

3 Julio de 2025



TESTING REPORT STUDENT 3

REPOSITORIO: <https://github.com/AlbertoValenzuelaMunoz1/DP2-C2.002>

GRUPO: C2.002

Fernando Cobos García (fercobgar@alum.us.es)

Contenido

Resumen ejecutivo	3
Introducción	3
Pruebas realizadas	4
3.1. Testing flightAssignment.....	4
Procedimientos comunes.....	4
Create.....	5
Update.....	6
Publish.....	6
Delete	7
Show y List	7
3.2. Testing activityLog.....	9
Procedimientos comunes.....	9
Create.....	10
Update.....	11
Publish.....	12
Delete	12
Show y List	13
Rendimiento obtenido	14
4.1. Gráficos rendimiento	14
4.2. Datos estadísticos	15
Comparativa rendimiento tras creación índices.....	16
Comparativa entre distintos ordenadores.....	17
Conclusión.....	17
Bibliografía.....	18

Resumen ejecutivo

Este informe presenta los resultados de las pruebas funcionales y de rendimiento realizadas sobre el sistema desarrollado. En la primera parte, se detalla el procedimiento realizado para probar la aplicación, de forma general para todas las entidades y a continuación se han detallados los detalles concretos para probar algunas entidades. Las pruebas han permitido identificar y corregir diversos fallos críticos, especialmente en áreas como autenticación y validación de datos de entrada.

En la segunda parte, se analizó el rendimiento del sistema ejecutando las pruebas funcionales en dos equipos con diferente capacidad de procesamiento. Se recopilaron los tiempos de respuesta y se generaron intervalos de confianza del 95% para cada conjunto de datos. Asimismo, se realizó una prueba de hipótesis estadística con un 95% de confianza para realizar una comparativa entre el rendimiento obtenido antes y después de la creación de los índices.

Tabla de versiones

Versión	Fecha	Descripción
1.0.0	03/07/2025	Creación documento e informe de pruebas realizadas Informe de rendimiento y comparativa tras la creación de índices Comparativa de rendimiento con otro miembro del grupo y finalización del documento

Introducción

Este documento recoge de manera detallada los resultados obtenidos durante el proceso de pruebas del sistema desarrollado, tanto a nivel funcional como de rendimiento. El objetivo principal de este informe es evaluar la calidad del software mediante la verificación del cumplimiento de sus funcionalidades esperadas y el análisis de su en diferentes entornos de ejecución.

Las pruebas funcionales se centraron en verificar que las características clave del sistema operan correctamente en distintos escenarios, incluyendo tanto casos de uso comunes como situaciones de error. Para ello, se diseñaron casos de prueba específicos para cada

funcionalidad del sistema, los cuales permitieron detectar errores, validar la lógica de negocio y comprobar la robustez de las validaciones implementadas.

Por otro lado, las pruebas de rendimiento se llevaron a cabo para medir el tiempo de respuesta del sistema durante la ejecución de las pruebas funcionales, utilizando dos ordenadores con distintas capacidades de hardware y en el mismo ordenador antes y después de la creación de índices. Esta evaluación permitió generar intervalos de confianza del 95% y realizar un contraste de hipótesis que permitió determinar en cuál de los equipos el sistema se desempeña mejor.

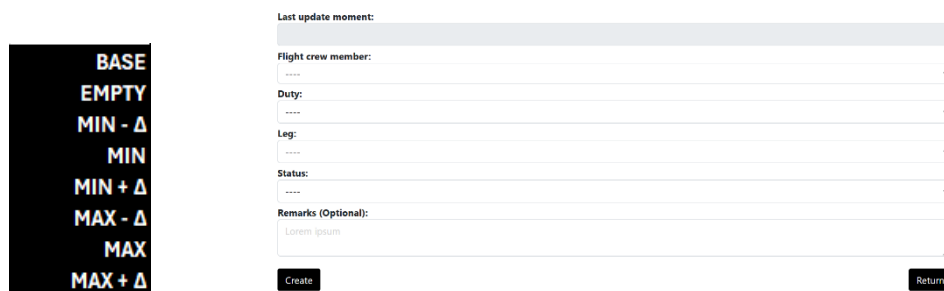
Este documento se estructura en dos capítulos principales. En el Capítulo 1, se presenta el conjunto de pruebas funcionales, organizadas por característica. En el Capítulo 2, se expone el análisis de las pruebas de rendimiento, incluyendo gráficos representativos, los intervalos de confianza calculados y los resultados del contraste estadístico.

