

Universidad de Sevilla

Escuela Técnica Superior de Ingeniería Informática

## Testing Report



Grado en Ingeniería Informática – Ingeniería del Software


DISEÑO Y PRUEBAS II

Curso 2024 – 2025

Fecha	Versión
03/07/2025	v2r0


<b>Grupo:</b> C2.006	
<b>Autores</b>	
Pérez Garrido, Rubén	rubpergar@alum.us.es

Repositorio GITHUB: <https://github.com/rubpergar/Acme-ANS-C2>

	<p style="text-align: center;">DISEÑO Y PRUEBAS II <b>Testing Report</b> - Rubén Pérez Garrido</p>
---	--


## Índice de contenido

1. Tabla de versiones	3
2. Resumen ejecutivo	4
3. Introducción	5
4. Functional testing	6
5. Performance testing	12
6. Comparativa máquinas	15
7. Conclusiones	19
8. Bibliografía	20

	<p style="text-align: center;">DISEÑO Y PRUEBAS II <b>Testing Report</b> - Rubén Pérez Garrido</p>
---	--

## 1. Tabla de versiones

Fecha	Versión	Descripción
19/05/2025	v1r0	Creación del documento y redacción de los puntos: Resumen ejecutivo, Introducción y parte del Functional Testing.
26/05/2025	v1r1	Corrección de contenido anterior y compleción de los puntos restantes.
03/07/2025	v2r0	Actualización de contenidos acorde a los cambios realizados para la entrega C2


	<p style="text-align: center;">DISEÑO Y PRUEBAS II <b>Testing Report</b> - Rubén Pérez Garrido</p>
---	--

## 2. Resumen ejecutivo

Para este documento de testing se han desarrollado y explicado cada una de las implementaciones presentes en los requisitos obligatorios de la entrega C2.

Se han evaluado tanto el rendimiento funcional de todas y cada una de las funciones solicitadas, en este caso, las relativas a “flight assignment” y a “activity log”. Para ello se siguió la metodología proporcionada en “S01 - Formal testing” y “S02 - Performance testing”.

El sistema muestra un comportamiento robusto en términos de funcionalidad. En términos de rendimiento, las dos computadoras en las que se ha probado el sistema, tras el estudio realizado, nos confirma que la segunda es más poderosa que la primera.

	<p style="text-align: center;">DISEÑO Y PRUEBAS II <b>Testing Report</b> - Rubén Pérez Garrido</p>
---	--

### 3. Introducción

Este documento se divide en dos secciones distintas:

1. Functional testing: se presentarán los casos de prueba implementados, agrupados por funcionalidad. Por cada uno se dará una descripción y una indicación de cuán efectivo es detectando errores. Para la efectividad, se usará el coverage del código para comprobar que se han probado todas las decisiones posibles durante la ejecución del programa y así evitar la existencia de errores.
2. Performance testing: se proporcionarán los gráficos adecuados y un intervalo de confianza del 95% para el tiempo tomado para las solicitudes en las pruebas en dos ordenadores distintos. Además, tras las pruebas en los diferentes ordenadores, se indicará cuál de estos es el más potente y ofrece mejor rendimiento.

#### 4. Functional testing

Casos de prueba relativos a asignaciones de vuelos (Flight Assignments):

acme.features.authenticated.flightCrewMember.flightAssignment	100,0 %	1.690	0	1.690
> FlightAssignmentController.java	100,0 %	42	0	42
> FlightAssignmentCreateService.java	100,0 %	472	0	472
> FlightAssignmentDeleteService.java	100,0 %	67	0	67
> FlightAssignmentListCompletedService.java	100,0 %	52	0	52
> FlightAssignmentListUncompletedService.java	100,0 %	52	0	52
> FlightAssignmentPublishService.java	100,0 %	203	0	203
> FlightAssignmentShowService.java	100,0 %	253	0	253
> FlightAssignmentUpdateService.java	100,0 %	549	0	549

- Create
  - Descripción: Se valida la creación de una asignación mediante la prueba de las restricciones de todos los campos del formulario de creación de una asignación de vuelo con valores relativos a casos positivos, negativos y de hacking. Se valida que los campos de solo lectura no permitan la intrusión de valores a través de las herramientas de desarrollador del navegador y que al hacer Post Hacking de los campos Status, Duty y Leg, salte un error 500. Se valida que un usuario no registrado como miembro (flight crew member) no tenga acceso a esta funcionalidad y que un miembro no disponible no pueda ser asignado para un flight assignment.
  - Coverage: 100%
  - Efectividad: Entendemos que una cobertura alrededor del 95% es un valor alto de efectividad y por tanto, al obtener un 100% de cobertura, se puede afirmar que el nivel de efectividad es el más alto que se puede obtener, pues todas las líneas de código de la funcionalidad son probadas y se ha verificado que no introduzcan ningún bug o error. Sin embargo, se ha detectado que hay un caso en el que no todas las ramas posibles han sido probadas:

```
public void unbind(final FlightAssignment flightAssignment) {
    int flightCrewMemberAirlineId;
    int flightCrewMemberId;

    flightCrewMemberAirlineId = this.repository.getMemberById(super.getRequest().getPrincipal().getActiveRealm().getId(), flightCrewMemberId);
    flightCrewMemberId = super.getRequest().getPrincipal().getActiveRealm().getId();

    Collection<Leg> allAvailableLegs = this.repository.findAvailableLegs(MomentHelper.getCurrentMoment());
    List<Leg> availableLegs = allAvailableLegs.stream().filter(l -> l.getFlight().getAirlineManager().getAirlineId() != flightCrewMemberAirlineId).toList();
    List<Leg> memberAssignedLegs = this.repository.getAllLegsByMemberId(flightCrewMemberId);

    Collection<Leg> compatibleLegs = new ArrayList<>();

    for (Leg candidate : availableLegs) {
        boolean isCompatible = true;

        for (Leg assigned : memberAssignedLegs) {
            boolean departureOverlap = MomentHelper.isInRange(candidate.getScheduledDeparture(), assigned.getScheduledDeparture(), assigned.getScheduledArrival());
            boolean arrivalOverlap = MomentHelper.isInRange(candidate.getScheduledArrival(), assigned.getScheduledDeparture(), assigned.getScheduledArrival());

            if (departureOverlap || arrivalOverlap) {
                isCompatible = false;
                break;
            }
        }
    }
}
```

