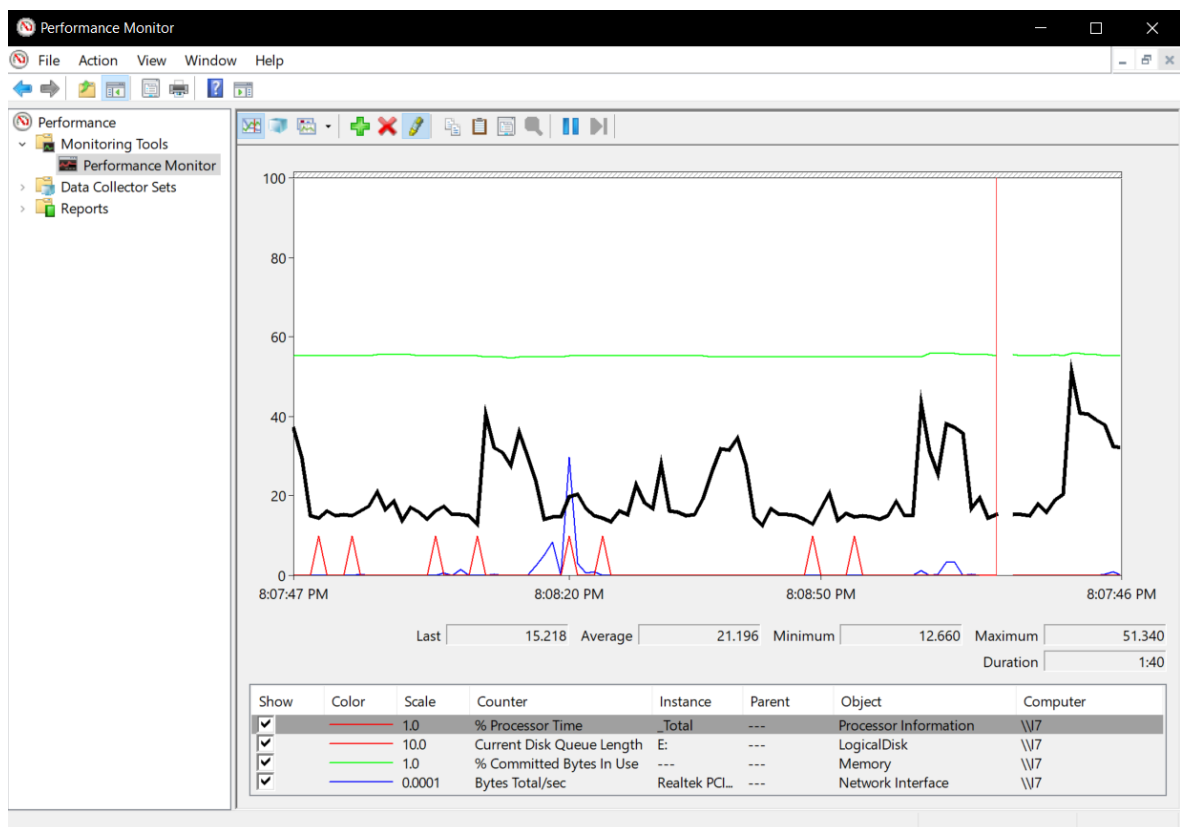


Figura 3.15: Rendimiento del hardware dispositivo 2.