Pruebas realizadas

3.1. Testing flightAssignment

Procedimientos comunes

- Para los update/create/publish .safe, cada atributo se ha probado con los valores necesarios para probar todos los posibles casos. La propiedad Flight crew member es un desplegable con los crew member a los que le podemos asignar la flight assignment, que serán a todos aquellos que sean de su misma aerolínea. Duty es un enum que indica que rol tiene el member, leg son las legs asignables al crew member, status es otro enum que indica la disponibilidad del member y remark es un string opcional con un límite de 255 caracteres. Primero de todo habría que enviar un formulario vacío. Hay que probar los siguientes casos:



- Para la mayoría de pruebas .hack se hace uso de la herramienta de desarrollador de Firefox para poder modificar el id oculto de los formularios, siempre probaremos a poner un id de una assignment que no esté publicada y no pertenece a nuestro member, de un assignment que no exista y de un assignment que ya está publicado. Hay que abrir la herramienta de desarrollador antes de que se inicie el record, si no se hace una petición GET extra y al hacer el replay de las pruebas da fallo.

```
<form id="form"> (event)
  <input id="id" name="id" value="0" type="hidden"> (event)
  <input id="version" name="version" value="0" type="hidden"> (event)
  <input id="_csrf" name="_csrf" value="2ccfc408-ce05-4727-a6bc-6654f2731846" type="hidden"> (event)
  <div class="form-group"> ... </div>
  <div class="form-group"> ... </div>
  <script type="text/javascript"> ... </script>
  <div class="form-group"> ... </div>
  <script type="text/javascript"> ... </script>
  <div class="form-group"> ... </div>
```

Create

- Archivo .safe

Como se comenta en los procedimientos comunes, hay que probar a enviar un formulario vacío, seguido atributo por atributo probar con cada valor que se genera en el excel “sample-data” dependiendo del tipo de atributo, por último, hay que enviar un formulario correcto para crear un flightAssignment. Hacemos shut down para acabar con el record.

Last update moment:

Flight crew member:

May not be null.

Duty:

May not be null.

Leg:

May not be null.

Status:

May not be null.

Remarks (Optional):
Lorem ipsum

Create

- Archivo .hack

Como se indica en los procedimientos comunes, en el método authorise() se controlan varios escenarios para validar la creación de una flightAssignment (create).

- En primer lugar, se verifica que el usuario tenga el rol adecuado (FlightCrewMember). Luego, si la petición es un POST, se comprueba que:
- El id recibido sea igual a 0 (lo que indica que se trata de una creación y no de una actualización). Si se recibe un valor distinto, se rechaza la operación.
- El flightLeg y el flightCrewMember recibidos correspondan a entidades válidas (es decir, que existan en la base de datos). Si alguno no existe, no se autoriza.
- El flightLeg debe cumplir condiciones adicionales: no estar en borrador (isDraftMode), pertenecer a la misma aerolínea que el usuario autenticado y tener una hora de llegada en el futuro. Esto evita asignaciones a vuelos antiguos o que no correspondan a la aerolínea del usuario.
- Finalmente, se valida que el flightCrewMember (si se especifica) pertenezca a la misma aerolínea que el usuario autenticado, evitando la asignación de miembros de otras aerolíneas.

Además hay que comprobar que, el miembro seleccionado esté disponible en el status, que la leg no sea solapada, que la leg no tenga un piloto o copiloto ya. La mayoría de estas validaciones también se comprueban tanto en el update como en el publish.

```
@Override
public void authorize() {
    boolean status;
    boolean validLeg;
    boolean isMember;
    FlightCrewMember member;

    isMember = super.getRequest().getPrincipal().hasRoleOfType(FlightCrewMember.class);
    status = isMember;

    boolean validId = true;
    boolean validFlightCrewMemberId = true;
    if (super.getRequest().getMethod().equals("POST")) {
        int legId = super.getRequest().getData("flightLeg", int.class);
        int flightCrewMemberId = super.getRequest().getData("flightCrewMember", int.class);
        FlightCrewMember flightCrewMember = this.repository.findById(flightCrewMemberId);
        legLeg = this.repository.findById(legId);
        int id = super.getRequest().getData("id", int.class, 0);
        validId = id == 0;

        validFlightCrewMemberId = flightCrewMemberId == 0 || flightCrewMember != null;
        validLeg = legId == 0 || leg != null;
        if (validLeg && leg != null) {
            member = (FlightCrewMember) super.getRequest().getPrincipal().getActiveRole();
            boolean isFuture = MomentHelper.isBefore(MomentHelper.getCurrentMoment(), leg.getScheduleArrival());
            boolean isByAirline = leg.getAirCraft().getAirline().getId() == member.getAirline().getId();
            validLeg = !leg.isDraftMode() && isFuture && isByAirline;
        }
        status = validLeg && validId;
        if (validFlightCrewMemberId && flightCrewMember != null) {
            member = (FlightCrewMember) super.getRequest().getPrincipal().getActiveRole();
            validFlightCrewMemberId = flightCrewMember.getAirline().equals(member.getAirline());
        }
        status = status && validFlightCrewMemberId;
    }
    super.getResponse().setAuthorised(status);
}
```

Unexpected error

We are very sorry! If this unexpected error prevents you from working with our system, please, do contact us and we will try to solve it as soon as possible.