Podemos ver en amarillo la línea

“ if (departureOverlap || arrivalOverlap) { ”. Esta línea permite que se genere un error en caso de que el horario de salida o de llegada de un tramo (*leg*) de una asignación de vuelo, al que se va a asignar un miembro de la tripulación, coincida con otro tramo de una asignación que dicho miembro ya tenga. Es decir, valida que un mismo miembro de la tripulación no pueda estar asignado a dos vuelos cuyos tramos se solapen en el tiempo.


La causa de que este bloque if se encuentre en amarillo es que no se han podido cubrir todos los estados posibles de la condición durante las pruebas. Al tratarse de una condición con dos variables booleanas, existen cuatro combinaciones posibles:

1. Ambos valores son falsos (no hay solapamiento).
2. Ambos valores son verdaderos (hay solapamiento).
3. Uno es verdadero y el otro falso (dos combinaciones posibles).

El caso que falta por cubrir, y que impide que esta línea aparezca en verde tras las pruebas de cobertura, es aquel en el que solo una de la condición `departureOverlap` es verdadera.


Este problema se debe a que falta en los datos de prueba un *leg* cuyo horario de salida se solape con otro. Esta carencia se detectó de forma tardía, y añadir un nuevo *leg* implicaría que todo el equipo de desarrollo tuviera que volver a ejecutar los tests de sus respectivas funcionalidades. Para evitar problemas de alcance y tiempo, este único caso se ha probado de forma aislada, generando datos específicos para validar que no se produce ningún fallo. El resultado fue positivo.

Por ello, aunque esta prueba no haya quedado reflejada en los informes de cobertura, se puede afirmar que la funcionalidad se comporta correctamente.

	<p style="text-align: center;">DISEÑO Y PRUEBAS II <b>Testing Report</b> - Rubén Pérez Garrido</p>
---	--

- Delete
  - Descripción: Se prueba la eliminación de una asignación de vuelo. Se valida que solo el propietario de la asignación de vuelo (sin publicar) puede eliminarla, eliminando así todos los registros de actividades vinculados a dicha asignación (si tiene). Se valida que un usuario no registrado como miembro no tenga acceso a esta funcionalidad.
  - Coverage: 100%
  - Efectividad: Máxima, ya que se han probado todas las líneas de código de la funcionalidad y se ha verificado que no introduzcan ningún bug o error.
  
- List completed
  - Descripción: Se prueba el listado de asignaciones de vuelo con tramos completados, es decir, que la fecha de finalización del tramo es anterior a la fecha actual. Se valida que un usuario no registrado como miembro no tenga acceso a esta funcionalidad y que un miembro solo pueda ver el listado de sus flight assignments con tramos completados.
  - Coverage: 100%
  - Efectividad: Máxima, ya que se han probado todas las líneas de código de la funcionalidad y se ha verificado que no introduzcan ningún bug o error.
  
- List Uncompleted
  - Descripción: Se prueba el listado de asignaciones de vuelo con tramos incompletos, es decir, que la fecha de finalización del tramo es posterior a la fecha actual. Se valida que un usuario no registrado como miembro no tenga acceso a esta funcionalidad y que un miembro solo pueda ver el listado de sus flight assignments con tramos no completados.
  - Coverage: 100%
  - Efectividad: Máxima, ya que se han probado todas las líneas de código de la funcionalidad y se ha verificado que no introduzcan ningún bug o error.
  
- Publish
  - Descripción: Se prueba la publicación de una asignación de vuelo. Se valida que solo el propietario de la asignación de vuelo puede publicarla. Se valida que un usuario no registrado como miembro no tenga acceso a esta funcionalidad y que no se pueda publicar un flight assignment con un miembro no disponible.
  - Coverage: 100%
  - Efectividad: Máxima, ya que se han probado todas las líneas de código de la funcionalidad y se ha verificado que no introduzcan ningún bug o error.
  