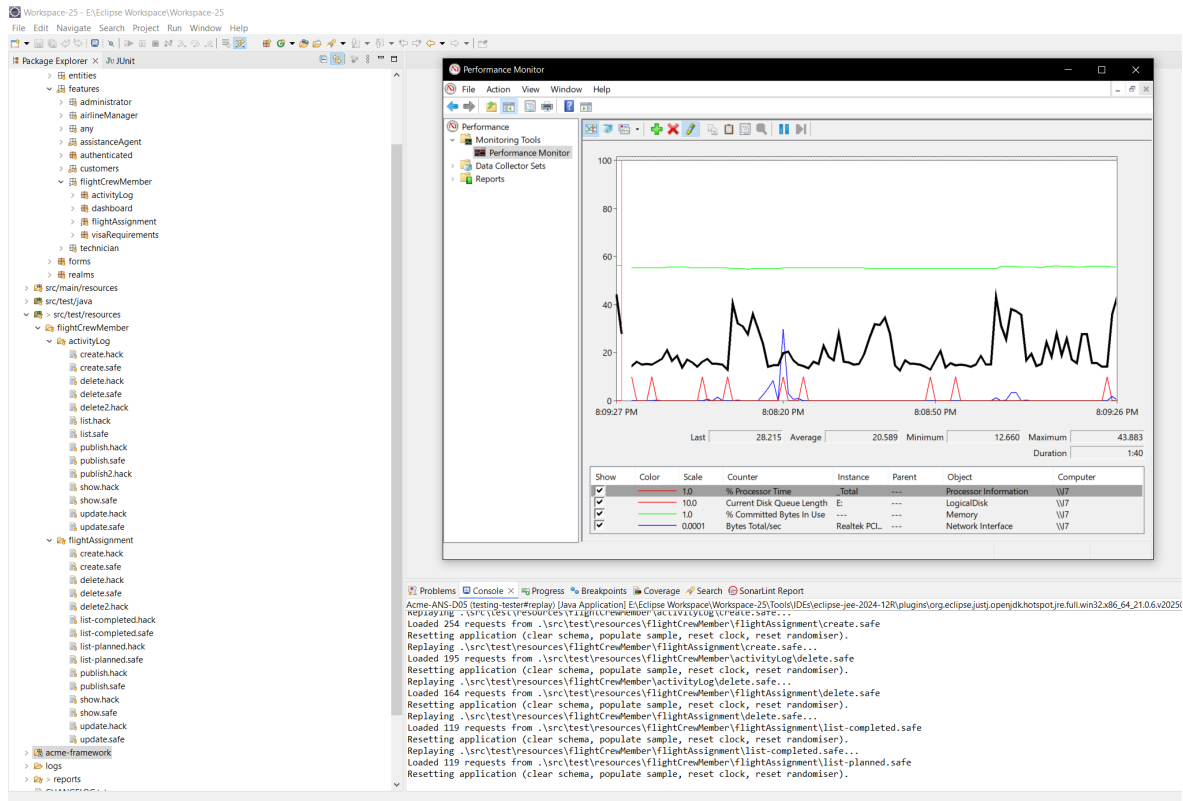


Figura 3.16: Rendimiento del hardware dispositivo 2.

Como se puede observar, el rendimiento sobre el dispositivo 2 es similar al del dispositivo 1 durante la ejecución de las pruebas funcionales del sistema. Ambos dispositivos no tienen dificultad para afrontar las pruebas, ya que no existe cuello de botella en su hardware.

A partir de las distintas pruebas realizadas, se puede concluir que el dispositivo 2 (PC) presenta un rendimiento sustancialmente superior en comparación con el dispositivo 1 (portátil). Las métricas obtenidas además muestran que los tiempos de respuesta son más reducidos, y presenta una mayor estabilidad durante la ejecución de las pruebas funcionales. Esto evidencia que el PC, al contar con mejores prestaciones a nivel de hardware, responde mucho mejor a la carga de trabajo de las pruebas funcionales, lo que lo convierte en una opción más adecuada para ejecutar este tipo de pruebas.

4. Cobertura

Analizaremos la cobertura del código que hemos podido lograr gracias a los tests funcionales realizados. La cobertura de código indica qué porcentaje del código fuente ha sido ejecutado durante las pruebas, esto permite identificar las áreas que han sido verificadas y aquellas que aún no se han podido analizar (las cuales es posible que contengan errores aún no detectados):

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
acme.features.flightCrewMember.flightAssignment	99,9 %	1.447	2	1.449
> FlightCrewMemberFlightAssignmentPublishService.java	99,4 %	338	2	340
> FlightCrewMemberFlightAssignmentController.java	100,0 %	42	0	42
> FlightCrewMemberFlightAssignmentCreateService.java	100,0 %	239	0	239
> FlightCrewMemberFlightAssignmentDeleteService.java	100,0 %	198	0	198
> FlightCrewMemberFlightAssignmentListCompletedService.java	100,0 %	104	0	104
> FlightCrewMemberFlightAssignmentListPlannedService.java	100,0 %	104	0	104
> FlightCrewMemberFlightAssignmentShowService.java	100,0 %	183	0	183
> FlightCrewMemberFlightAssignmentUpdateService.java	100,0 %	239	0	239
acme.features.flightCrewMember.activityLog	99,9 %	1.048	1	1.049
> FlightCrewMemberActivityLogCreateService.java	99,6 %	245	1	246
> FlightCrewMemberActivityLogController.java	100,0 %	35	0	35
> FlightCrewMemberActivityLogDeleteService.java	100,0 %	146	0	146
> FlightCrewMemberActivityLogListService.java	100,0 %	163	0	163
> FlightCrewMemberActivityLogPublishService.java	100,0 %	144	0	144
> FlightCrewMemberActivityLogShowService.java	100,0 %	145	0	145
> FlightCrewMemberActivityLogUpdateService.java	100,0 %	170	0	170

Figura 4.1: Cobertura de las funcionalidades.