Request:
POST http://localhost:8080/Acme-ANS-C2/flight-crew-member/flight-assignment/create HTTP/1.1

Status:
500 Internal Server Error

Exceptions:
acme.client.helpers.Assert.state(Assert.java:45)
java.lang.AssertionError: Access is not authorised

Return

Update

- Archivo .safe

Como se comenta en los procedimientos comunes, hay que probar a enviar un formulario vacío, seguido atributo por atributo probar con cada valor que se genera en el excel “sample-data” dependiendo del tipo de atributo, por último, hay que enviar un formulario correcto para actualizar un flightAssignment. Hacemos shut down para acabar con el record.

Last update moment:
2025/01/01 00:00

Flight crew member:

May not be null.
Duty:

May not be null.
Leg:

May not be null.
Status:

May not be null.
Remarks (Optional):
Lorem ipsum

Log Update Delete Publish

- Archivo .hack

Como se comenta en los procedimientos comunes, tenemos que intentar modificar un flightAssigment de otro member no publicado, uno ya publicado y uno que no exista, debería de saltar el autorise()).

```
@Override
public void authorize() {
    boolean status;
    boolean validLog;
    int memberId;
    FlightAssignment assignment;
    FlightCrewMember member;

    memberId = super.getRequest().getData("id", int.class);
    assignment = this.repository.findById(memberId);
    member = assignment == null ? null : assignment.getFlightCrewMember();
    status = assignment != null && assignment.getId().equals(member.getId());

    boolean validFlightCrewMemberId = true;
    if (super.getRequest().getMethod().equals("POST") && status) {
        int logId = super.getRequest().getData("flightLog", int.class);
        log = this.repository.findById(logId);
        int flightCrewMemberId = super.getRequest().getData("flightCrewMember", int.class);
        FlightCrewMember flightCrewMember = this.repository.findById(flightCrewMemberId);

        validFlightCrewMemberId = flightCrewMemberId == 0 || flightCrewMember != null;
        validLog = logId == 0 || log != null;
        if (validLog && log != null) {
            boolean isAssigned = assignment.getFlightLog() != null && log.getId() == assignment.getFlightLog().getId();
            boolean isFuture = momentHelper.isBefore(momentHelper.getCurrentMoment(), log.getScheduleDerival());
            boolean isAircraft = log.getAircraft().getId() == member.getAircraft().getId();
            validLog = (log.getId() && isFuture || isAssigned);
        }
        status = status && validLog;

        if (validFlightCrewMemberId && flightCrewMember != null) {
            member = (FlightCrewMember) super.getRequest().getPrincipal().getActiveRole();
            validFlightCrewMemberId = flightCrewMember.getId().equals(member.getId());
        }
        status = status && validFlightCrewMemberId;
    }
    super.getResponse().setAuthorized(status);
}
```

Unexpected error

We are very sorry! If this unexpected error prevents you from working with our system, please, do contact us and we will try to solve it as soon as possible.

Request:
GET http://localhost:8080/Acme-ANS-C2/flight-crew-member/flight-assignment/update?id=247 HTTP/1.1

Status:
500 Internal Server Error

Exceptions:
acme.client.helpers.Assert.state(Assert.java:45)
↳ java.lang.AssertionError: Access is not authorised

Return

Publish

- Archivo .safe

Como se comenta en los procedimientos comunes, hay que probar a enviar un formulario vacío, seguido atributo por atributo probar con cada valor que se genera en el excel “sample-data” dependiendo del tipo de atributo, por último, hay que enviar un formulario correcto para publicar un flightAssignment. En el publish tenemos que probar también lo mismo que en el create.

Last update moment:
2025/01/01 00:00

Flight crew member:

May not be null.
Duty:

May not be null.
Leg:

May not be null.
Status:

May not be null.
Remarks (Optional):
Lorem ipsum

- Archivo .hack

Como se comenta en los procedimientos comunes, tenemos que intentar publicar un flightAssignment de otro manager, uno ya publicado y uno que no exista, debería de saltar el autorise().

```
@Override
public void authorize() {
    boolean status;
    boolean validLog;
    int masterId;
    FlightAssignment assignment;
    FlightCrewmember member;

    masterId = super.getRequest().getData("id", int.class);
    assignment = this.repository.findByIdAssignmentById(masterId);
    member = assignment == null ? null : assignment.getFlightCrewmember();
    status = assignment != null && assignment.isDraftMode() && super.getRequest().getPrincipal().hasRole(member);

    boolean validFlightCrewmemberId = true;
    if (super.getRequest().getHeader("token", String.class) != null) {
        int logId = super.getRequest().getHeader("flightLog", int.class);
        logLog = this.repository.findByIdLog(logId);
        int flightCrewmemberId = super.getRequest().getHeader("flightCrewmember", int.class);
        FlightCrewmember flightCrewmember = this.repository.findByIdFlightCrewmember(flightCrewmemberId);

        validFlightCrewmemberId = flightCrewmemberId == 0 || flightCrewmember != null;
        validLog = logId == 0 || log != null;
        if (validLog && log != null) {
            boolean isCrewmemberAssigned = assignment.getFlightLog() != null && log.getLog() == assignment.getFlightLog().getLogId();
            boolean isMaster = !member.isSuperAdmin() && member.getId() == log.getCreatedBy().getId();
            boolean isSuperAdmin = log.getCreatedBy().getRole().getName() == "SUPERADMIN" && member.getRole().getName() == "SUPERADMIN";
            validLog = (isCrewmemberAssigned || isMaster || isSuperAdmin) && !isCrewmemberAssigned || !isMaster || !isSuperAdmin;
        }
        status = status && validLog;
    }
    if (validFlightCrewmemberId && flightCrewmember != null) {
        member = (FlightCrewmember) super.getRequest().getPrincipal().getAuthentication().getPrincipal();
        validFlightCrewmemberId = flightCrewmember.getId() == member.getId() || member.getId() == 0;
    }
    status = status && validFlightCrewmemberId;
    super.getResponse().setAuthorized(status);
}
```

We are very sorry! If this unexpected error prevents you from working with our system, please, do contact us and we will try to solve it as soon as possible.

Request:
GET http://localhost:8080/Acme-ANS-C2/flight-crew-member/flight-assignment/publishId=258 HTTP/1.1

Status:
500 Internal Server Error

Exceptions:
acme.client.helpers.Assert.state(Assert.java:45)
↳ java.lang.AssertionError: Access is not authorised

Return

Delete

- Archivo .safe

Para el archivo .safe lo único que tenemos que hacer es entrar a una flightAssignemnt y borrarlo. También tenemos que probar a borrar un flightAssignemnt que tiene una activityLog publicada. Hacemos shut down para acabar con el record.

- Archivo .hack

Como se comenta en los procedimientos comunes, tenemos que intentar borrar un flightAssignment de otro manager que este sin publicar, uno ya publicado y uno que no exista, debería de saltar el autorise().

```
@Override
public void authorize() {
    boolean status;
    int masterId;
    FlightAssignment assignment;
    FlightCrewmember member;

    masterId = super.getRequest().getData("id", int.class);
    assignment = this.repository.findByIdAssignmentById(masterId);
    member = assignment == null ? null : assignment.getFlightCrewmember();
    status = assignment != null && assignment.isDraftMode() && super.getRequest().getPrincipal().hasRole(member);

    super.getResponse().setAuthorized(status);
}
```

We are very sorry! If this unexpected error prevents you from working with our system, please, do contact us and we will try to solve it as soon as possible.

Request:
GET http://localhost:8080/Acme-ANS-C2/flight-crew-member/flight-assignment/deleteId=258 HTTP/1.1

Status:
500 Internal Server Error

Exceptions:
acme.client.helpers.Assert.state(Assert.java:45)
↳ java.lang.AssertionError: Access is not authorised

Return

Show y List

- Archivo .safe

En mi caso para el show y el list se ha hecho en una misma prueba. Para el archivo .safe lo único que tenemos que hacer es entrar en el listado tanto de los flightAssignmet publicados que ya han ocurrido de todos como a los propios, y lo mismo con los planificados. Y a su vez en cada listado entrar a uno que este tanto publicado o no publicado. Hacemos shut down para acabar con el record.

- Archivos .hack

Como el link <http://localhost:8080/Acme-ANS-C2/flight-crew-member/flight-assignment/list-completed> lista los flightAssignment publicados de todos los member, habría que probar a ver uno que no este publicado de otro member y esto no debería de ser posible, debería saltar el autorise().

```
@Override
public void authorise() {
    boolean status;
    int masterId;
    FlightAssignment assignment;
    FlightCrewMember member;

    masterId = super.getRequest().getData("id", int.class);
    assignment = this.repository.findFlightAssignmentById(masterId);
    member = assignment == null ? null : assignment.getFlightCrewMember();
    status = assignment != null;

    status = status && (!assignment.isDraftMode() || super.getRequest().getPrincipal().hasRealm(member));
    super.getResponse().setAuthorised(status);
}
```

We are very sorry! If this unexpected error prevents you from working with our system, please, do contact us and we will try to solve it as soon as possible.

