

# Diseño y Pruebas II

## Testing Report

**C2.018**

Felipe Solís Agudo (student3)

[felsolagu@alum.us.es](mailto:felsolagu@alum.us.es)

<https://github.com/davidg43/Acme-SF-D>



<b>Tabla de versiones</b>	<b>2</b>
<b>Tabla de revisiones</b>	<b>2</b>
<b>Resumen Ejecutivo</b>	<b>3</b>
<b>Introducción</b>	<b>4</b>
<b>Contenido</b>	<b>5</b>
Testing funcional	5
TrainingModule	6
TrainingSession	25
Rendimiento del testing funcional	44
Performance testing sin índices vs con índices	44
Performance testing con índices vs con índices ordenador de pabespnar	48
<b>Conclusiones</b>	<b>52</b>
<b>Bibliography</b>	<b>52</b>

## Tabla de versiones

Versión	Fecha	Descripción
1.0	27/05/2024	Versión inicial del documento (junio)
1.1	22/06/2024	Testing funcional
1.2	28/06/2024	Rendimiento del testing funcional
2.0	02/07/2024	Versión final

## Tabla de revisiones

N. Revisión	Fecha	Descripción
-------------	-------	-------------

1	27/05/2024	D04 (Julio)
2	20/06/2024	D04 (Julio)
3	22/06/2024	D04 (Julio)
4	28/06/2024	D04 (Julio)
5	29/06/2024	D04 (Julio)
6	02/07/2024	D04 (Julio)
7	06/07/2024	D04 (Julio)

## Resumen Ejecutivo

El objetivo de este informe es proporcionar una descripción detallada de los procedimientos seguidos en el testing formal del proyecto Acme-SF. Se abordan desde la generación de datos de ejemplo hasta la verificación de la correcta implementación de los requisitos funcionales. Además, se incluyen las herramientas estadísticas utilizadas para comparar los tiempos de ejecución entre peticiones y evaluar el rendimiento en diferentes ordenadores , así como la generación de gráficos para visualizar estos resultados.

Este informe se compone de dos secciones principales: testing funcional y rendimiento del testing funcional.

En la sección de testing funcional se presenta un inventario exhaustivo de los casos de prueba implementados, organizados según las distintas características del sistema. Cada caso de prueba contiene una breve descripción y una evaluación de su efectividad para detectar errores.

La sección de análisis de rendimiento muestra gráficos que ilustran el tiempo de respuesta (wall time) del sistema durante la ejecución de las pruebas funcionales en dos ordenadores diferentes. Se comparan los resultados obtenidos con y sin índices, y se calcula un intervalo de confianza del 95% para estos tiempos de respuesta. Esto permite determinar cuál de los dos ordenadores ofrece un mejor rendimiento al procesar las solicitudes del sistema.

En resumen, se ha adoptado un enfoque meticuloso para identificar y resolver errores durante el proceso de testeo, garantizando un producto de alta calidad que cumpla con las expectativas del cliente. Este documento respalda y detalla estos esfuerzos.

# Introducción

En este informe se explica cómo se han realizado el testing funcional y el rendimiento de dicho testing funcional, con el propósito de entregar un proyecto de alta calidad. Para ello, se ejecutan diversas pruebas relacionadas con los requisitos funcionales 6 y 7.

El documento se organiza según el anexo proporcionado en la plataforma de enseñanza virtual. Comienza con una portada que incluye las credenciales del autor, seguida de una tabla de versiones que enumera las modificaciones realizadas, indicando su número y fecha. Luego, se presenta un resumen ejecutivo para contextualizar al lector sobre el contenido del informe, seguido de una introducción que describe brevemente el contenido y estructura del documento.

El trabajo se divide en dos secciones principales: testing funcional y rendimiento del testing funcional. En la primera, se detallan los casos de prueba implementados, organizados por características, y se evalúa su efectividad para detectar errores. En la segunda, se incluyen gráficos relevantes y un intervalo de confianza del 95% para el tiempo de respuesta del proyecto ante las solicitudes funcionales, así como una comparación pertinente. Estos capítulos ofrecen una evaluación completa de la calidad y eficiencia del proyecto en desarrollo. Finalmente, se presenta la conclusión y la bibliografía utilizada.

# Contenido

## Testing funcional:



































































Esta sección se dividirá en dos apartados, enfocados en las entidades “TrainingModule” y “TrainingSession”, donde se hablará de la cobertura obtenida en el proceso de testing funcional, el proceso realizado para realizar las pruebas (pruebas positivas, negativas y de hacking) y obtener dicha cobertura, y posibles mejoras respecto al código en el caso de que fuera posible aumentar la cobertura ya obtenida.

Es necesario mencionar que todas estas pruebas se han realizado en la rama “July-Student3” de manera individual tras haber resuelto los errores en la entrega de junio, el día 22/06/2024. Esta rama contiene los últimos cambios de todos los estudiantes que se presentaron en la entrega final de junio.

Se debe destacar esto ya que durante la entrega de junio de este mismo proyecto, al mover las pruebas funcionales de una rama a otra, hubo clases afectadas y se perdió cobertura. Esto puede volver a ocurrir en esta nueva entrega de julio. A la hora de realizar el merge, dicha cobertura puede verse afectada por cambios de otros compañeros en sus respectivas ramas a la hora de resolver sus errores encontrados en la entrega de junio, que hayan sido resueltos de manera posterior a la realización de mis tests, como por ejemplo la ampliación del csv de la entidad “Project”. Este csv es el que más podría afectar a mis tests y a la cobertura de estos, ya que podría provocar que en los tests guardados como .safe, aparecieran diversos FAILED TO al ejecutar el “tester.replayer”, haciendo parecer que no se han grabado correctamente los tests, **y esto no es así**. Estos tests se han grabado con un csv de Project con proyectos publicados y no publicados, y si el student1 decide ampliar su csv de “Project”, este seguirá teniendo las entradas con las que se realizaron estos tests, **lo cual es suficiente para comprobar la funcionalidad de mi parte individual**.



Además, a nivel de código, la rama “July-Student3” contendrá exactamente las mismas clases que el archivo correspondiente a la entrega final. **Por ende, esto solo afectará a la cobertura, no a la funcionalidad, ni a los resultados de la ejecución del “tester.analyser”**.

Es necesario mencionar también que, a la hora de ejecutar dichos tests, aparecen numerosos FAILED TO debidos a los errores controlados del “payload” de los banners.

Element	Coverage	Covered Instru...	Missed Instru...	Total Instructio...
>  Acme-Framework-24.5.0	 47,2 %	14.377	16.087	30.464
▼  Acme-SF-D04	 30,5 %	5.117	11.641	16.758
▼  src/main/java	 30,5 %	5.117	11.641	16.758
>  acme.features.auditor.codeAudit	 3,8 %	60	1.528	1.588
>  acme.features.auditor.auditRecord	 4,1 %	59	1.377	1.436
>  acme.features.manager.project	 4,4 %	59	1.267	1.326
>  acme.features.client.contract	 5,0 %	59	1.118	1.177
>  acme.features.client.progresslog	 5,8 %	59	959	1.018
>  acme.features.manager.userStory	 6,4 %	59	870	929
>  acme.features.manager.assignment	 6,5 %	49	710	759
>  acme.features.administrator.banner	 6,5 %	49	707	756
>  acme.features.any.claim	 6,4 %	41	595	636
>  acme.forms	 0,0 %	0	330	330
>  acme.features.authenticated.developer	 7,6 %	22	267	289
>  acme.features.authenticated.auditor	 8,8 %	22	228	250
>  acme.features.authenticated.provider	 9,6 %	22	206	228
>  acme.features.authenticated.consumer	 10,1 %	22	196	218
>  acme.features.developer.dashboard	 6,5 %	12	173	185
>  acme.features.auditor.dashboard	 7,2 %	13	167	180
>  acme.features.manager.dashboard	 8,2 %	13	146	159
>  acme.entities	 58,5 %	199	141	340
>  acme.features.client.dashboard	 9,6 %	12	113	125
>  acme.entities.project	 57,5 %	134	99	233
>  acme.features.developer.trainingSessions	 96,3 %	1.993	77	2.070
>  acme.features.developer.trainingModule	 95,1 %	1.471	75	1.546
>  acme.datatypes	 4,2 %	3	68	71
>  acme.roles	 66,3 %	130	66	196
>  acme.entities.codeAudit	 76,1 %	162	51	213
>  acme.entities.contract	 59,3 %	70	48	118
>  acme.entities.sponsorShip	 66,7 %	90	45	135
>  acme.components	 86,0 %	49	8	57
>  acme.systemConfiguration	 64,7 %	11	6	17
>  acme.entities.trainingModule	 100,0 %	173	0	173

## TrainingModule:

















En el proceso de pruebas de la entidad “TrainingModule” se ha logrado alcanzar una cobertura del 95,1%. Esto es un indicador muy positivo del nivel de exhaustividad de nuestras pruebas. Esta cobertura se traduce en que, de un total de 1546 instrucciones que componen esta entidad, 1471 han sido cubiertas por los casos de prueba, dejando solo 75 instrucciones sin cubrir.