- Show




	<p style="text-align: center;">DISEÑO Y PRUEBAS II <b>Testing Report</b> - Rubén Pérez Garrido</p>
---	--

- Descripción: Se prueba la visualización de los detalles de una asignación de vuelo. Se valida que un usuario registrado como miembro pueda visualizar los detalles de la asignación siempre y cuando esta se encuentre publicada. El miembro al que le corresponde la asignación de vuelo siempre podrá ver los detalles independientemente de si está publicada o no. Se valida que un usuario no registrado como miembro no tenga acceso a esta funcionalidad independientemente de si la asignación está publicada o no.
  - Coverage: 100%
  - Efectividad: Máxima, ya que se han probado todas las líneas de código de la funcionalidad y se ha verificado que no introduzcan ningún bug o error.
- Update
    - Descripción: Se valida la actualización de los datos de una asignación mediante la prueba de las restricciones de todos los campos del formulario de actualización con valores relativos a casos positivos, negativos y de hacking. Se valida que los campos de solo lectura no permitan la intrusión de valores a través de las herramientas de desarrollador del navegador. Se valida que solo el propietario de la asignación de vuelo (sin publicar) puede actualizarla. Se valida que un usuario no registrado como miembro no tenga acceso a esta funcionalidad y que un miembro no disponible no pueda ser asignado para un flight assignment.
    - Coverage: 100%
    - Efectividad: Máxima, ya que se han probado todas las líneas de código de la funcionalidad y se ha verificado que no introduzcan ningún bug o error. Sin embargo, acontece la misma incidencia que el create (explicado anteriormente).

Casos de prueba relativos a registros de actividad (Activity Logs):

▼ acme.features.authenticated.flightCrewMember.activityLog	100,0 %	829	0	829
> ActivityLogController.java	100,0 %	35	0	35
> ActivityLogCreateService.java	100,0 %	191	0	191
> ActivityLogDeleteService.java	100,0 %	57	0	57
> ActivityLogListService.java	100,0 %	120	0	120
> ActivityLogPublishService.java	100,0 %	132	0	132
> ActivityLogShowService.java	100,0 %	127	0	127
> ActivityLogUpdateService.java	100,0 %	167	0	167

- Create
  - Descripción: Se valida la creación de un registro de actividad mediante la prueba de las restricciones de todos los campos del formulario de creación con valores relativos a casos positivos, negativos y de hacking. Se valida que los campos de solo lectura no permitan la intrusión de valores a través de las herramientas de desarrollador del navegador. Se valida que un miembro que no pertenece a la asignación de vuelo, no pueda introducir registros de actividad para dicha asignación. Se valida que un registro de actividad no se puede crear si la asignación de vuelo a la que se pertenece se encuentra en un tramo sin completar. Se valida que un usuario no registrado como miembro no tenga acceso a esta funcionalidad.
  - Coverage: 100%
  - Efectividad: Máxima, ya que se han probado todas las líneas de código de la funcionalidad y se ha verificado que no introduzcan ningún bug o error.
- Delete
  - Descripción: Se prueba la eliminación de un registro de actividad. Se valida que solo el propietario del registro (sin publicar) puede eliminarlo. Se valida que un usuario no registrado como miembro no tenga acceso a esta funcionalidad.
  - Coverage: 100%
  - Efectividad: Máxima, ya que se han probado todas las líneas de código de la funcionalidad y se ha verificado que no introduzcan ningún bug o error.
- List
  - Descripción: Se prueba el listado de registros de actividad. Se valida que solo el propietario de la asignación de vuelo (sin publicar) puede acceder a este listado. Se valida que un usuario no registrado como miembro no tenga acceso a esta funcionalidad.
  - Coverage: 100%
  - Efectividad: Máxima, ya que se han probado todas las líneas de código de la funcionalidad y se ha verificado que no introduzcan ningún bug o error.

	<p style="text-align: center;">DISEÑO Y PRUEBAS II <b>Testing Report</b> - Rubén Pérez Garrido</p>
---	--

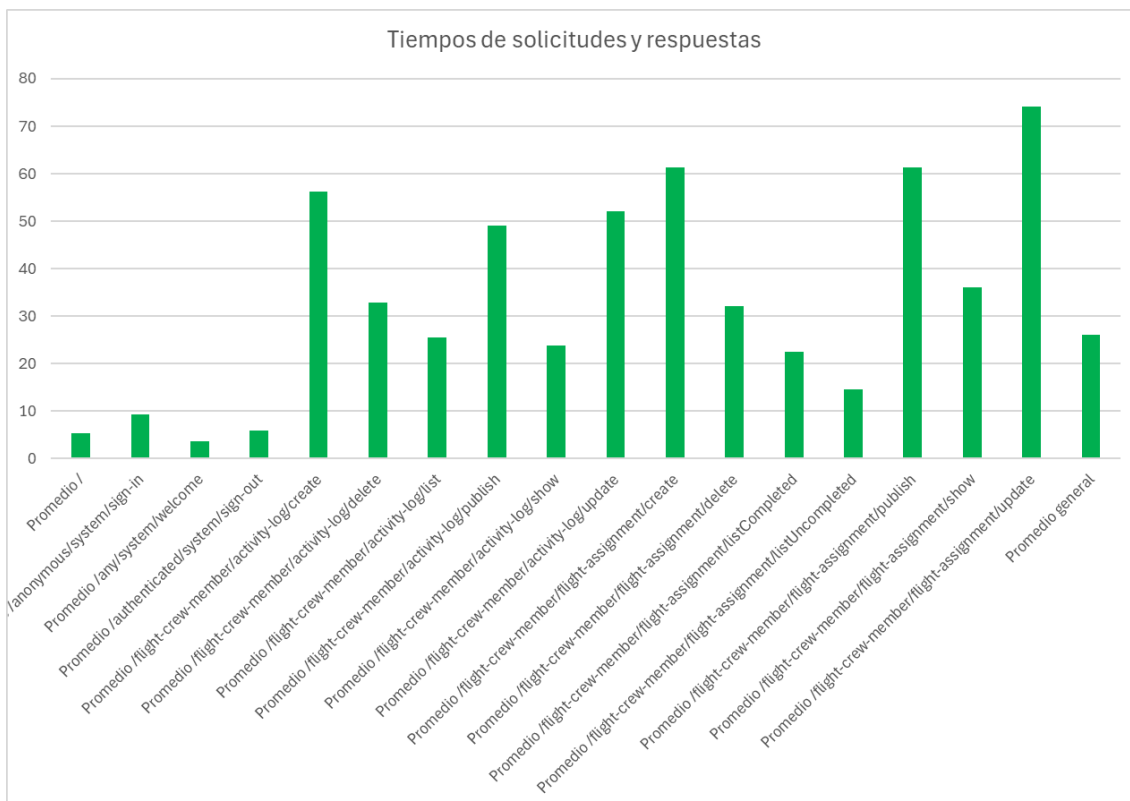
- Publish
  - Descripción: Se prueba la publicación de un registro de actividad. Se valida que solo el propietario del registro puede publicarla. Se valida que un registro no pueda ser publicado si la asignación de vuelo a la que pertenece no se encuentra publicada. Se valida que un usuario no registrado como miembro no tenga acceso a esta funcionalidad.
  - Coverage: 100%
  - Efectividad: Máxima, ya que se han probado todas las líneas de código de la funcionalidad y se ha verificado que no introduzcan ningún bug o error.
  