Request:
GET http://localhost:8080/Acme-ANS-C2/flight-crew-member/flight-assignment/show?id=257 HTTP/1.1

[Return](#)

Status:
500 Internal Server Error

Exceptions:
acme.client.helpers.Assert.state(Assert.java:45)
↳ java.lang.AssertionError: Access is not authorised

3.2. Testing activityLog

Procedimientos comunes

- Para los update/create/publish .safe, cada atributo se ha probado con los valores necesarios para probar todos los posibles casos. La propiedad IncidentType es un String con un maximo de 50 caracteres, Description es otro String con un maximo de 255 caracteres, y SeverityLevel es un Integer con un rango de 0 a 10. Primero de todo habría que enviar un formulario vacío. Hay que probar los siguientes casos:

BASE
EMPTY
MIN - Δ
MIN
MIN + Δ
MAX - Δ
MAX
MAX + Δ

Registration moment:
2020/09/01 10:05

Incident type:
Lorem ipsum
May not be null.

Description:
Lorem ipsum
May not be null.

Severity level:
123,456
Invalid value.

Update Delete Publish

- Para la mayoría de pruebas .hack se hace uso de la herramienta de desarrollador de Firefox para poder modificar el id oculto de los formularios, siempre probaremos a poner un id de un flightAssignment al que no pertenece el activityLog, de una leg que no exista y de un flightAssignment que no esté publicado. Hay que abrir la herramienta de desarrollador antes de que se inicie el récord, si no se hace una petición GET extra y al hacer el replay de las pruebas da fallo.

```
<h1>Activity log details</h1>
<form id="form">
  <input id="id" name="id" value="285" type="hidden">
  <input id="version" name="version" value="0" type="hidden">
  <input id="_csrf" name="_csrf" value="125b34bd-33ac-40ea-9c47-6ff2bef7492f" type="hidden">
  <div class="form-group">
  <div class="form-group">
  <div class="form-group">
  <div class="form-group">
  <button class="btn btn-dark" type="submit" formmethod="POST" onclick="javascript: form.action =
getAbsolutePath('/flight-crew-member/activity-log/update')">Update</button>
```

Create

- Archivo .safe

Como se detalla en los procedimientos comunes, el método `authorise()` de este servicio valida si un miembro de la tripulación tiene permiso para crear un `ActivityLog` asociado a una asignación de vuelo.

En primer lugar, se comprueba que, si la petición es un POST, el campo `id` recibido sea igual a 0. Esto asegura que la operación corresponde a una creación y no a una actualización. Si el `id` es distinto de 0, la operación se invalida.

Luego, se recupera el `masterId`, que corresponde al identificador de la asignación de vuelo (`FlightAssignment`) a la cual se quiere asociar el registro. Se verifica que:

- La asignación exista en la base de datos.
- El usuario autenticado tenga un "realm" que coincida con el `FlightCrewMember` asociado a esa asignación (es decir, que realmente forme parte de dicha asignación).

Solo si se cumplen ambas condiciones (el `id == 0` y la asignación pertenece al usuario actual), la petición se considera autorizada.

Además, en la propia entidad `ActivityLog`, se incluye una validación adicional: el `registrationMoment` del log debe ser posterior al momento de inicio de la `FlightAssignment`. Esta restricción garantiza coherencia temporal en los registros, evitando que se creen logs con fechas anteriores a la asignación de vuelo.

Registration moment:

2025/01/01 00:00

The log registration moment must occur after the leg finishes.

Incident type:

Lorem ipsum

May not be null.

Description:

Lorem ipsum

May not be null.

Severity level:

123,456

Invalid value.

Create

- Archivo .hack

Tenemos que intentar crear el `activityLog` de otro manager que su `flightAssignment` no este publicada, uno que ya este publicado o uno que no existe, cambiando el `id` oculto. Aunque como en la entidad, en el `create` el `id` se manda como 0, por tanto si se identifica un valor distinto debería de saltar el `authorise()`

```
@Override
public void authorise() {
    boolean status;
    int masterId;
    FlightAssignment assignment;

    boolean validId = true;
    if (super.getRequest().getMethod().equals("POST")) {
        int id = super.getRequest().getData("id", int.class, 0);
        validId = id == 0;
    }

    masterId = super.getRequest().getData("masterId", int.class);
    assignment = this.repository.findFlightAssignmentById(masterId);
    status = assignment != null && super.getRequest().getPrincipal().hasRealm(assignment.getFlightCrewMember());

    super.getResponse().setAuthorised(status && validId);
}
```

We are very sorry! If this unexpected error prevents you from working with our system, please, do contact us and we will try to solve it as soon as possible.