▼  acme.features.developer.trainingModule  95,1 % 1.471 75 1.546

Aunque la cobertura es un excelente indicador, es fundamental recordar que el objetivo no es alcanzar el 100% de cobertura, sino asegurarse de que las áreas más

críticas y propensas a errores del código estén bien cubiertas, pudiéndose asegurar que esto se ha logrado con éxito.

A continuación, se presenta en detalle la cobertura de pruebas de cada característica implementada para esta entidad.

▼  acme.features.developer.trainingModule	 95,1 %	1.471	75	1.546
>  DeveloperTrainingModuleUpdateService.java	 95,0 %	342	18	360
>  DeveloperTrainingModuleCreateService.java	 93,8 %	242	16	258
>  DeveloperTrainingModuleDeleteService.java	 94,1 %	255	16	271
>  DeveloperTrainingModulePublishService.java	 95,7 %	356	16	372
>  DeveloperTrainingModuleListService.java	 95,0 %	96	5	101
>  DeveloperTrainingModuleShowService.java	 97,3 %	145	4	149
>  DeveloperTrainingModuleController.java	 100,0 %	35	0	35

Es importante señalar que todos los tests muestran líneas amarillas correspondientes a “assert object !=null”, lo cual es completamente normal. Si estas líneas se hubieran cubierto, significaría que el sistema habría encontrado un fallo irreversible durante las pruebas.

Respecto a los casos de pruebas positivos y negativos, se han probado realizando los pasos que realizaría un usuario estándar logueado como “**developer**” al utilizar la aplicación:

List.safe	<ul style="list-style-type: none"> <li>- Listar módulos de entrenamiento de un desarrollador (además de un listado vacío).</li> </ul>
Show.safe	<ul style="list-style-type: none"> <li>- Mostrar los detalles de los módulos de entrenamiento (publicado o sin publicar).</li> </ul>
Delete.safe	<ul style="list-style-type: none"> <li>- Eliminar módulos de entrenamiento no publicados.</li> <li>- Comprobar restricciones a la hora de borrar un módulo de entrenamiento (no se puede borrar si tiene una sesión de entrenamiento publicada asociada).</li> </ul>

	<ul style="list-style-type: none"> <li>- Comprobar que se ha eliminado correctamente.</li> </ul>
Create.safe	<ul style="list-style-type: none"> <li>- Crear módulos de entrenamiento, probando los campos del formulario con todo tipo de valores (además de con valores positivos, también se han probado valores negativos rechazados por el sistema, como valores nulos, repetidos, duplicados, URLs incorrectas,... comprobando los límites establecidos para estos).</li> <li>- Comprobar que se ha creado correctamente.</li> <li>- Comprobar que podemos ver sus detalles.</li> <li>- Comprobar que el atributo "creationMoment" se ha generado correctamente.</li> </ul>
Update.safe	<ul style="list-style-type: none"> <li>- Actualizar módulos de entrenamiento no publicados, probando los campos del formulario con todo tipo de valores (además de con valores positivos, también se han probado valores negativos rechazados por el sistema, como valores nulos, repetidos, duplicados, URLs incorrectas,... comprobando los límites establecidos para estos).</li> <li>- Comprobar que se ha actualizado correctamente.</li> <li>- Comprobar que podemos ver sus detalles.</li> </ul>



	<ul style="list-style-type: none"> <li>- Comprobar que el atributo "updateMoment" se ha generado correctamente.</li> </ul>
Publish.safe	<ul style="list-style-type: none"> <li>- Publicar módulos de entrenamiento no publicados, probando los campos del formulario con todo tipo de valores (además de con valores positivos, también se han probado valores negativos rechazados por el sistema, como valores nulos, repetidos, duplicados, URLs incorrectas,... comprobando los límites establecidos para estos).</li> <li>- Comprobar restricciones a la hora de publicar un módulo de entrenamiento (un módulo de entrenamiento no puede ser publicado si no tiene ninguna sesión de entrenamiento asociada publicada, y tampoco puede ser publicado si tiene asociada alguna sesión de entrenamiento sin publicar).</li> <li>- Comprobar que se ha publicado correctamente.</li> <li>- Comprobar que podemos ver sus detalles.</li> <li>- Comprobar que se ha actualizado el atributo "updateMoment" (publicar consiste en cambiar el valor del atributo "draftMode" de true a false, lo cual es una actualización de la entidad).</li> </ul>

Además, se debe destacar que se han realizado pruebas de hacking mediante la inserción de URLs para acceder sin permisos a funcionalidades concretas de un “developer” siendo otro rol, cualquier usuario o anónimo, acceder a las funcionalidades o elementos de otro “developer” siendo “developer” o a elementos inexistentes, y mediante el uso de la consola de comandos para introducir valores no válidos o erróneos por pantalla, y ver cómo el sistema respondía ante ellos.

Estas pruebas se pueden desglosar de la siguiente manera:

List.hack	<ul style="list-style-type: none"><li>- Tratar de acceder a un listado de módulos de entrenamiento siendo cualquier otro rol que no sea el de “developer” o sin estar logueado.</li></ul>
Show.hack	<ul style="list-style-type: none"><li>- Tratar de acceder a los detalles de un módulo de entrenamiento desde cualquier rol que no sea el de “developer” o sin estar logueado.</li><li>- Tratar de mostrar un módulo de entrenamiento inexistente.</li><li>- Tratar de mostrar un módulo de entrenamiento de otro “developer” (estando logueado como “developer”).</li></ul>
Delete.hack	<ul style="list-style-type: none"><li>- Tratar de acceder a esta funcionalidad (eliminar módulos de entrenamiento) sin estar logueado o estando logueado con otro rol que no sea el de “developer”.</li><li>- Tratar de eliminar un módulo de entrenamiento que no exista.</li><li>- Tratar de eliminar un módulo de entrenamiento que esté publicado.</li></ul>

	<ul style="list-style-type: none"> <li>- Tratar de eliminar un módulo de entrenamiento de otro “developer” (estando logueado como “developer”).</li> </ul>
Create.hack	<ul style="list-style-type: none"> <li>- Tratar de acceder a esta funcionalidad sin estar logueado o estando logueado con otro rol que no sea el de “developer”.</li> <li>- Tratar de crear un módulo de entrenamiento asociándolo a un proyecto no publicado (por consola de comandos, cambiando la id).</li> </ul>
Update.hack	<ul style="list-style-type: none"> <li>- Tratar de acceder a esta funcionalidad sin estar logueado o estando logueado con otro rol que no sea el de “developer”.</li> <li>- Tratar de actualizar un módulo de entrenamiento asociándolo a un proyecto no publicado (por consola de comandos, cambiando la id).</li> <li>- Tratar de actualizar un módulo de entrenamiento que no existe.</li> <li>- Tratar de actualizar un módulo de entrenamiento que ya está publicado.</li> <li>- Tratar de actualizar un módulo de entrenamiento que pertenezca a otro “developer”.</li> <li>- Tratar de modificar por consola de comandos los valores de “creationMoment” y “updateMoment”, que son valores que se encuentran en “readOnly” y actualizar el módulo de entrenamiento.</li> </ul>

Publish.hack	<ul style="list-style-type: none"> <li>- Tratar de acceder a esta funcionalidad sin estar logueado o estando logueado con otro rol que no sea el de “developer”.</li> <li>- Tratar de actualizar un módulo de entrenamiento asociándolo a un proyecto no publicado (por consola de comandos, cambiando la id).</li> <li>- Tratar de publicar un módulo de entrenamiento que no existe.</li> <li>- Tratar de publicar un módulo de entrenamiento que ya está publicado.</li> <li>- Tratar de publicar un módulo de entrenamiento que pertenezca a otro “developer”.</li> <li>- Tratar de modificar por consola de comandos los valores de “creationMoment” y “updateMoment”, que son valores que se encuentran en “readOnly” y publicar el módulo de entrenamiento.</li> </ul>
--------------	---

Por ende, en varias pruebas de hacking, a la hora de reproducir los tests, nos encontramos con errores como:

“Expected 'status' to be '500', but got '200'. Expected 'payload' to be '{banner=78}', but got '{banner=78, difficultyLevelOptions=...”