Con respecto a las funcionalidades de ambas entidades (FlightAssignment y ActivityLog) podemos observar que alcanzan el 99,9 % de cobertura, esto implica que gran parte del código implementado para realizar estas funcionalidades es probado por los tests en caso de que exista algún error no contemplado.

Para justificar el 0,01 % restante en cada clase explicaré qué conceptos no se han analizado o tenido en cuenta:

```

FlightCrewMemberFlightAssignmentPublishService.java
26 private FlightCrewMemberFlightAssignmentRepository repository;
27
28 // AbstractGuiService interface -----
29
30
31 @Override
32 public void authorise() {
33     boolean isAuthorised = false;
34
35     if (super.getRequest().getPrincipal().hasRealmOfType(FlightCrewMember.class))
36         // Only is allowed to publish a flight assignment if the creator is associated.
37         // A flight assignment cannot be published if the assignment is in published mode and not in draft mode.
38         if (super.getRequest().getMethod().equals("POST") && super.getRequest().getData("id", Integer.class) != null) {
39             Integer flightAssignmentId = super.getRequest().getData("id", Integer.class);
40
41             if (flightAssignmentId != null) {
42                 FlightAssignment flightAssignment = this.repository.findFlightAssignmentById(flightAssignmentId);
43                 FlightCrewMember flightCrewMember = (FlightCrewMember) super.getRequest().getPrincipal().getActiveRealm();
44
45                 // Only is allowed to publish a flight assignment if the leg selected is between the options shown.
46                 Collection<Leg> legs = this.repository.findAllLegsByAirlineId(flightCrewMember.getAirline().getId());
47                 Leg legSelected = super.getRequest().getData("leg", Leg.class);
48
49                 isAuthorised = flightAssignment != null && flightAssignment.getDraftMode() && flightAssignment.getFlightCrewMember().equals(flightCrewMember);
50             }
51         }
52     }
53 }
54
55
56
57
58

```

Figura 4.2: Cobertura de la funcionalidad publish (Flight Assignment).

En el método `authorise` se contempló realizar pruebas con dos roles, siendo uno de ellos el adecuado para estar autorizado por el sistema. Aunque Eclipse no consiguió identificar esta prueba, en el log de la prueba aparece la petición. Además, en caso de que el 'flightAssignment' llegase a ser nulo en la línea 53 del código, no se podría contemplar el resto de escenarios donde se necesitan sus métodos.

```


FlightCrewMemberFlightAssignmentPublishService.java
70 @Override
71 public void bind(final FlightAssignment flightAssignment) {
72     super.bindObject(flightAssignment, "duty", "currentStatus", "remarks", "leg");
73 }
74
75 @Override
76 public void validate(final FlightAssignment flightAssignment) {
77     if (flightAssignment.getFlightCrewMember() != null) {
78         // Only flight crew members with an "AVAILABLE" status can be assigned
79         boolean isAvailable = flightAssignment.getFlightCrewMember().getAvailabilityStatus().equals(AvailabilityStatus.AVAILABLE);
80         super.state(isAvailable, "flightCrewMember", "acme.validation.flightAssignment.flightCrewMember.available");
81
82         // Cannot be assigned to multiple legs simultaneously
83         boolean isAlreadyAssigned = this.repository.hasFlightCrewMemberLegAssociated(flightAssignment.getFlightCrewMember().getId(), flightAssignment.getLeg().getId());
84         super.state(!isAlreadyAssigned, "flightCrewMember", "acme.validation.flightAssignment.flightCrewMember.multipleLegs");
85     }
86
87     // To publish a flight assignment these cannot be linked to legs that are not published, do not belongs to the flight crew member a
88     if (flightAssignment.getLeg() != null) {
89         boolean isDraftModeLeg = flightAssignment.getLeg().isDraftMode();
90         super.state(!isDraftModeLeg, "leg", "acme.validation.flightAssignment.leg.draftmode");
91
92         boolean isLinkedToPastLeg = flightAssignment.getLeg().getScheduledDeparture().before(MomentHelper.getCurrentMoment());
93         super.state(!isLinkedToPastLeg, "leg", "acme.validation.flightAssignment.leg.moment");
94     }
95
96     // Each leg can only have one pilot and one co-pilot
97     if (flightAssignment.getDuty() != null && flightAssignment.getLeg() != null) {
98         boolean isDutyAlreadyAssigned = this.repository.hasDutyAssigned(flightAssignment.getLeg().getId(), flightAssignment.getDuty().getId());
99         super.state(!isDutyAlreadyAssigned, "duty", "acme.validation.flightAssignment.duty");
100     }
101 }
102
103