Request:
GET http://localhost:8080/Acme-ANS-C2/flight-crew-member/activity-log/create?id=289 HTTP/1.1

Status:
500 Internal Server Error

Exceptions:
`acme.client.components.models.WorkArea.getData(WorkArea.java:206)`
`↳ java.lang.AssertionError`

Return

Update

- Archivo .safe

Como se comenta en los procedimientos comunes, hay que probar a enviar un formulario vacío, seguido atributo por atributo probar con cada valor que se genera en el excel “sample-data” dependiendo del tipo de atributo, por último, hay que enviar un formulario correcto para actualizar una activityLog.

Registration moment:
2020/09/01 10:05

Incident type:
Lorem ipsum
May not be null.

Description:
Lorem ipsum
May not be null.

Severity level:
123,456
Invalid value.

Update

Delete

Publish

- Archivo .hack

Tenemos que intentar actualizar una activityLog de otro manager que no tenga publicada la flightAssignment, uno que ya este publicado o de uno que no existe, cambiando el id oculto. Esto no está permitido por lo tanto debería de saltar el `authorise ()`.

```
@Override
public void authorise() {
    boolean status;
    int logId;
    FlightCrewMember member;
    ActivityLog log;

    logId = super.getRequest().getData("id", int.class);
    log = this.repository.findById(logId);
    member = log == null ? null : log.getFlightAssignment().getFlightCrewMember();
    status = log != null && log.isDraftNode() && super.getRequest().getPrincipal().hasRealm(member);

    super.getResponse().setAuthorised(status);
}
```

We are very sorry! If this unexpected error prevents you from working with our system, please, do contact us and we will try to solve it as soon as possible.

Request:
GET http://localhost:8080/Acme-ANS-C2/flight-crew-member/activity-log/update?id=289 HTTP/1.1

[Return](#)

Status:
500 Internal Server Error

Exceptions:
`acme.client.helpers.AssertState(Assert.java:45)`
`↳ java.lang.AssertionError: Access is not authorised`

Publish

- Archivo .safe

Como se comenta en los procedimientos comunes, hay que probar a enviar un formulario vacío, seguido atributo por atributo probar con cada valor que se genera en el excel “sample-data” dependiendo del tipo de atributo, por último, hay que enviar un formulario correcto para publicar un activityLog.

Además de lo que se valida en el create, en el publish también se valida que el flightAssignment al cual el activityLog este asociada, este publicada.

Hacemos shut down para acabar con el record.

This log cannot be published as the selected flight assignment has not been published yet.

Registration moment:
2020/09/01 10:07

Incident type:
Lorem ipsum dolor sit amet, consectetur adipiscing

Description:
Lo

Severity level:
9

[Update](#) [Delete](#) [Publish](#)

- Archivo .hack

Tenemos que intentar publicar el activityLog de otro member que no tenga la flightAssignment publicada, una que ya este publicada o una que no existe, cambiando el id oculto. Esto no está permitido por lo tanto debería de saltar el authorise ().

```
@Override
public void authorise() {
    boolean status;
    int logId;
    FlightCrewMember member;
    ActivityLog log;

    logId = super.getRequest().getData("id", int.class);
    log = this.repository.findById(logId);
    member = log == null ? null : log.getFlightAssignment().getFlightCrewMember();
    status = log != null && log.isDraftMode() && super.getRequest().getPrincipal().hasRole(member);

    super.getResponse().setAuthorised(status);
}
```

We are very sorry! If this unexpected error prevents you from working with our system, please, do contact us and we will try to solve it as soon as possible.

Request:
GET http://localhost:8080/Acme-ANS-C2/flight-crew-member/activity-log/publish?id=289 HTTP/1.1

[Return](#)

Status:
500 Internal Server Error

Exceptions:
acme.client.helpers.Assert.state(Assert.java:45)
↳ java.lang.AssertionError: Access is not authorised

Delete

- Archivo .safe

Para el archivo .safe lo único que tenemos que hacer es entrar a un activityLog y borrarlo. Hacemos shut down para acabar con el record.

- Archivo .hack

Como se comenta en los procedimientos comunes, tenemos que intentar borrar un activityLog de otro member que su flightAssignment no este publicada, uno ya publicado y uno que no exista, debería de saltar el authorise ().

```
@Override
public void authorise() {
    boolean status;
    int logId;
    FlightCrewMember member;
    ActivityLog log;

    logId = super.getRequest().getData("id", int.class);
    log = this.repository.findActivityLogById(logId);
    member = log == null ? null : log.getFlightAssignment().getFlightCrewMember();
    status = log != null && log.isDraftMode() && super.getRequest().getPrincipal().hasRealm(member);

    super.getResponse().setAuthorised(status);
}
```

We are very sorry! If this unexpected error prevents you from working with our system, please, do contact us and we will try to solve it as soon as possible.