“Expected 'status' to be '500', but got '200'. Expected 'payload' to be '{banner=80}', but got '{banner=80, code=AFT-143,...”

**Esto no indica que se haya realizado una prueba de hacking con éxito.** Como se ha comentado anteriormente, se han realizado pruebas utilizando la consola de comandos, y para probarlas se ha accedido de manera lícita a algunas funcionalidades.

Estos errores vinieron dados por:

- A la hora de crear un módulo de entrenamiento nuevo, intentar asociarlo a un proyecto no publicado por consola. Para esta prueba de hacking se accedió de manera lícita a la funcionalidad “create” de training module, y desde ahí se intentó cambiar el proyecto disponible por consola de comandos.
- Intentar publicar un módulo de entrenamiento asociado a un proyecto no publicado . Para esto, se accedió a la funcionalidad “show” de manera lícita, para cambiar la URL desde ahí.

Es decir, este tipo de error no indica en ninguno de los casos y ocasiones que aparece que se consiguió romper la funcionalidad del sistema.

### DeveloperTrainingModuleListService.java:

```
@Service
public class DeveloperTrainingModuleListService extends AbstractService<Developer, TrainingModule> {

    // Internal state -----

    @Autowired
    private DeveloperTrainingModuleRepository repository;

    // AbstractService interface -----

    @Override
    public void authorise() {
        super.getResponse().setAuthorised(super.getRequest().getPrincipal().hasRole(Developer.class));
    }

    @Override
    public void load() {
        Collection<TrainingModule> objects;
        int developerId;

        developerId = super.getRequest().getPrincipal().getActiveRoleId();
        objects = this.repository.findManyTrainingModuleByDeveloperId(developerId);

        super.getBuffer().addData(objects);
    }

    @Override
    public void unbind(final TrainingModule object) {
        assert object != null;

        Collection<Project> projects = this.repository.findAllProjects();
        SelectChoices projectsChoices = SelectChoices.from(projects, "code", object.getProject());

        final Dataset dataset = super.unbind(object, "code", "project.title");
        dataset.put("project", projectsChoices.getSelected().getLabel());
        dataset.put("projects", projectsChoices);

        if (object.getDraftMode())
```

```


@Override
public void unbind(final TrainingModule object) {
    assert object != null;

    Collection<Project> projects = this.repository.findAllProjects();
    SelectChoices projectsChoices = SelectChoices.from(projects, "code", object.getProject());

    final Dataset dataset = super.unbind(object, "code", "project.title");
    dataset.put("project", projectsChoices.getSelected().getLabel());
    dataset.put("projects", projectsChoices);

    if (object.getDraftMode())
        dataset.put("draftMode", "No");
    else {
        final Locale local = super.getRequest().getLocale();
        dataset.put("draftMode", local.equals(Locale.ENGLISH) ? "Yes" : "Si");
    }

    super.getResponse().addData(dataset);
}
}

>  DeveloperTrainingModuleListService.java 95,0 % 96 5 101

```

En esta primera característica observamos que todas las líneas de la clase han sido cubiertas por completo, salvo la línea “dataset.put(“draftMode”, local.equals(Locale.ENGLISH) ? “Yes” : “Si”);”, debido a que la aplicación, a la hora de ejecutar los tests, sólo se puede probar en inglés, no en castellano. Por ende, esta línea se ejecuta parcialmente. Ha conseguido pasar todas las pruebas con éxito, se ha obtenido un alto porcentaje de cobertura y no se encontró ningún fallo respecto a esta característica durante el desarrollo de los casos de prueba.

Por lo tanto, respecto a la efectividad de los tests a la hora de encontrar fallos o vulnerabilidades, es necesario decir que para este servicio fue baja, no se encontraron.

## DeveloperTrainingModuleShowService.java:

```
@Service
public class DeveloperTrainingModuleShowService extends AbstractService<Developer, TrainingModule> {

    // Internal state -----

    @Autowired
    private DeveloperTrainingModuleRepository repository;

    // AbstractService interface -----

    @Override
    public void authorise() {
        boolean status;
        int masterId;
        TrainingModule trainingModule;
        Developer developer;

        masterId = super.getRequest().getData("id", int.class);
        trainingModule = this.repository.findOneTrainingModuleById(masterId);
        developer = trainingModule == null ? null : trainingModule.getDeveloper();
        status = trainingModule != null && //
            super.getRequest().getPrincipal().hasRole(developer) && //
            trainingModule.getDeveloper().getId() == super.getRequest().getPrincipal().getActiveRoleId();

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

    @Override
    public void load() {
        TrainingModule object;
        int id;

        id = super.getRequest().getData("id", int.class);
        object = this.repository.findOneTrainingModuleById(id);

        super.getBuffer().addData(object);
    }



    @Override
    public void unbind(final TrainingModule object) {
        assert object != null;

        Dataset dataset;
        SelectChoices choices;
        final Collection<Project> projects = this.repository.findAllProjectsPublished();

        SelectChoices choicesDifficultyLevel = SelectChoices.from(DifficultyLevel.class, object.getDifficultyLevel());

        choices = SelectChoices.from(projects, "code", object.getProject());

        dataset = super.unbind(object, "code", "creationMoment", "details", "difficultyLevel", "updateMoment", "link", "totalTime", "draftMode");
        dataset.put("project", choices.getSelected().getKey());
        dataset.put("projects", choices);
        dataset.put("difficultyLevelOptions", choicesDifficultyLevel);
        super.getResponse().addData(dataset);
    }
}
```

>  DeveloperTrainingModuleShowService.java	 97,3 %	145	4	149
---	--	-----	---	-----

En esta clase observamos que todas las líneas de código han sido cubiertas exitosamente, salvo una de las condiciones en el método “authorise”, la cual se ejecutó parcialmente debido a que no fue completamente necesaria a la hora de realizar los test de hacking. Es necesario mencionar que pese a ejecutarse parcialmente dicha línea, no se pudo “romper” este servicio bajo ninguna prueba de hacking utilizada. La línea se ha dejado intacta en el código ya que, mediante el uso

de otra URL, podría ser necesaria para garantizar la seguridad de este servicio. Se ha obtenido un alto porcentaje de cobertura por lo que no se ha considerado un gran problema que esa línea no estuviera cubierta al completo, pues no es una línea roja. No se encontraron fallos probando este servicio. Por lo tanto, respecto a la efectividad de los tests a la hora de encontrar fallos o vulnerabilidades, es necesario decir que para este servicio fue baja, no se encontraron.

### DeveloperTrainingModuleCreateService.java:

```
@Service
public class DeveloperTrainingModuleCreateService extends AbstractService<Developer, TrainingModule> {

    // Internal state -----

    @Autowired
    private DeveloperTrainingModuleRepository repository;

    // AbstractService interface -----

    @Override
    public void authorise() {
        super.getResponse().setAuthorised(true);
    }

    @Override
    public void load() {
        final int id = super.getRequest().getPrincipal().getActiveRoleId();

        TrainingModule object = new TrainingModule();
        object.setDeveloper(this.repository.findOneDeveloperById(id));
        object.setDraftMode(true);

        super.getBuffer().addData(object);
    }
}
```



```

@Override
public void bind(final TrainingModule object) {
    assert object != null;

    int projectId = super.getRequest().getData("project", int.class);
    Project project = this.repository.findOneProjectById(projectId);

    //quitamos creationMoment y updateMoment
    super.bind(object, "code", "details", "difficultyLevel", "link", "totalTime", "project");

    Date currentMoment = MomentHelper.getCurrentMoment();

    Date creationMoment = new Date(currentMoment.getTime() - 1000000);

    object.setCreationMoment(creationMoment);
    object.setProject(project);
}

@Override
public void validate(final TrainingModule object) {
    assert object != null;

    if (!super.getBuffer().getErrors().hasErrors("code")) {
        final boolean duplicatedCode = this.repository.findAllTrainingModules().stream().anyMatch(e -> e.getCode().equals(object.getCode()));

        super.state(!duplicatedCode, "code", "developer.trainingModule.form.error.duplicated-code");
    }

    if (!super.getBuffer().getErrors().hasErrors("project")) {
        Project project = object.getProject();

        super.state(!project.isDraft(), "project", "developer.trainingModule.form.error.invalid-project");
    }
}

@Override
public void perform(final TrainingModule object) {
    assert object != null;

    this.repository.save(object);
}

@Override
public void unbind(final TrainingModule object) {
    assert object != null;

    Collection<Project> projects = this.repository.findAllProjectsPublished();
    SelectChoices projectsChoices = SelectChoices.from(projects, "code", object.getProject());

    SelectChoices choices = SelectChoices.from(DifficultyLevel.class, object.getDifficultyLevel());

    Dataset dataset = super.unbind(object, "code", "creationMoment", "details", "difficultyLevel", "updateMoment", "link", "totalTime", "project", "draftMode");

    dataset.put("project", projectsChoices.getSelected().getKey());
    dataset.put("projects", projectsChoices);
    dataset.put("difficultyLevelOptions", choices);

    super.getResponse().addData(dataset);
}
}

```

>  DeveloperTrainingModuleCreateService.java  93,8 % 242 16 258

En esta característica observamos que todas las líneas de la clase han sido cubiertas por completo y se ha obtenido un alto porcentaje de cobertura. Ha conseguido pasar todas las pruebas con éxito. Sin embargo, mediante la realización de esta prueba se pudo detectar un fallo. Realizando el test de hacking, se comprobó que se podía introducir por consola la id de un proyecto no publicado y crear un módulo de entrenamiento asociado a este, cosa que no tiene sentido. Esto se solucionó añadiendo una restricción en el método “validate”.

