

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
26/05/2025	v1r1


Grupo: C1.006	
Autores	
Pérez Garrido, Rubén	rubpergar@alum.us.es

Repositorio GITHUB: <https://github.com/rubpergar/Acme-ANS-D04.git>

	<p style="text-align: center;">DISEÑO Y PRUEBAS II Testing Report - 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	11
6. Comparativa máquinas	14
7. Conclusiones	18
8. Bibliografía	19

	<p style="text-align: center;">DISEÑO Y PRUEBAS II Testing Report - 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.


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

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 D04.

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 Testing Report - 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 %	1.138	0	1.138
> FlightAssignmentController.java	100,0 %	42	0	42
> FlightAssignmentCreateService.java	100,0 %	347	0	347
> FlightAssignmentDeleteService.java	100,0 %	67	0	67
> FlightAssignmentListCompletedService.java	100,0 %	51	0	51
> FlightAssignmentListUncompletedService.java	100,0 %	51	0	51
> FlightAssignmentPublishService.java	100,0 %	61	0	61
> FlightAssignmentShowService.java	100,0 %	159	0	159
> FlightAssignmentUpdateService.java	100,0 %	360	0	360

- 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. Se valida que un usuario no registrado como miembro (flight crew member) no tenga acceso a esta funcionalidad.
 - 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:


```
// los miembros no pueden tener varios leg asignados simultáneamente
if (flightAssignment.getFlightCrewMember() != null && flightAssignment.getLeg() != null) {
    List<Leg> legsByMember = this.repository.getAllLegsByMemberId(flightAssignment.getFlightCrewMember().getId());

    for (Leg leg : legsByMember) {
        boolean departureIncompatible = MomentHelper.isInRange(flightAssignment.getLeg().getScheduledDeparture(), leg.getScheduledDeparture(), leg.getScheduledArrival());
        boolean arrivalIncompatible = MomentHelper.isInRange(flightAssignment.getLeg().getScheduledArrival(), leg.getScheduledDeparture(), leg.getScheduledArrival());

        if (departureIncompatible || arrivalIncompatible) {
            super.state(false, "flightCrewMember", "acme.validation.flight-assignment.incompatible-legs.message");
            break;
        }
    }
}
```


Podemos ver en amarillo la línea

“ if (departureIncompatible || arrivalIncompatible) { ”. Esta línea permite hacer que salte un error en el 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, sea el mismo que otro tramo de una asignación de vuelo que ya posea dicho miembro, es decir, valida que un mismo miembro de la tripulación no esté asignado a dos vuelos con tramos que se solapan en el tiempo. La causa de

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

que este bloque if se encuentre en amarillo, es que no se han podido probar todos los estados de la condición debido a la limitación de los datos que se encuentran en la aplicación (no es posible probar todos casos ya que requiere la modificación de los datos de población de otro desarrollador para otra funcionalidad). Sin embargo, se ha probado creando datos nuevos y no se contempla ningún fallo.

- **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.
 - 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.
 - 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.
 - Coverage: 100%
 - Efectividad: Máxima, ya que se han probado todas las líneas de código de la

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

funcionalidad y se ha verificado que no introduzcan ningún bug o error.

- Show

- 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.
- 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 %	833	0	833
> 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 %	136	0	136
> 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 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 Testing Report - 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.