Request:
GET http://localhost:8080/Acme-ANS-CZ/flight-crew-member/activity-log/delete?id=289 HTTP/1.1

Status:
500 Internal Server Error

Exceptions:
acme.client.helpers.Assert.state(Assert.java:45)
↳ java.lang.AssertionError: Access is not authorised

Return

Show y List

- Archivo .safe

Al igual que para el flightAssignment, he realizado las pruebas tanto del list como del show juntas. Para el archivo .safe lo único que tenemos que hacer es entrar al listado de los activityLog de una flightAssignment y entrar en una de las activityLog. Hacemos shut down para acabar con el record.

- Archivo .hack

Tenemos que probar a entrar en una de las activityLog de un flightAssignment de otro member que no tenga la flightAssignment publicada o de uno que no exista. Esto no está permitido por tanto debería de saltar el authorise ().

```
@Override
public void authorise() {
    boolean status;
    int masterId;
    ActivityLog log;
    FlightCrewMember member;

    masterId = super.getRequest().getData("id", int.class);
    log = this.repository.findActivityLogById(masterId);
    member = log == null ? null : log.getFlightAssignment().getFlightCrewMember();
    status = log != null;

    status = status && (!log.isDraftMode() || super.getRequest().getPrincipal().hasRealm(member));

    super.getResponse().setAuthorised(status);
}
```

We are very sorry! If this unexpected error prevents you from working with our system, please, do contact us and we will try to solve it as soon as possible.

Request:
GET http://localhost:8080/Acme-ANS-CZ/flight-crew-member/activity-log/show?id=2000 HTTP/1.1

Status:
500 Internal Server Error

Exceptions:
acme.client.helpers.Assert.state(Assert.java:45)
↳ java.lang.AssertionError: Access is not authorised

Return

Rendimiento obtenido

El objetivo de este apartado es realizar un análisis del rendimiento obtenido en las pruebas descritas anteriormente.

4.1. Gráficos rendimiento

request-meth	request-path	response-status	time
Promedio /			5.71581087
Promedio /anonymous/system/sign-in			8.931975
Promedio /any/system/welcome			3.54068318
Promedio /authenticated/system/sign-out			4.94248667
Promedio /flight-crew-member/activity-log/create			67.2816977
Promedio /flight-crew-member/activity-log/delete			36.4271571
Promedio /flight-crew-member/activity-log/list			519.447939
Promedio /flight-crew-member/activity-log/publish			40.5331881
Promedio /flight-crew-member/activity-log/show			32.1884174
Promedio /flight-crew-member/activity-log/update			41.6152605
Promedio /flight-crew-member/flight-assignment/create			38.7774391
Promedio /flight-crew-member/flight-assignment/delete			30.4286667
Promedio /flight-crew-member/flight-assignment/list-completed			20.1787167
Promedio /flight-crew-member/flight-assignment/list-mine-completed			18.1833152
Promedio /flight-crew-member/flight-assignment/list-mine-planned			19.0560421
Promedio /flight-crew-member/flight-assignment/list-planned			20.84428
Promedio /flight-crew-member/flight-assignment/publish			43.8996056
Promedio /flight-crew-member/flight-assignment/show			28.5854657
Promedio /flight-crew-member/flight-assignment/update			43.7696811
Promedio general			46.4100195



Figura 1: Promedio de tiempo de cada tipo de petición

Como se puede observar las peticiones más ineficientes son los create y list de activityLog.

El alto tiempo registrado en el endpoint /flight-crew-member/activity-log/list puede deberse a que este método se invoca de forma repetida e implícita durante muchas otras pruebas (por ejemplo, después de crear, actualizar o eliminar registros), acumulando así múltiples ejecuciones.

Además, aunque la operación list no realiza validaciones complejas, los ActivityLog están vinculados a FlightAssignment, lo que puede implicar la carga de relaciones

adicionales (como legs o crewMember). Esto incrementa el coste de la consulta, especialmente si se utilizan estrategias de carga.

También es posible que el volumen de datos devueltos sea elevado debido a la acumulación de logs creados en las pruebas, lo que aumenta el tiempo de serialización y respuesta.

En resumen, el tiempo no refleja necesariamente complejidad funcional, sino carga acumulada, relaciones entre entidades y frecuencia de uso durante los tests.

Por otra parte, el promedio general obtenido ha sido de 46.4100195 milisegundos, lo cual es una cifra buena.