Por lo tanto, respecto a la efectividad de los tests a la hora de encontrar fallos o vulnerabilidades, es necesario decir que para este servicio fue alta, ya que si se encontraron errores.

## DeveloperTrainingModuleUpdateService.java:

```
@Service
public class DeveloperTrainingModuleUpdateService extends AbstractService<Developer, TrainingModule> {

    // Internal state -----

    @Autowired
    private DeveloperTrainingModuleRepository repository;

    // AbstractService interface -----

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

        final Principal principal = super.getRequest().getPrincipal();
        final int userAccountId = principal.getAccountId();

        masterId = super.getRequest().getData("id", int.class);
        TrainingModule trainingModule = this.repository.findOneTrainingModuleById(masterId);
        Developer developer = trainingModule == null ? null : trainingModule.getDeveloper();
        status = trainingModule != null && trainingModule.getDraftMode() && principal.hasRole(developer) && trainingModule.getDeveloper().getUserAccount().getId() == userAccountId;

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

    @Override
    public void load() {
        int id = super.getRequest().getData("id", int.class);

        TrainingModule object = this.repository.findOneTrainingModuleById(id);

        super.getBuffer().addData(object);
    }

    @Override
    public void bind(final TrainingModule object) {
        assert object != null;

        int projectId = super.getRequest().getData("project", int.class);
        Project project = this.repository.findOneProjectById(projectId);

        Date currentMoment = MomentHelper.getCurrentMoment();
        Date updateMoment = new Date(currentMoment.getTime() - 1);

        super.bind(object, "code", "details", "difficultyLevel", "link", "totalTime", "project");

        TrainingModule original = this.repository.findOneTrainingModuleById(object.getId());
        object.setCreationMoment(original.getCreationMoment());
        object.setUpdateMoment(updateMoment);
        object.setProject(project);
    }

    @Override
    public void validate(final TrainingModule object) {
        assert object != null;

        final String UPDATE_MOMENT = "updateMoment";

        if (!super.getBuffer().getErrors().hasErrors(UPDATE_MOMENT)) {
            final boolean startBeforeEnd = MomentHelper.isAfter(object.getUpdateMoment(), object.getCreationMoment());
            super.state(startBeforeEnd, UPDATE_MOMENT, "developer.trainingModule.form.error.end-before-start");
        }

        if (!super.getBuffer().getErrors().hasErrors("code")) {
            final int trainingModuleId = super.getRequest().getData("id", int.class);
            final boolean duplicatedCode = this.repository.findAllTrainingModules().stream().filter(e -> e.getId() != trainingModuleId).anyMatch(e -> e.getCode().equals(object.getCode()));

            super.state(!duplicatedCode, "code", "developer.trainingModule.form.error.duplicated-code");
        }

        if (!super.getBuffer().getErrors().hasErrors("project")) {
            Project project = object.getProject();

            super.state(!project.isDraft(), "project", "developer.trainingModule.form.error.invalid-project");
        }
    }
}
```

```

@Override
public void perform(final TrainingModule object) {
    assert object != null;

    Date currentMoment = MomentHelper.getCurrentMoment();
    Date updateMoment = new Date(currentMoment.getTime() - 10000);
    object.setUpdateMoment(updateMoment);

    this.repository.save(object);
}

@Override
public void unbind(final TrainingModule object) {
    assert object != null;

    SelectChoices choices = SelectChoices.from(DifficultyLevel.class, object.getDifficultyLevel());

    Collection<Project> projects = this.repository.findAllProjectsPublished();
    SelectChoices projectsChoices = SelectChoices.from(projects, "code", object.getProject());



    Dataset dataset = super.unbind(object, "code", "details", "difficultyLevel", "link", "totalTime", "draftMode", "project");

    dataset.put("creationMoment", object.getCreationMoment());

    dataset.put("difficultyLevelOptions", choices);
    dataset.put("project", projectsChoices.getSelected().getKey());
    dataset.put("projects", projectsChoices);

    super.getResponse().addData(dataset);
}
}

```

>  DeveloperTrainingModuleUpdateService.java  95,0 % 342 18 360

En esta característica observamos que todas las líneas de la clase han sido cubiertas por completo, salvo varias condiciones en el método “authorise”, la cual se ejecutó parcialmente debido a que no fue completamente necesaria a la hora de realizar los test de hacking. Es necesario mencionar que pese a ejecutarse parcialmente dichas líneas, no se pudo “romper” este servicio bajo ninguna prueba de hacking utilizada. Estas líneas se han dejado intactas en el código ya que, mediante el uso de otra URL, podría ser necesaria para garantizar la seguridad de este servicio. Se ha obtenido un alto porcentaje de cobertura por lo que no se ha considerado un gran problema que esa línea no estuviera cubierta al completo, pues no es una línea roja.

Además, tampoco se ejecutó completamente una de las líneas en el método “validate”. Esta línea comprueba que el atributo updateMoment no contenga errores. Si no los contiene, se comprueba que el updateMoment es posterior al creationMoment del módulo de entrenamiento. Dicha línea se ejecuta parcialmente debido a que, a la hora de crear un trainingModule, inicialmente este no tiene un “updateMoment”, por lo que no podría comprobar si contiene errores. Por otro lado, en las pruebas de hacking se ha modificado el “updateMoment” por consola para que este contenga errores, por lo que en algunos casos no cumple esta condición y tampoco ejecutaría dicha línea.

Ha conseguido pasar todas las pruebas con éxito. Sin embargo, mediante la realización de esta prueba se pudo detectar un fallo. Realizando el test de hacking,

al igual que la clase “DeveloperTrainingModuleCreateService”, se comprobó que se podía introducir por consola la id de un proyecto no publicado y actualizar un módulo de entrenamiento asociado a este, cosa que no tiene sentido. Esto se solucionó añadiendo una restricción en el método “validate”.

Por lo tanto, respecto a la efectividad de los tests a la hora de encontrar fallos o vulnerabilidades, es necesario decir que para este servicio fue alta, ya que si se encontraron errores.

## DeveloperTrainingModuleDeleteService.java:

```
@Service
public class DeveloperTrainingModuleDeleteService extends AbstractService<Developer, TrainingModule> {

    // Internal state -----

    @Autowired
    private DeveloperTrainingModuleRepository repository;

    // AbstractService interface -----

    @Override
    public void authorise() {
        boolean status;
        int masterId;
        TrainingModule trainingModule;
        Developer developer;

        masterId = super.getRequest().getData("id", int.class);
        trainingModule = this.repository.findOneTrainingModuleById(masterId);
        developer = trainingModule == null ? null : trainingModule.getDeveloper();
        status = trainingModule != null && trainingModule.getDraftMode() && //
            super.getRequest().getPrincipal().hasRole(developer) && //
            trainingModule.getDeveloper().getId() == super.getRequest().getPrincipal().getActiveRoleId();

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

    @Override
    public void load() {
        TrainingModule object;
        int id;

        id = super.getRequest().getData("id", int.class);
        object = this.repository.findOneTrainingModuleById(id);

        super.getBuffer().addData(object);
    }
}
```

```

@Override
public void bind(final TrainingModule object) {
    assert object != null;

    int projectId;
    Project project;

    projectId = super.getRequest().getData("project", int.class);
    project = this.repository.findOneProjectById(projectId);

    super.bind(object, "code", "creationMoment", "details", "difficultyLevel", "updateMoment", "link");
    object.setProject(project);
}

@Override
public void validate(final TrainingModule object) {
    assert object != null;

    int masterId = super.getRequest().getData("id", int.class);
    List<TrainingSession> ls = this.repository.findManyTrainingSessionsByTrainingModuleId(masterId).stream().toList();
    final boolean thereAreDraftedTrainingSessions = ls.stream().allMatch(Session -> Session.getIsDraftMode());
    super.state(thereAreDraftedTrainingSessions, "*", "developer.trainingModule.form.error.trainingSession-NoDraft");
}

@Override
public void perform(final TrainingModule object) {
    assert object != null;

    Collection<TrainingSession> sessions;
    sessions = this.repository.findManyTrainingSessionsByTrainingModuleId(object.getId());
    this.repository.deleteAll(sessions);
    this.repository.delete(object);
}

@Override
public void unbind(final TrainingModule object) {
    assert object != null;

    Collection<Project> projects = this.repository.findAllProjects();
    SelectChoices projectsChoices = SelectChoices.from(projects, "code", object.getProject());
    SelectChoices choices = SelectChoices.from(DifficultyLevel.class, object.getDifficultyLevel());
    Dataset dataset = super.unbind(object, "code", "creationMoment", "details", "difficultyLevel", "updateMoment", "link", "totalTime", "project", "draftMode");
    dataset.put("difficultyLevelOptions", choices);
    dataset.put("project", projectsChoices.getSelected().getKey());
    dataset.put("projects", projectsChoices);
    super.getResponse().addData(dataset);
}