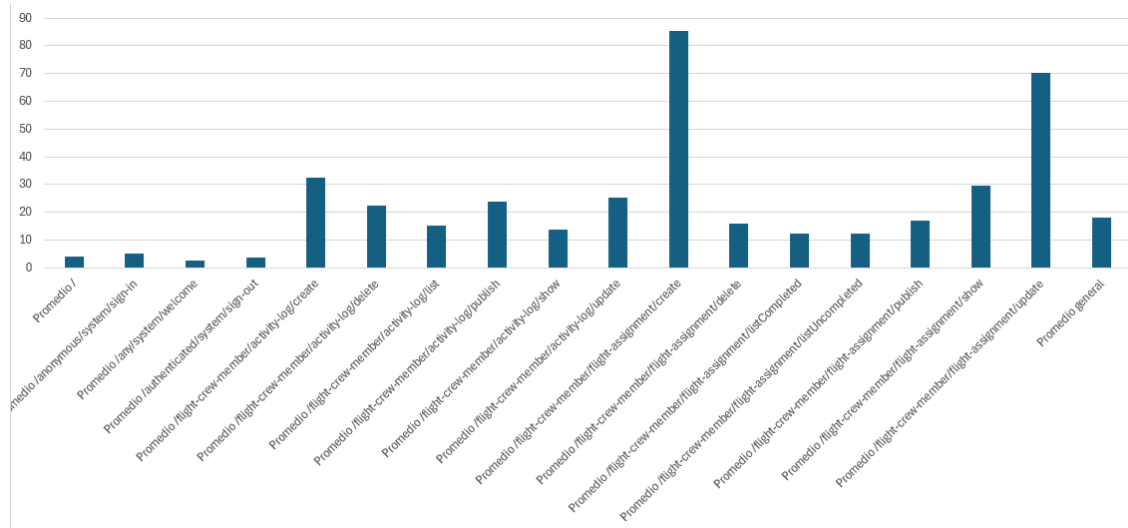
- Update
 - Descripción: Se valida la actualización de los datos de un registro de actividad mediante la prueba de las restricciones de todos los campos del formulario de actualización con valores relativos a casos positivos, negativos y de hacking. Se valida que los campos de solo lectura no permitan 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:

Promedio /	3.944033
Promedio /anonymous/system/sign-in	5.309669
Promedio /any/system/welcome	2.656484
Promedio /authenticated/system/sign-out	3.561988
Promedio /flight-crew-member/activity-log/create	32.65588
Promedio /flight-crew-member/activity-log/delete	22.5852
Promedio /flight-crew-member/activity-log/list	15.17921
Promedio /flight-crew-member/activity-log/publish	23.9873
Promedio /flight-crew-member/activity-log/show	13.63448
Promedio /flight-crew-member/activity-log/update	25.43775
Promedio /flight-crew-member/flight-assignment/create	85.1774
Promedio /flight-crew-member/flight-assignment/delete	15.8666
Promedio /flight-crew-member/flight-assignment/listCompleted	12.20371
Promedio /flight-crew-member/flight-assignment/listUncompleted	12.39401
Promedio /flight-crew-member/flight-assignment/publish	16.9325
Promedio /flight-crew-member/flight-assignment/show	29.56923
Promedio /flight-crew-member/flight-assignment/update	70.13725
Promedio general	18.03684



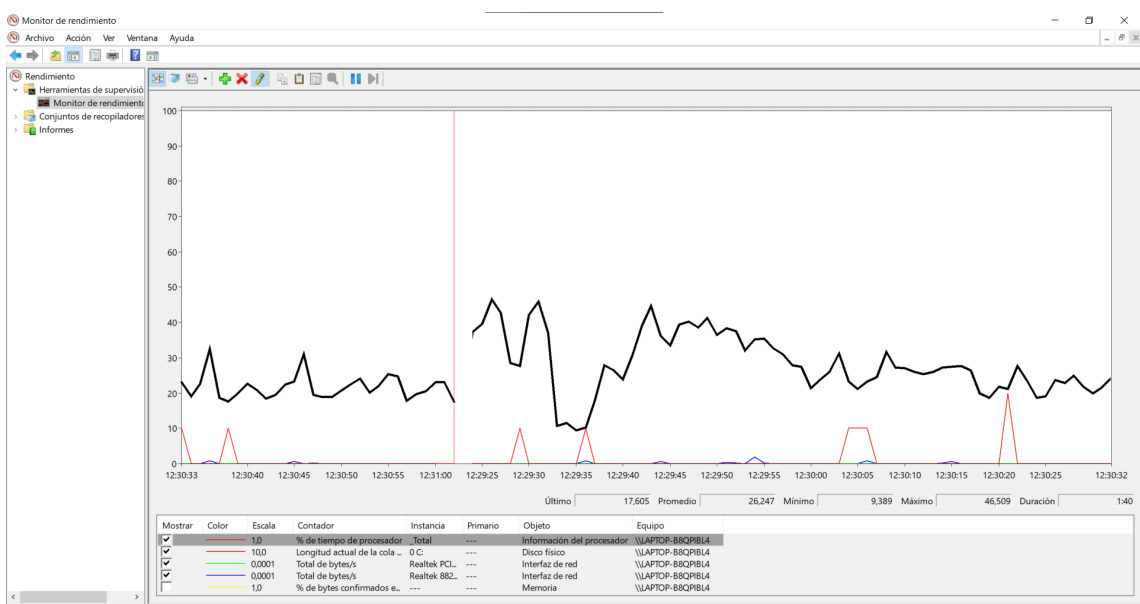
A partir de las dos imágenes anteriores, se observa que el tiempo promedio que tarda el sistema en realizar una petición es de unos 18 ms, es decir, 0,018 segundos, bastante rápido. Viendo el gráfico de barras, se puede afirmar de forma visual, que las peticiones que más tiempo tardan en ejecutarse son las relativas a la creación y actualización de las asignaciones de vuelo, ya que son las que más validaciones y mayor cantidad de datos manejan.