```

Figura 4.3: Cobertura de la funcionalidad publish 2 (Flight Assignment).

En las validaciones se contemplaron todos los casos de usos por lo que se permite publicar un flight assignment, pero no se pudo comprobar a publicar un flight

assignment ligado a una leg que no estuviese publicada, ya que el dataset de legs no contempla en su mayoría legs no publicadas que tengan relación con las aerolinea disponible de cada crew member.



```
25  @Override
26  public void authorise() {
27
28      boolean isAuthorised = false;
29
30      if (super.getRequest().getPrincipal().hasRealmOfType(FlightCrewMember.class)) {
31
32          if (super.getRequest().getMethod().equals("GET") && super.getRequest().getData("assignmentId", Integer.class) != null) {
33
34              Integer assignmentId = super.getRequest().getData("assignmentId", Integer.class);
35              FlightAssignment flightAssignment = this.repository.findFlightAssignmentById(assignmentId);
36
37              if (flightAssignment != null) {
38
39                  FlightCrewMember flightCrewMember = (FlightCrewMember) super.getRequest().getPrincipal().getActiveRealm();
40
41                  isAuthorised = !flightAssignment.getDraftMode() && flightAssignment.getLeg().getScheduledArrival().before(MomentHelper.ge
42              }
43          }
44      }
45
46      // Only is allowed to create an activity log if the creator is the flight crew member associated to the flight assignment.
47      // An activity log cannot be created if the assignment is planned, only complete are allowed.
48      if (super.getRequest().getMethod().equals("POST") && super.getRequest().getData("assignmentId", Integer.class) != null && super.g
49
50          Integer assignmentId = super.getRequest().getData("assignmentId", Integer.class);
51          FlightAssignment flightAssignment = this.repository.findFlightAssignmentById(assignmentId);
52
53          if (flightAssignment != null && super.getRequest().getData("id", Integer.class).equals(0)) {
54
55              FlightCrewMember flightCrewMember = (FlightCrewMember) super.getRequest().getPrincipal().getActiveRealm();
56
57              isAuthorised = !flightAssignment.getDraftMode() && flightAssignment.getLeg().getScheduledArrival().before(MomentHelper.ge
58          }
59
60      }
```

Figura 4.4: Cobertura de la funcionalidad create (Activity Log).

Como ya se mencionó el caso similar con el método `authorise` de arriba, también sucede que queden escenarios imposibles de comprobar por el caso donde la entidad pueda ser nula y sus propiedades imposibles de comprobar. Este código puede ser optimizado aún así, ya que aparecen estructuras que se repiten.


```

FlightCrewMemberFlightAssignmentPublishService.java  FlightCrewMemberActivityLogCreateService.java x
73 FlightAssignment flightAssignment = this.repository.findFlightAssignmentById(assignmentId);
74 activityLog.setFlightAssignment(flightAssignment);
75
76 activityLog.setDraftMode(true);
77 super.getBuffer().addData(activityLog);
78 }
79
80 @Override
81 public void bind(final ActivityLog activityLog) {
82     super.bindObject(activityLog, "typeOfIncident", "description", "severityLevel");
83 }
84
85 @Override
86 public void validate(final ActivityLog activityLog) {
87 }
88
89
90 @Override
91 public void perform(final ActivityLog activityLog) {
92     this.repository.save(activityLog);
93 }
94
95 @Override
96 public void unbind(final ActivityLog activityLog) {
97     Dataset dataset = super.unbindObject(activityLog, "registrationMoment", "typeOfIncident", "description", "severityLevel", "draftMode");
98
99     // Show create if the assignment is completed
100     if (activityLog.getFlightAssignment().getLeg().getScheduledArrival().before(MomentHelper.getCurrentMoment()))
101         super.getResponse().addGlobal("showAction", true);
102
103     dataset.put("assignmentId", super.getRequest().getData("assignmentId", int.class));
104     super.getResponse().addData(dataset);
105 }
106
107 }
108

```

Figura 4.5: Cobertura de la funcionalidad create 2 (Activity Log).