- Show
  - Descripción: Se prueba la visualización de los detalles de un registro de actividad. Se valida que un usuario registrado como miembro pueda visualizar los detalles del registro siempre y cuando este se encuentre publicado. El miembro al que le corresponde el registro de vuelo siempre podrá ver los detalles independientemente de si está publicada o no. Se valida que un usuario no registrado como miembro no tenga acceso a esta funcionalidad independientemente de si la asignación está publicada o no.
  - Coverage: 100%
  - Efectividad: Máxima, ya que se han probado todas las líneas de código de la funcionalidad y se ha verificado que no introduzcan ningún bug o error.
  
- Update
  - Descripción: Se valida la actualización de los datos de un registro de actividad mediante la prueba de las restricciones de todos los campos del formulario de actualización con valores relativos a casos positivos, negativos y de hacking. Se valida que los campos de solo lectura no permitan la intrusión de valores a través de las herramientas de desarrollador del navegador. Se valida que solo el propietario de la asignación de vuelo (sin publicar) puede actualizarlo. Se valida que un usuario no registrado como miembro no tenga acceso a esta funcionalidad.
  - Coverage: 100%
  - Efectividad: Máxima, ya que se han probado todas las líneas de código de la funcionalidad y se ha verificado que no introduzcan ningún bug o error.

En resumen, se han validado el 100% de las distintas funcionalidades con el objetivo de evitar la presencia de bugs y errores no esperados.

## 5. Performance testing

Tras realizar el conjunto de tests para las funcionalidades oportunas, se han realizado todos los pasos que se muestran en “S02 - Performance testing”, obteniendo los siguientes resultados:

request-path	time
Promedio /	5.28554651
Promedio /anonymous/system/sign-in	9.22070814
Promedio /any/system/welcome	3.60764947
Promedio /authenticated/system/sign-out	5.9204
Promedio /flight-crew-member/activity-log/create	56.1346538
Promedio /flight-crew-member/activity-log/delete	32.8916111
Promedio /flight-crew-member/activity-log/list	25.4412803
Promedio /flight-crew-member/activity-log/publish	49.0905171
Promedio /flight-crew-member/activity-log/show	23.8621078
Promedio /flight-crew-member/activity-log/update	52.1036625
Promedio /flight-crew-member/flight-assignment/create	61.3151274
Promedio /flight-crew-member/flight-assignment/delete	32.0910556
Promedio /flight-crew-member/flight-assignment/listCompleted	22.4172436
Promedio /flight-crew-member/flight-assignment/listUncompleted	14.4951561
Promedio /flight-crew-member/flight-assignment/publish	61.3895935
Promedio /flight-crew-member/flight-assignment/show	36.1198185
Promedio /flight-crew-member/flight-assignment/update	74.0712565
Promedio general	25.9667884