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



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

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, sin embargo, se detecta que no es necesario añadir ningún índice ya que, si observamos las imágenes inferiores correspondientes a los repositorios de las consultas SQL de las entidades que estamos trabajando, vemos que las consultas se realizan usando los identificadores de otras entidades que están relacionadas con la que se trabaja, por lo que no es necesario añadir dichos índices porque el framework ya los crea automáticamente.

```
public interface FlightAssignmentRepository extends AbstractRepository {

    @Query("SELECT fa from FlightAssignment fa where fa.leg.scheduledArrival < CURRENT_TIMESTAMP AND fa.flightCrewMember.id = :id")
    Collection<FlightAssignment> getCompletedFlightAssignmentsByMemberId(int id);

    @Query("SELECT fa from FlightAssignment fa where fa.leg.scheduledArrival > CURRENT_TIMESTAMP AND fa.flightCrewMember.id = :id")
    Collection<FlightAssignment> getUncompletedFlightAssignmentsByMemberId(int id);

    @Query("SELECT fa from FlightAssignment fa where fa.id = :id")
    FlightAssignment getFlightAssignmentById(int id);

    @Query("SELECT l from Leg l")
    Collection<Leg> findAllLegs();

    @Query("SELECT fcm from FlightCrewMember fcm")
    Collection<FlightCrewMember> findAllMembers();

    @Query("SELECT al from ActivityLog al WHERE al.flightAssignment.id = :id")
    Collection<ActivityLog> getAllActivityLogsFromAssignmentId(int id);

    @Query("SELECT fa from FlightAssignment fa WHERE fa.leg.id = :id")
    List<FlightAssignment> getAllFlightAssignmentsByLegId(int id);

    @Query("SELECT DISTINCT fa.leg FROM FlightAssignment fa WHERE fa.flightCrewMember.id = :id")
    List<Leg> getAllLegsByMemberId(int id);

    @Query("SELECT COUNT(1) > 0 FROM Leg l WHERE l.id = :id AND l.scheduledDeparture < CURRENT_TIMESTAMP")
    boolean isLegConcluded(int id);
}
```

```
public interface ActivityLogRepository extends AbstractRepository {

    @Query("select al from ActivityLog al WHERE al.flightAssignment.id = :id")
    Collection<ActivityLog> getActivityLogsByMasterId(int id);

    @Query("select al from ActivityLog al WHERE al.id = :id")
    ActivityLog getActivityLogById(int id);

    @Query("select fa from FlightAssignment fa where fa.id = :id")
    FlightAssignment findFlightAssignmentById(int id);

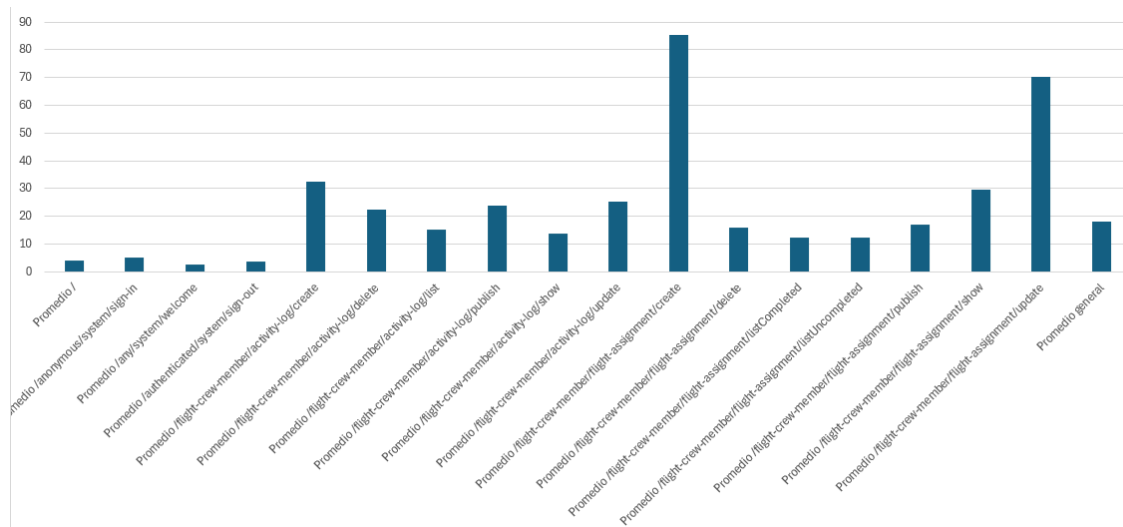
    @Query("select al.flightAssignment from ActivityLog al where al.id = :id")
    FlightAssignment findFlightAssignmentByActivityLogId(int id);
}
```