```

> [DeveloperTrainingModuleDeleteService.java](#) 94,1 % 255 16 271

En esta característica observamos que todas las líneas de la clase han sido cubiertas por completo, salvo unas condiciones en el método “authorise”, la cuales se ejecutaron parcialmente debido a que no fue completamente necesaria a la hora de realizar los test de hacking. Es necesario mencionar que pese a ejecutarse parcialmente dichas líneas, no se pudo “romper” este servicio bajo ninguna prueba de hacking utilizada. Estas líneas se han dejado intactas en el código ya que, mediante él uso de otra URL, podría ser necesaria para garantizar la seguridad de este servicio. Se ha obtenido un alto porcentaje de cobertura por lo que no se ha considerado un gran problema que esa línea no estuviera cubierta al completo, pues no es una línea roja. Ha conseguido pasar todas las pruebas con éxito. Sin embargo, mediante la realización de esta prueba se pudo detectar un fallo. Realizando los tests positivos y negativos, se comprobó que se podía borrar un módulo de entrenamiento con sesiones de entrenamiento ya publicadas, cosa que

no tiene sentido. Esto se solucionó añadiendo una restricción en el método “validate”.

Por lo tanto, respecto a la efectividad de los tests a la hora de encontrar fallos o vulnerabilidades, es necesario decir que para este servicio fue alta, ya que si se encontraron errores.

## DeveloperTrainingModulePublishService.java:

```
@Service
public class DeveloperTrainingModulePublishService extends AbstractService<Developer, TrainingModule> {

    // Internal state -----

    @Autowired
    private DeveloperTrainingModuleRepository repository;

    // AbstractService interface -----

    @Override
    public void authorise() {
        final Principal principal = super.getRequest().getPrincipal();
        final int userAccountId = principal.getAccountId();

        int masterId = super.getRequest().getData("id", int.class);
        TrainingModule trainingModule = this.repository.findOneTrainingModuleById(masterId);

        boolean status = trainingModule != null && trainingModule.getDraftMode() && trainingModule.getDeveloper().getUserAccount().getId() == userAccountId;

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

    @Override
    public void load() {
        int id = super.getRequest().getData("id", int.class);

        TrainingModule object = this.repository.findOneTrainingModuleById(id);

        super.getBuffer().addData(object);
    }

    @Override
    public void bind(final TrainingModule object) {}
    assert object != null;

    int projectId = super.getRequest().getData("project", int.class);
    Project project = this.repository.findOneProjectById(projectId);

    Date currentMoment = MomentHelper.getCurrentMoment();
    Date updateMoment = new Date(currentMoment.getTime() - 10000);

    super.bind(object, "code", "details", "difficultyLevel", "link", "totalTime", "project");
    TrainingModule original = this.repository.findOneTrainingModuleById(object.getId());
    object.setCreationMoment(original.getCreationMoment());
    object.setUpdateMoment(updateMoment);
    object.setProject(project);
}

    @Override
    public void validate(final TrainingModule object) {
        assert object != null;

        if (!super.getBuffer().getErrors().hasErrors("code")) {
            final int trainingModuleId = super.getRequest().getData("id", int.class);
            final boolean duplicatedCode = this.repository.findAllTrainingModules().stream().filter(e -> e.getId() != trainingModuleId).anyMatch(e -> e.getCode().equals(object.getCode()));

            super.state(1duplicatedCode, "code", "developer.trainingModule.form.error.duplicated-code");
        }

        if (!super.getBuffer().getErrors().hasErrors("project")) {
            Project project = object.getProject();

            super.state(!project.isDraft(), "project", "developer.trainingModule.form.error.invalid-project");
        }

        int masterId = super.getRequest().getData("id", int.class);
        List<TrainingSession> ls = this.repository.findManyTrainingSessionsByTrainingModuleId(masterId).stream().toList();
        final boolean someDraftTrainingSession = ls.stream().anyMatch(Session -> Session.isDraftMode());
        final boolean noSession = ls.isEmpty();
        super.state(!noSession, "", "developer.trainingModule.form.error.trainingSession-empty");
        super.state(!someDraftTrainingSession, "", "developer.trainingModule.form.error.trainingSession-draft");
    }
}
```

```

@Override
public void perform(final TrainingModule object) {
    assert object != null;

    object.setDraftMode(false);

    this.repository.save(object);
}

@Override
public void unbind(final TrainingModule object) {
    assert object != null;

    SelectChoices choices = SelectChoices.from(DifficultyLevel.class, object.getDifficultyLevel());

    Collection<Project> projects = this.repository.findAllProjectsPublished();
    SelectChoices projectsChoices = SelectChoices.from(projects, "code", object.getProject());



    Dataset dataset = super.unbind(object, "code", "creationMoment", "details", "difficultyLevel", "link", "totalTime", "draftMode", "project");

    dataset.put("creationMoment", object.getCreationMoment());

    dataset.put("difficultyLevelOptions", choices);
    dataset.put("project", projectsChoices.getSelected().getKey());
    dataset.put("projects", projectsChoices);

    super.getResponse().addData(dataset);
}
}

```

>  DeveloperTrainingModulePublishService.java  95,7 % 354 16 370

En esta característica observamos que todas las líneas de la clase han sido cubiertas por completo. Ha conseguido pasar todas las pruebas con éxito. Sin embargo, mediante la realización de esta prueba se pudo detectar el mismo fallo presente en las clases “DeveloperTrainingModuleCreateService” y “DeveloperTrainingModuleUpdateService”, se podía publicar un módulo de entrenamiento con un proyecto no publicado si este era introducido por consola cambiando la ip del mismo. Esto se solucionó añadiendo una restricción en el método “validate”.

Por lo tanto, respecto a la efectividad de los tests a la hora de encontrar fallos o vulnerabilidades, es necesario decir que para este servicio fue alta, ya que si se encontraron errores.

## DeveloperTrainingModuleController.java:

```
@Controller
public class DeveloperTrainingModuleController extends AbstractController<Developer, TrainingModule> {

    // Internal state -----

    @Autowired
    private DeveloperTrainingModuleListService    listService;

    @Autowired
    private DeveloperTrainingModuleShowService    showService;

    @Autowired
    private DeveloperTrainingModuleUpdateService    updateService;

    @Autowired
    private DeveloperTrainingModuleCreateService    createService;

    @Autowired
    private DeveloperTrainingModuleDeleteService    deleteService;

    @Autowired
    private DeveloperTrainingModulePublishService    publishService;
    // Constructors -----

    @PostConstruct
    protected void initialise() {
        super.addBasicCommand("show", this.showService);
        super.addBasicCommand("update", this.updateService);
        super.addBasicCommand("create", this.createService);
        super.addBasicCommand("delete", this.deleteService);
        super.addBasicCommand("list", this.listService);

        super.addCustomCommand("publish", "update", this.publishService);
    }
}
```

















>  DeveloperTrainingModuleController.java		100,0 %	35	0	35
--	--	---------	----	---	----

Esta clase ha conseguido ser cubierta al 100%, sin ningún problema.