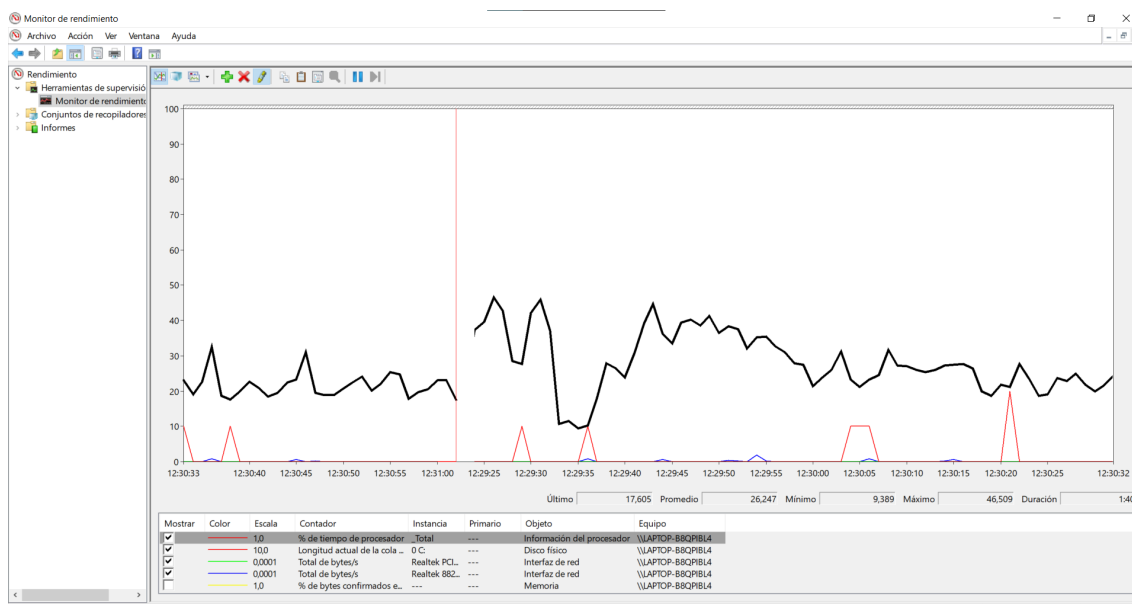
A partir de las dos imágenes anteriores, se observa que el tiempo promedio que tarda el sistema en realizar una petición es de unos 26 ms, es decir, 0,026 segundos. Viendo el gráfico de barras, se puede afirmar de forma visual, que las peticiones que más tiempo tardan en ejecutarse son las relativas a la creación, actualización y publicación de ambas entidades probadas, ya que son las que poseen mayor número de validaciones de campos y anti-hacking, además de poseer consultas que requieren la lectura y tratamiento de un gran volumen de datos de la base de datos.

Posteriormente se realizó un examen a la máquina sobre la que se estaban realizando las pruebas para obtener información acerca de ella y su rendimiento, con el objetivo de así descubrir si esta podría estar limitando la eficiencia del código. Estos fueron los resultados:

Name	Total Time (CPU)	Self Time (CPU)
acme.features.authenticated.flightCrewMember.flightAssignment.FlightAssignmentCreateService. <b>bind</b> ()	692 ms (16,1 %)	0,0 ms
acme.features.authenticated.flightCrewMember.flightAssignment.FlightAssignmentUpdateService. <b>validate</b> ()	599 ms (13,9 %)	0,0 ms
acme.features.authenticated.flightCrewMember.flightAssignment.FlightAssignmentCreateService. <b>unbind</b> ()	506 ms (11,8 %)	0,0 ms
acme.features.authenticated.flightCrewMember.activityLog.ActivityLogCreateService. <b>validate</b> ()	503 ms (11,7 %)	0,0 ms
acme.features.authenticated.flightCrewMember.flightAssignment.FlightAssignmentShowService. <b>unbind</b> ()	399 ms (9,3 %)	0,0 ms
acme.features.authenticated.flightCrewMember.flightAssignment.FlightAssignmentCreateService. <b>validate</b> ()	200 ms (4,7 %)	0,0 ms
acme.features.authenticated.flightCrewMember.activityLog.ActivityLogUpdateService. <b>authorise</b> ()	199 ms (4,6 %)	0,0 ms
acme.features.authenticated.flightCrewMember.activityLog.ActivityLogUpdateService. <b>load</b> ()	197 ms (4,6 %)	0,0 ms
acme.features.authenticated.flightCrewMember.flightAssignment.FlightAssignmentShowService. <b>load</b> ()	101 ms (2,4 %)	0,0 ms
acme.features.authenticated.flightCrewMember.flightAssignment.FlightAssignmentListUncompletedService. <b>load</b> ()	101 ms (2,4 %)	0,0 ms
acme.features.authenticated.flightCrewMember.activityLog.ActivityLogListService. <b>unbind</b> ()	100 ms (2,3 %)	0,0 ms
acme.features.authenticated.flightCrewMember.flightAssignment.FlightAssignmentUpdateService. <b>unbind</b> ()	100 ms (2,3 %)	0,0 ms
acme.features.authenticated.flightCrewMember.activityLog.ActivityLogCreateService. <b>load</b> ()	100 ms (2,3 %)	0,0 ms
acme.features.authenticated.flightCrewMember.activityLog.ActivityLogListService. <b>load</b> ()	99,8 ms (2,3 %)	0,0 ms
acme.features.authenticated.flightCrewMember.activityLog.ActivityLogPublishService. <b>validate</b> ()	99,8 ms (2,3 %)	0,0 ms
acme.features.authenticated.flightCrewMember.flightAssignment.FlightAssignmentUpdateService. <b>perform</b> ()	99,6 ms (2,3 %)	0,0 ms
acme.features.authenticated.flightCrewMember.activityLog.ActivityLogUpdateService. <b>validate</b> ()	98,8 ms (2,3 %)	0,0 ms
acme.features.authenticated.flightCrewMember.flightAssignment.FlightAssignmentUpdateService. <b>bind</b> ()	98,8 ms (2,3 %)	0,0 ms
acme.features.authenticated.flightCrewMember.flightAssignment.FlightAssignmentUpdateService. <b>authorise</b> ()	0,0 ms (0 %)	0,0 ms

Respecto al software, en la imagen superior podemos ver que los métodos que más tiempo tardan en ejecutarse son el bind y el unbind de la creación de asignaciones de vuelo, el bind de la actualización de asignaciones de vuelo y el validador de la creación de registros de actividad. Sin embargo, el “Self Time” vale 0,0ms, lo que nos indica que el código que se ha realizado no es el que está consumiendo ese tiempo, sino el método que invoca.