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:

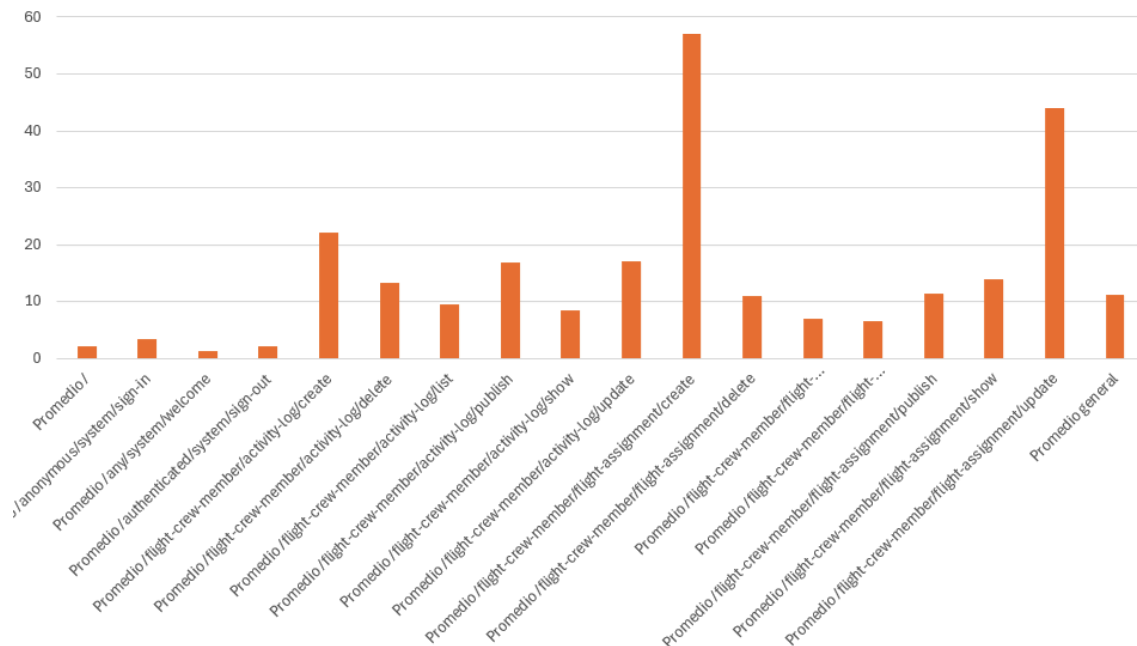
Ordenador 1:


Promedio /	3.944033
Promedio /anonymous/system/sign-in	5.309669
Promedio /any/system/welcome	2.656484
Promedio /authenticated/system/sign-out	3.561988
Promedio /flight-crew-member/activity-log/create	32.65588
Promedio /flight-crew-member/activity-log/delete	22.5852
Promedio /flight-crew-member/activity-log/list	15.17921
Promedio /flight-crew-member/activity-log/publish	23.9873
Promedio /flight-crew-member/activity-log/show	13.63448
Promedio /flight-crew-member/activity-log/update	25.43775
Promedio /flight-crew-member/flight-assignment/create	85.1774
Promedio /flight-crew-member/flight-assignment/delete	15.8666
Promedio /flight-crew-member/flight-assignment/listCompleted	12.20371
Promedio /flight-crew-member/flight-assignment/listUncompleted	12.39401
Promedio /flight-crew-member/flight-assignment/publish	16.9325
Promedio /flight-crew-member/flight-assignment/show	29.56923
Promedio /flight-crew-member/flight-assignment/update	70.13725
Promedio general	18.03684