## TrainingSession:

En el proceso de pruebas de la entidad “TrainingSession” se ha logrado alcanzar una cobertura del 96,3%. Esto es un indicador muy positivo del nivel de exhaustividad de nuestras pruebas. Esta cobertura se traduce en que, de un total de 2070 instrucciones que componen esta entidad, 1993 han sido cubiertas por los casos de prueba, dejando solo 77 instrucciones sin cubrir.

▼  acme.features.developer.trainingSessions	 96,3 %	1.993	77	2.070
>  DeveloperTrainingSessionCreateService.java	 96,8 %	481	16	497
>  DeveloperTrainingSessionDeleteService.java	 92,4 %	195	16	211
>  DeveloperTrainingSessionPublishService.java	 96,8 %	485	16	501
>  DeveloperTrainingSessionUpdateService.java	 96,7 %	471	16	487
>  DeveloperTrainingSessionListService.java	 95,2 %	179	9	188
>  DeveloperTrainingSessionShowService.java	 97,4 %	147	4	151
>  DeveloperTrainingSessionController.java	 100,0 %	35	0	35

Es importante señalar que todos los tests muestran líneas amarillas correspondientes a “assert object !=null”, lo cual es completamente normal. Si estas líneas se hubieran cubierto, significaría que el sistema ha encontrado un fallo irreversible durante las pruebas.

Respecto a los casos de pruebas positivos y negativos, se han probado realizando los pasos que realizaría un usuario estándar logueado como “**developer**” al utilizar la aplicación:

List.safe	<ul style="list-style-type: none"><li>- Listar sesiones de entrenamiento pertenecientes a un módulo de entrenamiento perteneciente al desarrollador con el que se está logueado (además de un listado vacío).</li></ul>
Show.safe	<ul style="list-style-type: none"><li>- Mostrar los detalles de las sesiones de entrenamiento (publicadas o sin publicar) pertenecientes a un módulo de</li></ul>

	<p>entrenamiento perteneciente al desarrollador con el que se está logueado.</p>
Delete.safe	<ul style="list-style-type: none"> <li>- Eliminar sesiones de entrenamiento sin publicar.</li> <li>- Comprobar que se ha eliminado correctamente.</li> </ul>
Create.safe	<ul style="list-style-type: none"> <li>- Crear sesiones de entrenamiento asociadas a módulos de entrenamiento no publicados (que un módulo de entrenamiento esté publicado implica que todas sus sesiones de entrenamiento lo están, por lo que no se podrían crear más sesiones de entrenamiento para dicho módulo) probando los campos del formulario con todo tipo de valores (además de con valores positivos, también se han probado valores negativos rechazados por el sistema, como valores nulos, repetidos, duplicados, URLs, correos y fechas incorrectas o no válidas por las restricciones establecidas,... comprobando los límites establecidos para estos).</li> <li>- Comprobar que se ha creado correctamente.</li> <li>- Comprobar que podemos ver sus detalles.</li> </ul>
Update.safe	<ul style="list-style-type: none"> <li>- Actualizar sesiones de entrenamiento no publicadas asociadas a módulos de entrenamiento no publicados</li> </ul>

	<p>(que un módulo de entrenamiento esté publicado implica que todas sus sesiones de entrenamiento lo están, por lo que no se podrían actualizar ninguna de ellas), probando los campos del formulario con todo tipo de valores (además de con valores positivos, también se han probado valores negativos rechazados por el sistema, como valores nulos, repetidos, duplicados, URLs, correos y fechas incorrectas o no válidas por las restricciones establecidas,... comprobando los límites establecidos para estos).</p> <ul style="list-style-type: none"> <li>- Comprobar que se ha actualizado correctamente.</li> <li>- Comprobar que podemos ver sus detalles.</li> </ul>
Publish.safe	<ul style="list-style-type: none"> <li>- Publicar sesiones de entrenamiento no publicadas asociadas a módulos de entrenamiento no publicados (que un módulo de entrenamiento esté publicado implica que todas sus sesiones de entrenamiento lo están, por lo que no se podríamos crear nuevas para publicarlas), probando los campos del formulario con todo tipo de valores (además de con valores positivos, también se han probado valores negativos rechazados por el sistema, como valores nulos, repetidos, duplicados, URLs, correos y fechas incorrectas o no válidas por las restricciones</li> </ul>

	<p>establecidas,... comprobando los límites establecidos para estos).</p> <ul style="list-style-type: none"> <li>- Comprobar que se ha publicado correctamente.</li> <li>- Comprobar que podemos ver sus detalles.</li> </ul>
--	---

Además, se debe destacar que se han realizado pruebas de hacking mediante la inserción de URLs para acceder sin permisos a funcionalidades concretas de un “developer” siendo otro rol, cualquier usuario o anónimo, acceder a las funcionalidades o elementos de otro “developer” siendo “developer” o a elementos inexistentes, y mediante el uso de la consola de comandos para introducir valores no válidos o erróneos por pantalla, y ver cómo el sistema respondía ante ellos.

Estas pruebas se pueden desglosar de la siguiente manera:

List.hack	<ul style="list-style-type: none"> <li>- Tratar de acceder a un listado de sesiones de entrenamiento siendo cualquier otro rol que no sea el de “developer” o sin estar logueado.</li> <li>- Tratar de acceder a un listado de sesiones de entrenamiento siendo “developer” cuyo módulo de entrenamiento no existe.</li> <li>- Tratar de acceder a un listado de sesiones de entrenamiento siendo “developer” cuyo módulo de entrenamiento pertenece a otro “developer”.</li> </ul>
Show.hack	<ul style="list-style-type: none"> <li>- Tratar de acceder a los detalles de una sesión de entrenamiento</li> </ul>

	<p>desde cualquier rol que no sea el de “developer” o sin estar logueado.</p> <ul style="list-style-type: none"> <li>- Tratar de mostrar una sesión de entrenamiento inexistente</li> <li>- Tratar de mostrar una sesión de entrenamiento de otro “developer” (estando logueado como “developer”).</li> </ul>
Delete.hack	<ul style="list-style-type: none"> <li>- Tratar de acceder a esta funcionalidad (eliminar sesiones de entrenamiento) sin estar logueado o estando logueado con otro rol que no sea el de “developer”.</li> <li>- Tratar de eliminar una sesión de entrenamiento que no exista.</li> <li>- Tratar de eliminar una sesión de entrenamiento que esté publicada.</li> <li>- Tratar de eliminar una sesión de entrenamiento de otro “developer” (estando logueado como “developer”).</li> </ul>
Create.hack	<ul style="list-style-type: none"> <li>- Tratar de acceder a esta funcionalidad sin estar logueado o estando logueado con otro rol que no sea el de “developer”.</li> <li>- Tratar de crear una sesión de entrenamiento que esté asociada a un módulo de entrenamiento que no exista.</li> <li>- Tratar de crear una sesión de entrenamiento que esté asociada a un módulo de entrenamiento que esté publicado.</li> </ul>

Update.hack	<ul style="list-style-type: none"> <li>- Tratar de acceder a esta funcionalidad sin estar logueado o estando logueado con otro rol que no sea el de “developer”.</li> <li>- Tratar de actualizar una sesión de entrenamiento que no existe.</li> <li>- Tratar de actualizar una sesión de entrenamiento asociada a un módulo de entrenamiento que no existe.</li> <li>- Tratar de actualizar una sesión de entrenamiento que ya está publicada.</li> <li>- Tratar de actualizar una sesión de entrenamiento que pertenezca a otro “developer”.</li> </ul>
Publish.hack	<ul style="list-style-type: none"> <li>- Tratar de acceder a esta funcionalidad sin estar logueado o estando logueado con otro rol que no sea el de “developer”.</li> <li>- Tratar de publicar una sesión de entrenamiento que no existe.</li> <li>- Tratar de publicar una sesión de entrenamiento asociada a un módulo de entrenamiento que no existe.</li> <li>- Tratar de publicar una sesión de entrenamiento que ya está publicada.</li> <li>- Tratar de publicar una sesión de entrenamiento que pertenezca a otro “developer”.</li> </ul>

Por ende, en varias pruebas de hacking, a la hora de reproducir los tests, nos encontramos con errores como:

"Expected 'status' to be '500', but got '200'. Expected 'payload' to be '{banner=79}', but got '{banner=79, code=TS-FUV-781,...}'"

"Expected 'status' to be '500', but got '200'. Expected 'payload' to be '{banner=79}', but got '{banner=79, code=TS-FUV-781,...}'"

**Esto no indica que se haya realizado una prueba de hacking con éxito.** Como se ha comentado anteriormente, se han realizado pruebas utilizando la consola de comandos, y para probarlas se ha accedido de manera lícita a algunas funcionalidades.