Y en el método unbind no se puede contemplar el caso en el que la variable showAction no se active (la cual activa en el form un botón de create), ya que para llegar a este escenario sería necesario entrar a un activity log de una flight assignment vinculado a una leg que aún no ha ocurrido por lo que no es posible. Podría solucionarse dejando la variable activada por defecto.

5. Creación de índices

Tras revisar las consultas con WHERE de mi repositorio de FlightAssignment y de ActivityLog, he decidido crear los índices correspondientes para aquellas consultas que solo utilizan atributos de mi entidad FlightAssignment y ActivityLog, y que usen más atributos además de solo el id para mejorar el rendimiento de las funcionalidades:

```
27@Entity
28@Getter
29@Setter
30@Table(name = "flight_assignment", indexes = {
31    @Index(name = "idx_fa_crew_moment", columnList = "flight_crew_member_id, moment"),
32    @Index(name = "idx_fa_leg_duty", columnList = "leg_id, duty"),
33    @Index(name = "idx_fa_leg_duty_draft", columnList = "leg_id, duty, draftMode, id"),
34    @Index(name = "idx_fa_crew_moment_draft", columnList = "flight_crew_member_id, moment, draftMode")
35})
36public class FlightAssignment extends AbstractEntity {
37
```

Figura 5.1: Generación de índices 1.

```
15
16@Repository
17public interface FlightCrewMemberFlightAssignmentRepository extends AbstractRepository {
18
19    @Query("SELECT fa FROM FlightAssignment fa WHERE fa.id = :flightAssignmentId")
20    FlightAssignment findFlightAssignmentById(int flightAssignmentId);
21
22    // List my completed assignments
23    @Query("SELECT fa FROM FlightAssignment fa WHERE fa.flightCrewMember.id = :flightCrewMemberId AND fa.leg.scheduledArrival < :moment")
24    Collection<FlightAssignment> findAllCompletedFlightAssignments(Date moment, int flightCrewMemberId);
25
26    // List my planned assignments
27    @Query("SELECT fa FROM FlightAssignment fa WHERE fa.flightCrewMember.id = :flightCrewMemberId AND fa.leg.scheduledDeparture >= :moment")
28    Collection<FlightAssignment> findAllPlannedFlightAssignments(Date moment, int flightCrewMemberId);
29
30    // List published assignments
31    @Query("SELECT fa FROM FlightAssignment fa WHERE fa.draftMode = true")
32    Collection<FlightAssignment> findAllPublishedFlightAssignments();
33
34    // Show all the legs in the select choices
35    @Query("SELECT l FROM Leg l WHERE l.aircraft.airline.id = :airlineId AND l.draftMode = false")
36    Collection<Leg> findAllLegsByAirlineId(int airlineId);
37
38    // Find all the activity logs to remove the flight assignment
39    @Query("SELECT a FROM ActivityLog a WHERE a.flightAssignment.id = :flightAssignmentId")
40    Collection<ActivityLog> findAllActivityLogs(int flightAssignmentId);
41
42    @Query("SELECT fa FROM FlightAssignment fa WHERE fa.leg.id = :legId AND fa.duty = :duty")
43    FlightAssignment findFlightAssignmentByLegAndDuty(int legId, Duty duty);
44
45    @Query("SELECT COUNT(fa) > 0 FROM FlightAssignment fa WHERE fa.leg.id = :legId AND fa.duty IN ('PILOT', 'CO_PILOT') AND fa.duty = :duty AND fa.id != :id AND fa.draftMode = false")
46    Boolean hasDutyAssigned(int legId, Duty duty, int id);
47
48    @Query("SELECT COUNT(fa) > 0 FROM FlightAssignment fa WHERE fa.flightCrewMember.id = :flightCrewMemberId AND fa.moment >= :moment AND fa.draftMode = false")
49    Boolean hasFlightCrewMemberLegAssociated(int flightCrewMemberId, Date moment);
50
51    @Query("SELECT l FROM Leg l WHERE l.aircraft.airline.id = :airlineId")
52    Collection<Leg> findAllLegsFromAirline(int airlineId);
53
```

Figura 5.2: Generación de índices 2.