Ordenador 2:

Promedio /	2.16364535
Promedio /anonymous/system/sign-in	3.45666667
Promedio /any/system/welcome	1.27280708
Promedio /authenticated/system/sign-out	2.18316538
Promedio /flight-crew-member/activity-log/create	22.1622946
Promedio /flight-crew-member/activity-log/delete	13.37165
Promedio /flight-crew-member/activity-log/list	9.54964286
Promedio /flight-crew-member/activity-log/publish	16.794475
Promedio /flight-crew-member/activity-log/show	8.47214444
Promedio /flight-crew-member/activity-log/update	16.9868784
Promedio /flight-crew-member/flight-assignment/create	57.0309393
Promedio /flight-crew-member/flight-assignment/delete	11.03332
Promedio /flight-crew-member/flight-assignment/listCompleted	7.10486875
Promedio /flight-crew-member/flight-assignment/listUncompleted	6.60435556
Promedio /flight-crew-member/flight-assignment/publish	11.41918
Promedio /flight-crew-member/flight-assignment/show	13.8959042
Promedio /flight-crew-member/flight-assignment/update	44.0886533
Promedio general	11.2145263



	<p style="text-align: center;">DISEÑO Y PRUEBAS II Testing Report - 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 8ms más rápido que el ordenador 1. Continuando con el estudio comparativo tenemos lo siguiente:

<i>Ordenador 1</i>			<i>Ordenador 2</i>		
Media	20.10459272		Media	12.54294494	
Error típico	1.015201416		Error típico	0.676013769	
Mediana	11.8347		Mediana	5.9817	
Moda	8.9667		Moda	1.1412	
Desviación estándar	24.08831446		Desviación estándar	16.04019852	
Varianza de la muestra	580.2468937		Varianza de la muestra	257.2879686	
Curtosis	3.267662565		Curtosis	5.503207016	
Coefficiente de asimetría	1.899943499		Coefficiente de asimetría	2.280502561	
Rango	118.3717		Rango	92.6159	
Mínimo	1.8107		Mínimo	0.6129	
Máximo	120.1824		Máximo	93.2288	
Suma	11318.8857		Suma	7061.678	
Cuenta	563		Cuenta	563	
Nivel de confianza(95.0%)	1.994052589		Nivel de confianza(95.0%)	1.327822228	


Interval (ms)	18.11054013	22.0986453	Interval (ms)	11.21512271	13.8707672
Interval (s)	0.01811054	0.02209865	Interval (s)	0.011215123	0.01387077

Con la imagen superior, se reafirma lo expuesto anteriormente, ya que el ordenador 2 posee un intervalo de confianza (11.21, 13.87) inferior al del ordenador 1 (18.11, 22.09), 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	20.10459272	12.54294494
Varianza (conocida)	580	257
Observaciones	563	563
Diferencia hipotética de las medias	0	
z	6.201659426	
P(Z<=z) una cola	2.79354E-10	
Valor crítico de z (una cola)	1.644853627	
Valor crítico de z (dos colas)	5.58709E-10	
Valor crítico de z (dos colas)	1.959963985	


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

Nos fijamos en el primer valor crítico de z (dos colas), es decir, $5.58709E-10$ (0.000000000558709). Estamos trabajando con un nivel de confianza del 95% ($0,95$), luego nuestro valor $\alpha = 1 - 0,95 = 0,05$. De esta manera, podemos observar que el p-valor obtenido, se encuentra dentro del intervalo $[0, 0,05)$ y muy próximo a 0. Esto nos indica que los tests realizados con las dos máquinas son distintos y el rendimiento varía de forma notoria entre ambas pruebas. Gracias a este estudio, se puede concluir afirmando de forma definitiva que el ordenador 2 es más potente, computacionalmente hablando, y ofrece mejor rendimiento que el ordenador 1.

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

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 Testing Report - 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.