4.2. Datos estadísticos

Before		
Media	89.74383715	
Error típico	10.11891551	
Mediana	46.1044	
Moda	6.9012	
Desviación estándar	275.8216654	
Varianza de la muestra	76077.59112	
Curtosis	40.90169845	
Coeficiente de asimetría	6.424210311	
Rango	2217.405	
Mínimo	2.879	
Máximo	2220.284	
Suma	66679.671	
Cuenta	743	
Nivel de confianza(95.0%)	19.86511339	
Interval (ms)	69.87872375	109.608951
Interval (s)	0.069878724	0.10960895

Figura 2: datos estadísticos obtenidos

Estos son los resultados estadísticos obtenidos en el análisis. El intervalo de confianza se sitúa entre 69.87 y 109,60 ms, que se puede considerar como válido, ya que en el proyecto no se impone ningún requisito relacionado con el rendimiento.

Comparativa rendimiento tras creación índices

En este apartado se realizará una comparativa de rendimiento tras la creación de los índices para optimizar las consultas de la base de datos, utilizando la herramienta del z-analysis de Excel.

Prueba z para medias de dos muestras		
	<i>before</i>	<i>after</i>
Media	89.74383715	46.41001952
Varianza (conocida)	7607759112	2204862893
Observaciones	743	743
Diferencia hipotética de las medias	0	
z	0.011924188	
P(Z<=z) una cola	0.49524305	
Valor crítico de z (una cola)	1.644853627	
Valor crítico de z (dos colas)	0.9904861	
Valor crítico de z (dos colas)	1.959963985	

En cuanto al valor de p-value obtenido es de 0.495243049880565, lo cual significa que no hay una diferencia significativa de rendimiento, esto es de esperar debido a que el volumen de datos que se utiliza en las pruebas es muy pequeño, por lo que es lógico que no se obtenga a penas diferencia en el rendimiento.

Comparativa entre distintos ordenadores

A continuación, se realizará una comparativa entre el rendimiento obtenido en mi ordenador y el ordenador de otro integrante del grupo (después de la creación de índices).



Figura 5: Promedio de tiempos obtenidos en otro ordenador

Compañero	
Media	11.5580773
Error típico	1.18186035
Mediana	5.9598
Moda	1.1327
Desviación estándar	32.2151804
Varianza de la muestra	1037.81785
Curtosis	42.9158913
Coefficiente de asimetría	6.42738037
Rango	323.3847
Mínimo	0.538
Máximo	323.9227
Suma	8587.6514
Cuenta	743
Nivel de confianza(95.0%)	2.32018834

Figura 6: Datos estadísticos obtenidos en otro ordenador

Como en mi ordenador la media obtenida es de 89.74 ms y en el otro es de 11,55 ms, se llega a la conclusión de que el rendimiento el ordenador de mi compañero es bastante mejor (alrededor de un 87.12%).

Conclusión

La elaboración del conjunto completo de pruebas ha resultado de gran utilidad para identificar errores residuales en la aplicación, así como para detectar posibles vectores de ataque que no se habían considerado inicialmente. Este proceso ha permitido reforzar tanto la robustez funcional como la seguridad del sistema.

En lo que respecta a la comparativa de rendimiento, las diferencias observadas tras la incorporación de índices fueron mínimas, lo cual era previsible dado el reducido volumen de datos utilizado durante las pruebas. No obstante, al comparar los resultados con los de otro integrante del grupo, se constató que **el rendimiento en mi equipo fue claramente inferior**, con una media de **89,74 ms** frente a los **11,55 ms** registrados en su máquina. Esto pone de manifiesto que las capacidades del hardware utilizado pueden influir notablemente en los tiempos de respuesta, incluso en entornos de pruebas locales. En este caso, su equipo fue aproximadamente **un 87,12% más rápido** que el mío.

Por otro lado, la cobertura de pruebas alcanzada ha sido muy alta: el módulo FlightAssignment alcanza un **99,6%**, mientras que ActivityLog llega al **99,7%**. Se han ejecutado múltiples pruebas adicionales con el objetivo de alcanzar el 100%, sin embargo, Eclipse sigue marcando algunas líneas en amarillo en el informe de cobertura. Estas líneas amarillas suelen representar **ramas de ejecución parcialmente cubiertas**, es decir, fragmentos de código donde **una parte del if o del else ha sido**

ejecutada, pero no ambas. También puede indicar que en sentencias como `if (...)` se ha ejecutado el bloque `true` pero no el `false`, o viceversa. Aunque el código se ha recorrido, no se han probado todos los posibles caminos lógicos, lo que impide que la cobertura sea considerada total.

Aun así, el alto nivel de cobertura logrado garantiza que la mayor parte del código ha sido validada correctamente, y las posibles ramas no cubiertas tienen un impacto mínimo en el comportamiento general del sistema.

Bibliografía

Intencionalmente en blanco.