Para mejorar el rendimiento de las consultas de JQPL sobre la entidad FlightAssignment, se han definido los siguientes índices basados en los filtros que ya se utilizan en el repositorio relacionado con la entidad. Estos índices cubren todos los escenarios de acceso más frecuentes para reducir al mínimo los tiempo de búsqueda y el uso de recursos de la base de datos.

```

24@Entity
25@Getter
26@Setter
27@Table(name = "activity_log", indexes = {
28    @Index(name = "idx_activity_log_flight_assignment", columnList = "flight_assignment_id")
29})
30public class ActivityLog extends AbstractEntity {
31

```

Figura 5.3: Generación de índices 3.

```

12
13@Repository
14public interface FlightCrewMemberActivityLogRepository extends AbstractRepository {
15
16    // List all the activity logs from a flight assignment Id
17    @Query("SELECT a FROM ActivityLog a WHERE a.flightAssignment.id = :flightAssignmentId")
18    Collection<ActivityLog> findActivityLogsByFlightAssignmentId(int flightAssignmentId);
19
20    // Show an activity log by its id
21    @Query("SELECT a FROM ActivityLog a WHERE a.id = :activityLogId")
22    ActivityLog findActivityLogById(int activityLogId);
23
24    @Query("SELECT a FROM FlightAssignment a WHERE a.id = :id")
25    FlightAssignment findFlightAssignmentById(int id);
26
27    @Query("select a from ActivityLog a where a.flightAssignment.leg.flight.id=:flightId")
28    Collection<ActivityLog> findActivityLogsByFlightId(int flightId);
29
30    @Query("select a from ActivityLog a where a.flightAssignment.leg.id=:legId")
31    Collection<ActivityLog> findActivityLogsByLegId(int legId);
32}
33

```

Figura 5.4: Generación de índices 4.

Para la entidad `ActivityLog`, los accesos solo se centran en las relacionadas con `FlightAssignment`, ya sea con la `id` o de manera indirecta a través de `leg`. Es por eso que el índice implementado permite resolver rápidamente las consultas que obtienen todos los activity logs vinculados a un flight assignment concreto.

Dado que varias consultas tanto directas como indirectas acceden a la entidad `Leg` a través de sus relaciones con `FlightAssignment` y otras entidades, se podría sugerir la incorporación de índices adicionales en la tabla `Leg`. En concreto, podría ser beneficioso indexar columnas como `flight.id`, `scheduledDeparture` y `scheduledArrival`, ya que son utilizadas frecuentemente en filtros de búsqueda de `FlightAssignment` y `ActivityLog`. Estos índices podrían mejorar el rendimiento de las consultas considerablemente, optimizando los tiempos de respuesta en operaciones que ahora mismo están mal optimizadas como `ListPlanned` y `ListCompleted` de `FlightAssignment`. Como esta entidad no pertenece al `Student 3` se ha decidido no añadirse para no intervenir las que el `Student` oportuno considere adecuadas.

6. Conclusiones

Dado que queremos evaluar el impacto de los tests funcionales y el rendimiento del sistema, se llevaron a cabo múltiples pruebas funcionales que satisfacen con un 99,9% de cobertura para las entidades FlightAssignment, ActivityLog y sus funcionalidades.

En el testing de rendimiento se puede afirmar que en ninguno de los dos dispositivos se observaron mejoras estadísticamente significativas en los tiempos de respuesta tras la inclusión de índices. Si bien en el dispositivo 1 la reducción fue más perceptible a nivel de media, no fue suficiente para superar el umbral estadístico de confianza de 0,05. Por otro lado, el dispositivo 2, gracias a su mayor capacidad de procesamiento, ofreció tiempos de respuesta globalmente más bajos, con menor dispersión, confirmando así su superioridad técnica en términos de rendimiento.

Los intervalos de confianza alcanzados por ambos dispositivos al 95% son:
Dispositivo 1:

- Antes del cambio: [22,80 ms, 35,86 ms]
- Después del cambio: [16,81 ms, 29,34 ms]

Dispositivo 2:

- Antes del cambio: [12,02 ms, 14,81 ms]
- Después del cambio: [11,64 ms, 14,4 ms]

Además gracias a VisualVM se identificaron algunos *MIR* (métodos intensivos en recursos), que representan oportunidades de mejora adicionales, como posibles refactorizaciones o rediseño de los índices aplicados en las entidades como posibles sugerencias.