Este error viene dado por:

- A la hora de actualizar una sesión de entrenamiento, intentar por url actualizar una sesión de entrenamiento ya publicada. Para esta prueba de hacking se accedió de manera lícita a la funcionalidad "show" de training module, y desde ahí se realizó dicha prueba.
- A la hora de publicar una sesión de entrenamiento, intentar por url publicar una sesión de entrenamiento que no pertenece a ese "developer". Para esta prueba de hacking se accedió de manera lícita a la funcionalidad "show" de training module, y desde ahí se realizó dicha prueba.

Es decir, este tipo de error no indica en ninguno de los casos y ocasiones que aparece que se consiguió romper la funcionalidad del sistema.

## DeveloperTrainingSessionListService.java:

```
@Service
public class DeveloperTrainingSessionListService extends AbstractService<Developer, TrainingSession> {

    // Internal state -----

    @Autowired
    private DeveloperTrainingSessionsRepository repository;

    // AbstractService interface -----

    @Override
    public void authorise() {

        boolean status;
        int id;
        TrainingModule trainingModule;

        id = super.getRequest().getData("masterId", int.class);
        trainingModule = this.repository.findOneTrainingModuleById(id);
        status = trainingModule != null && super.getRequest().getPrincipal().hasRole(trainingModule.getDeveloper());

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

    @Override
    public void load() {
        Collection<TrainingSession> objects;
        int masterId;

        masterId = super.getRequest().getData("masterId", int.class);
        objects = this.repository.findManyTrainingSessionsByTrainingModuleId(masterId);

        super.getBuffer().addData(objects);
    }
}
```



```

@Override
public void unbind(final TrainingSession object) {

    assert object != null;

    Collection<TrainingModule> trainingModules = this.repository.findAllTrainingModules();
    SelectChoices trainingModulesChoices = SelectChoices.from(trainingModules, "code", object.getTrainingModule());

    final Dataset dataset = super.unbind(object, "code", "trainingModule");
    dataset.put("trainingModule", trainingModulesChoices.getSelected().getLabel());
    dataset.put("trainingModules", trainingModulesChoices);

    if (object.getIsDraftMode())
        dataset.put("draftMode", "No");
    else {
        final Locale local = super.getRequest().getLocale();
        dataset.put("draftMode", local.equals(Locale.ENGLISH) ? "Yes" : "Si");
    }

    int masterId;

    masterId = super.getRequest().getData("masterId", int.class);

    super.getResponse().addGlobal("masterId", masterId);

    super.getResponse().addData(dataset);
}

@Override
public void unbind(final Collection<TrainingSession> objects) {



    assert objects != null;

    int masterId;
    TrainingModule trainingModule;
    final boolean showCreate;

    masterId = super.getRequest().getData("masterId", int.class);
    trainingModule = this.repository.findOneTrainingModuleById(masterId);
    showCreate = trainingModule.getDraftMode() && super.getRequest().getPrincipal().hasRole(trainingModule.getDeveloper());

    super.getResponse().addGlobal("masterId", masterId);
    super.getResponse().addGlobal("showCreate", showCreate);
}

```

>  DeveloperTrainingSessionListService.java	 95,2 %	179	9	188
--	--	-----	---	-----

En esta primera característica observamos que todas las líneas de la clase han sido cubiertas por completo, salvo la línea “dataset.put(“draftMode”, local.equals(Locale.ENGLISH) ? “Yes” : “Si”);”, debido a que la aplicación, a la hora de ejecutar los tests, sólo se puede probar en inglés, no en castellano. La línea del “showCreate” tampoco se ha ejecutado completamente, debido a que, a la hora de mostrar el botón “create” para crear más sesiones de entrenamiento, este solo se mostrará si se tiene el rol esperado y el “draftMode” del módulo de entrenamiento es true. Por ende, al listar sesiones de entrenamiento cuyo módulo de entrenamiento está publicado, dicho botón no aparecerá. No se ha refactorizado debido a que, aunque esta línea se ejecute parcialmente, es completamente necesaria que esté diseñada de esta forma. Ha conseguido pasar todas las pruebas con éxito, se ha obtenido un alto porcentaje de cobertura y no se encontró ningún fallo respecto a esta característica durante el desarrollo de los casos de prueba. Por lo tanto, respecto a la efectividad de los tests a la hora de encontrar fallos o vulnerabilidades, es necesario decir que para este servicio fue baja, ya que no se encontraron errores.

## DeveloperTrainingSessionShowService.java:

```
@Service
public class DeveloperTrainingSessionShowService extends AbstractService<Developer, TrainingSession> {

    // Internal state -----

    @Autowired
    private DeveloperTrainingSessionsRepository repository;

    // AbstractService interface -----

    @Override
    public void authorise() {
        boolean status;
        int masterId;
        TrainingSession trainingSession;
        final TrainingModule trainingModule;
        Developer developer;

        masterId = super.getRequest().getData("id", int.class);
        trainingSession = this.repository.findOneTrainingSessionById(masterId);
        trainingModule = trainingSession == null ? null : trainingSession.getTrainingModule();
        developer = trainingModule == null ? null : trainingModule.getDeveloper();
        status = trainingSession != null && //
            trainingModule != null && //
            super.getRequest().getPrincipal().hasRole(developer) && //
            trainingModule.getDeveloper().getId() == super.getRequest().getPrincipal().getActiveRoleId();

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

    @Override
    public void load() {
        TrainingSession object;
        int id;

        id = super.getRequest().getData("id", int.class);
        object = this.repository.findOneTrainingSessionById(id);

        super.getBuffer().addData(object);
    }

    @Override
    public void unbind(final TrainingSession object) {



        assert object != null;

        Dataset dataset;
        TrainingModule trainingModule;
        boolean draftMode;

        trainingModule = object.getTrainingModule();
        draftMode = trainingModule.getDraftMode();

        dataset = super.unbind(object, "code", "iniDate", "finalDate", "location", "instructor", "contactEmail", "link", "isDraftMode");
        dataset.put("masterId", trainingModule.getId());
        dataset.put("draftMode", draftMode);
        dataset.put("trainingModule", trainingModule);

        super.getResponse().addData(dataset);
    }
}
```

>  DeveloperTrainingSessionShowService.java  97,4 % 147 4 151

En esta clase observamos que todas las líneas de código han sido cubiertas exitosamente, salvo una de las condiciones en el método “authorise”, la cual se ejecutó parcialmente debido a que no fue completamente necesaria a la hora de

realizar los test de hacking. Es necesario mencionar que pese a ejecutarse parcialmente dicha línea, no se pudo “romper” este servicio bajo ninguna prueba de hacking utilizada. La línea se ha dejado intacta en el código ya que, mediante el uso de otra URL, podría ser necesaria para garantizar la seguridad de este servicio. Se ha obtenido un alto porcentaje de cobertura por lo que no se ha considerado un gran problema que esa línea no estuviera cubierta al completo, pues no es una línea roja. No se encontraron fallos probando este servicio. Por lo tanto, respecto a la efectividad de los tests a la hora de encontrar fallos o vulnerabilidades, es necesario decir que para este servicio fue baja, ya que no se encontraron errores.