Respecto al hardware, tenemos los siguientes resultados:



A simple vista se observa que no se alcanza el 100% para ninguno de los valores estudiados, por lo que se puede afirmar que el sistema no tiene ningún cuello de botella que pueda comprometer la eficiencia de la aplicación.

Una vez verificados estos puntos y habiendo estudiado el comportamiento del sistema al ejecutar la aplicación, se estudia el añadir índices para las consultas SQL que lo requieran para optimizar la eficiencia del sistema. Tras el estudio de todas las consultas que requieran de algunas de las entidades que he desarrollado (flight assignment, activity log y flight crew member), se observa que el único índice que se requiere es el relativo a esta consulta:

```
@Query("SELECT DISTINCT fa.leg FROM FlightAssignment fa WHERE fa.flightCrewMember.id = :memberId AND fa.leg.id != :legId")
List<Leg> getAllLegsByMemberIdExceptSelfLeg(int memberId, int legId);
```

Como la consulta emplea los identificadores de esas dos entidades conectadas con flight assignment, se crea un índice compuesto con ambas en dicha entidad:

```
@Table(indexes = {
    @Index(columnList = "flight_crew_member_id, leg_id")
})
public class FlightAssignment extends AbstractEntity {
```

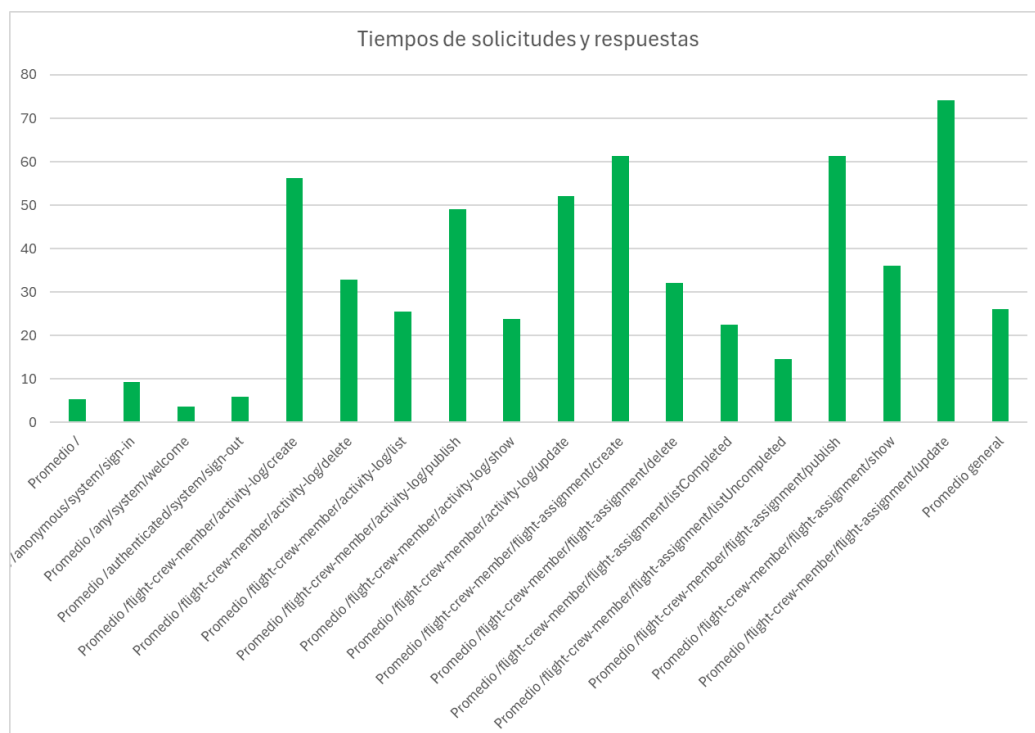
El resto de posibles índices no son necesarios añadirlos ya que el framework los crea automáticamente.

## 6. Comparativa máquinas

Se comparará ahora el rendimiento de la aplicación en la máquina donde se ha desarrollado el código (Ordenador 1) con el de otra compañera de grupo (Ordenador 2). Estos son los resultados:

Ordenador 1:

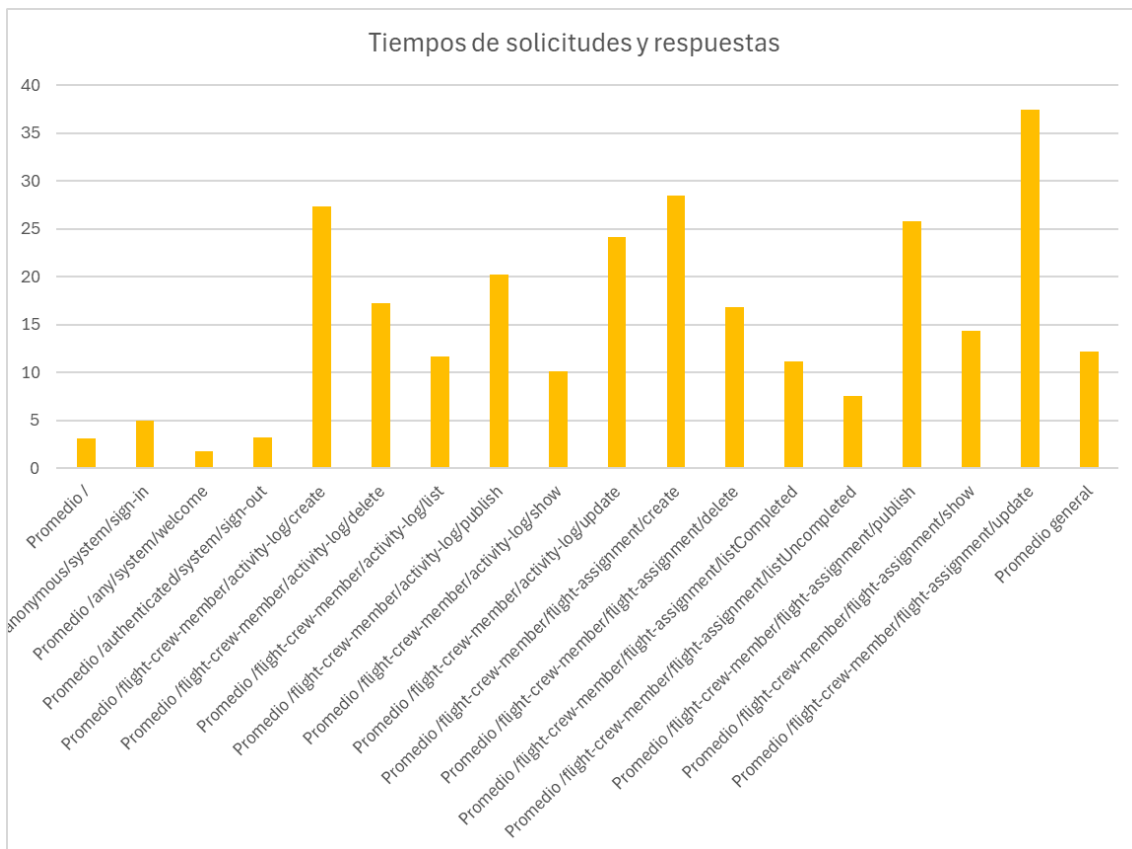
request-path	time
Promedio /	5.28554651
Promedio /anonymous/system/sign-in	9.22070814
Promedio /any/system/welcome	3.60764947
Promedio /authenticated/system/sign-out	5.9204
Promedio /flight-crew-member/activity-log/create	56.1346538
Promedio /flight-crew-member/activity-log/delete	32.8916111
Promedio /flight-crew-member/activity-log/list	25.4412803
Promedio /flight-crew-member/activity-log/publish	49.0905171
Promedio /flight-crew-member/activity-log/show	23.8621078
Promedio /flight-crew-member/activity-log/update	52.1036625
Promedio /flight-crew-member/flight-assignment/create	61.3151274
Promedio /flight-crew-member/flight-assignment/delete	32.0910556
Promedio /flight-crew-member/flight-assignment/listCompleted	22.4172436
Promedio /flight-crew-member/flight-assignment/listUncompleted	14.4951561
Promedio /flight-crew-member/flight-assignment/publish	61.3895935
Promedio /flight-crew-member/flight-assignment/show	36.1198185
Promedio /flight-crew-member/flight-assignment/update	74.0712565
Promedio general	25.9667884






Ordenador 2:

request-path	time
<b>Promedio /</b>	3.14126667
<b>Promedio /anonymous/system/sign-in</b>	5.00927965
<b>Promedio /any/system/welcome</b>	1.76850479
<b>Promedio /authenticated/system/sign-out</b>	3.20966
<b>Promedio /flight-crew-member/activity-log/create</b>	27.3754738
<b>Promedio /flight-crew-member/activity-log/delete</b>	17.2496333
<b>Promedio /flight-crew-member/activity-log/list</b>	11.6592141
<b>Promedio /flight-crew-member/activity-log/publish</b>	20.1885293
<b>Promedio /flight-crew-member/activity-log/show</b>	10.1841373
<b>Promedio /flight-crew-member/activity-log/update</b>	24.1027825
<b>Promedio /flight-crew-member/flight-assignment/create</b>	28.4362986
<b>Promedio /flight-crew-member/flight-assignment/delete</b>	16.8221
<b>Promedio /flight-crew-member/flight-assignment/listCompleted</b>	11.1161436
<b>Promedio /flight-crew-member/flight-assignment/listUncompleted</b>	7.54524035
<b>Promedio /flight-crew-member/flight-assignment/publish</b>	25.8042516
<b>Promedio /flight-crew-member/flight-assignment/show</b>	14.3791574
<b>Promedio /flight-crew-member/flight-assignment/update</b>	37.4805032
<b>Promedio general</b>	12.1835727






	<p style="text-align: center;">DISEÑO Y PRUEBAS II <b>Testing Report</b> - Rubén Pérez Garrido</p>
---	--

A simple vista se observa que el ordenador 2 arroja mejores tiempos de manera general para todas las tareas, siendo casi 13 ms más rápido que el ordenador 1. Continuando con el estudio comparativo tenemos lo siguiente:

<i>Ordenador 1</i>			<i>Ordenador 2</i>		
Media	25.96678841		Media	12.18357269	
Error típico	0.784470098		Error típico	0.395115878	
Mediana	17.0299		Mediana	7.69855	
Moda	3.6244		Moda	1.3632	
Desviación estándar	27.5571293		Desviación estándar	13.87976337	
Varianza de la muestra	759.3953753		Varianza de la muestra	192.6478313	
Curtosis	7.229973779		Curtosis	16.06156312	
Coefficiente de asimetría	2.048293698		Coefficiente de asimetría	2.863218021	
Rango	250.1807		Rango	163.7038	
Mínimo	2.4549		Mínimo	0.9084	
Máximo	252.6356		Máximo	164.6122	
Suma	32043.0169		Suma	15034.5287	
Cuenta	1234		Cuenta	1234	
Nivel de confianza(95.0%)	1.539043904		Nivel de confianza(95.0%)	0.775173821	
Interval (ms)	24.42774451	27.5058323	Interval (ms)	11.40839887	12.9587465
Interval (s)	0.024427745	0.02750583	Interval (s)	0.011408399	0.01295875

Con la imagen superior, se reafirma lo expuesto anteriormente, ya que el ordenador 2 posee un intervalo de confianza (11.40, 12.95) inferior al del ordenador 1 (24.42, 27.50), dejando ver que es muy posible que la segunda máquina sea más rápida y eficaz a la hora de ejecutar la aplicación. Para confirmar esta hipótesis, utilizaremos el z-test:

	<i>Ordenador 1</i>	<i>Ordenador 2</i>
Media	25.96678841	12.18357269
Varianza (conocida)	771	194
Observaciones	1234	1234
Diferencia hipotética de las medias	0	
z	15.58635187	
P(Z<=z) una cola	0	
Valor crítico de z (una cola)	1.644853627	
P(Z<=z) dos colas	0	
Valor crítico de z (dos colas)	1.959963985	

	<p style="text-align: center;">DISEÑO Y PRUEBAS II <b>Testing Report</b> - Rubén Pérez Garrido</p>
---	--

El valor  $p$  asociado a la prueba  $Z$  de dos colas ( $P(Z \leq z)$  dos colas) nos permite determinar si existe evidencia estadísticamente significativa para afirmar que uno de los ordenadores presenta un mejor rendimiento que el otro. En este análisis, se ha establecido un nivel de significación ( $\alpha$ ) de 0,05.


Se pueden presentar las siguientes situaciones:

1. Si el valor  $p$  es menor o igual a 0,05, se concluye que existe una diferencia significativa entre los tiempos medios de ejecución de los dos ordenadores, por lo que se puede realizar la comparativa de los promedios de los tiempos.
2. Si el valor  $p$  es mayor a 0,05, implica que no se dispone de evidencia suficiente para afirmar que uno de los ordenadores sea más eficiente que el otro ya que las diferencias globales entre los tiempos no son significativas.
3. Si el valor  $p$  es próximo a 0,05, se recomienda considerar la ampliación de la muestra para obtener resultados más precisos y robustos ya que los resultados del test no son conclusivos.

En el caso analizado, el valor  $p$  obtenido es prácticamente 0, lo que se traduce en una probabilidad extremadamente baja de que la diferencia observada en los tiempos medios de ejecución sea debida al azar.


Por tanto, se puede concluir que existe una diferencia estadísticamente significativa en los rendimientos de los dos ordenadores. Observando los valores medios de los tiempos de ejecución, se concluye que el ordenador 2 es significativamente más eficiente, con un tiempo medio de 12.18 ms, frente a los 25.96 ms del ordenador 1.

Esta diferencia es lo suficientemente clara y robusta como para afirmar que, en este contexto de pruebas, el ordenador 2 ofrece un rendimiento superior.

	<div>DISEÑO Y PRUEBAS II</div> <div>Testing Report - Rubén Pérez Garrido</div>
---	--

## 7. Conclusiones

Tras la elaboración de este documento sobre testing, se ha concluido que esta fase del ciclo de vida de un proyecto es crucial, ya que aporta mucha información útil sobre factores que no se tienen en cuenta o que pueden pasar por alto durante el desarrollo. Validar que todas las funcionalidades desarrolladas operen correctamente y sean revisadas minuciosamente para minimizar errores o bugs, además de asegurar que el rendimiento esté optimizado al máximo, son factores fundamentales de cara al cliente. Un sistema bien probado permite que el usuario final lo utilice de manera rápida e intuitiva, evitando problemas que puedan afectar negativamente su experiencia. Además, un proceso de testing riguroso contribuye a la satisfacción del cliente y a la reputación del producto, garantizando que las expectativas de calidad y eficiencia sean cumplidas de manera consistente.

	<p style="text-align: center;">DISEÑO Y PRUEBAS II <b>Testing Report</b> - Rubén Pérez Garrido</p>
---	--

## 8. Bibliografía

- 06 Annexes – Material proporcionado en la asignatura Diseño y Pruebas II por la Universidad de Sevilla.
- L04 - S01 - Formal testing - Material proporcionado en la asignatura Diseño y Pruebas II por la Universidad de Sevilla.
- L04 - S02 - Performance testing - Material proporcionado en la asignatura Diseño y Pruebas II por la Universidad de Sevilla.