## DeveloperTrainingSessionCreateService.java:

```
@Service
public class DeveloperTrainingSessionCreateService extends AbstractService<Developer, TrainingSession> {
    // Internal state -----

    @Autowired
    private DeveloperTrainingSessionsRepository repository;

    // AbstractService interface -----

    @Override
    public void authorise() {

        boolean status;
        int masterId;
        TrainingModule trainingModule;

        masterId = super.getRequest().getData("masterId", int.class);
        trainingModule = this.repository.findOneTrainingModuleById(masterId);
        status = trainingModule != null && trainingModule.getDraftMode() && super.getRequest().getPrincipal().hasRole(trainingModule.getDeveloper());

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

    @Override
    public void load() {

        final int masterId = super.getRequest().getData("masterId", int.class);
        TrainingModule trainingModule = this.repository.findOneTrainingModuleById(masterId);

        TrainingSession trainingSession = new TrainingSession();
        trainingSession.setTrainingModule(trainingModule);
        trainingSession.setIsDraftMode(true);

        super.getBuffer().addData(trainingSession);
    }

    @Override
    public void bind(final TrainingSession object) {
        assert object != null;

        super.bind(object, "code", "iniDate", "finalDate", "location", "instructor", "contactEmail", "link");
    }
}
```

```

@Override
public void validate(final TrainingSession object) {
    assert object != null;

    final String INI_DATE = "iniDate";
    final String FINAL_DATE = "finalDate";
    final String CREATION_MOMENT = "creationMoment";

    if (!super.getBuffer().getErrors().hasErrors(INI_DATE) && !super.getBuffer().getErrors().hasErrors(FINAL_DATE)) {
        final boolean startBeforeEnd = MomentHelper.isAfter(object.getFinalDate(), object.getIniDate());
        super.state(startBeforeEnd, FINAL_DATE, "developer.training-session.form.error.end-before-start");

        if (startBeforeEnd) {
            final boolean startOneWeekBeforeEndMinimum = MomentHelper.isLongEnough(object.getIniDate(), object.getFinalDate(), 7, ChronoUnit.DAYS);
            super.state(startOneWeekBeforeEndMinimum, FINAL_DATE, "developer.training-session.form.error.small-display-period");
        }
    }

    if (!super.getBuffer().getErrors().hasErrors(CREATION_MOMENT) && !super.getBuffer().getErrors().hasErrors(INI_DATE)) {
        final boolean startBeforeCreation = MomentHelper.isAfter(object.getIniDate(), object.getTrainingModule().getCreationMoment());
        super.state(startBeforeCreation, INI_DATE, "developer.training-session.form.error.start-before-creation");

        if (startBeforeCreation) {
            final boolean createOneWeekBeforeStartMinimum = MomentHelper.isLongEnough(object.getTrainingModule().getCreationMoment(), object.getIniDate(), 7, ChronoUnit.DAYS);
            super.state(createOneWeekBeforeStartMinimum, INI_DATE, "developer.training-session.form.error.start-before-creation");
        }
    }

    if (!super.getBuffer().getErrors().hasErrors(CREATION_MOMENT) && !super.getBuffer().getErrors().hasErrors(FINAL_DATE)) {
        final boolean endBeforeCreation = MomentHelper.isAfter(object.getFinalDate(), object.getTrainingModule().getCreationMoment());
        super.state(endBeforeCreation, FINAL_DATE, "developer.training-session.form.error.end-before-creation");

        if (endBeforeCreation) {
            final boolean createOneWeekBeforeEndMinimum = MomentHelper.isLongEnough(object.getTrainingModule().getCreationMoment(), object.getFinalDate(), 7, ChronoUnit.DAYS);
            super.state(createOneWeekBeforeEndMinimum, FINAL_DATE, "developer.training-session.form.error.end-before-creation");
        }
    }
}

if (!super.getBuffer().getErrors().hasErrors("code")) {
    final boolean duplicatedCode = this.repository.findAllTrainingSessions().stream().anyMatch(e -> e.getCode().equals(object.getCode()));
    super.state(!duplicatedCode, "code", "developer.training-session.form.error.duplicated-code");
}

int masterId = super.getRequest().getData("masterId", int.class);
TrainingModule trainingModule = this.repository.findOneTrainingModuleById(masterId);
final boolean noDraftTrainingModule = trainingModule.getDraftMode();
super.state(noDraftTrainingModule, "", "developer.training-session.form.error.trainingModule-noDraft");

Date minDate;
Date maxDate;

minDate = MomentHelper.parse("2000-01-01 00:00", "yyyy-MM-dd HH:mm");
maxDate = MomentHelper.parse("2200-12-31 23:59", "yyyy-MM-dd HH:mm");

if (!super.getBuffer().getErrors().hasErrors(INI_DATE))
    super.state(MomentHelper.isAfterOrEqual(object.getIniDate(), minDate), INI_DATE, "developer.training-session.form.error.before-min-date");

if (!super.getBuffer().getErrors().hasErrors(INI_DATE))
    super.state(MomentHelper.isBeforeOrEqual(object.getIniDate(), maxDate), INI_DATE, "developer.training-session.form.error.after-max-date");

if (!super.getBuffer().getErrors().hasErrors(INI_DATE))
    super.state(MomentHelper.isBeforeOrEqual(object.getIniDate(), MomentHelper.deltaFromMoment(maxDate, -7, ChronoUnit.DAYS)), INI_DATE, "developer.training-session.form.error.no-room-for-min");

if (!super.getBuffer().getErrors().hasErrors(FINAL_DATE))
    super.state(MomentHelper.isAfterOrEqual(object.getFinalDate(), minDate), FINAL_DATE, "developer.training-session.form.error.before-min-date");

if (!super.getBuffer().getErrors().hasErrors(FINAL_DATE))
    super.state(MomentHelper.isBeforeOrEqual(object.getFinalDate(), maxDate), FINAL_DATE, "developer.training-session.form.error.after-max-date");

if (!super.getBuffer().getErrors().hasErrors(FINAL_DATE))
    super.state(MomentHelper.isAfterOrEqual(object.getFinalDate(), MomentHelper.deltaFromMoment(minDate, 7, ChronoUnit.DAYS)), FINAL_DATE, "developer.training-session.form.error.no-room-for-max");
}

@Override
public void perform(final TrainingSession object) {
    assert object != null;


    this.repository.save(object);
}

@Override
public void unbind(final TrainingSession object) {
    assert object != null;

    Dataset dataset = super.unbind(object, "code", "iniDate", "finalDate", "location", "instructor", "contactEmail", "link", "isDraftMode");
    dataset.put("masterId", super.getRequest().getData("masterId", int.class));
    dataset.put("draftMode", object.getTrainingModule().getDraftMode());

    super.getResponse().addData(dataset);
}
}

```

>  DeveloperTrainingSessionCreateService.java  96,8 % 481 16 497

En esta característica observamos que todas las líneas de la clase han sido cubiertas por completo salvo dos de ellas en el método “validate”. La primera línea comprueba si “creationMoment” e “iniDate” contienen algún error. Esta línea se ejecuta parcialmente debido a que se ha probado en los casos de prueba negativos con fechas iniciales incorrectas, por lo que dicho atributo contendrá valores

incorrectos y en algunos casos no entrará a dicha condición. Con la segunda línea en amarillo del método “validate” ocurre lo mismo, esta vez comprobando “creationMoment” y “finalDate”. Por ende, la última condición en amarillo (ifEndBeforeCreation), se ejecutó parcialmente ya que, si previamente se validó que la fecha inicial debe ser anterior a la final, y que la inicial debe ser posterior al momento de creación del módulo de entrenamiento, aunque la fecha final se establezca como anterior al momento de creación del modulo de entrenamiento, el sistema nos indicará que esto no es posible ya que la fecha final es anterior a la inicial. Por otro lado, mediante la realización de esta prueba se pudo detectar un fallo. Era posible crear una sesión de entrenamiento cuya fecha inicial no fuera posterior, al menos de 7 días, respecto al “creationMoment” del módulo de entrenamiento. Este error ya ha sido corregido. Por lo tanto, respecto a la efectividad de los tests a la hora de encontrar fallos o vulnerabilidades, es necesario decir que para este servicio fue alta, ya que si se encontraron errores.

## DeveloperTrainingSessionUpdateService.java:

```
@Service
public class DeveloperTrainingSessionUpdateService extends AbstractService<Developer, TrainingSession> {

    // Internal state -----

    @Autowired
    protected DeveloperTrainingSessionsRepository repository;

    // AbstractService interface -----

    @Override
    public void authorise() {
        int id = super.getRequest().getData("id", int.class);
        TrainingSession trainingSession = this.repository.findOneTrainingSessionById(id);

        final Principal principal = super.getRequest().getPrincipal();
        final int userAccountId = principal.getAccountId();
        //
        final boolean authorise = trainingSession != null && trainingSession.getIsDraftMode() && trainingSession.getTrainingModule().getDeveloper().getUserAccount().getId() == userAccountId;

        super.getResponse().setAuthorised(authorise);
    }

    @Override
    public void load() {
        int id = super.getRequest().getData("id", int.class);
        TrainingSession trainingSession = this.repository.findOneTrainingSessionById(id);

        super.getBuffer().addData(trainingSession);
    }

    @Override
    public void bind(final TrainingSession object) {
        assert object != null;

        super.bind(object, "code", "iniDate", "finalDate", "location", "instructor", "contactEmail", "link");
    }

    @Override
    public void validate(final TrainingSession object) {
        assert object != null;

        final String INI_DATE = "iniDate";
        final String FINAL_DATE = "finalDate";
        final String CREATION_MOMENT = "creationMoment";

        if (!super.getBuffer().getErrors().hasErrors(INI_DATE) && !super.getBuffer().getErrors().hasErrors(FINAL_DATE)) {
            final boolean startBeforeEnd = MomentHelper.isAfter(object.getFinalDate(), object.getIniDate());
            super.state(startBeforeEnd, FINAL_DATE, "developer.training-session.form.error.end-before-start");

            if (startBeforeEnd) {
                final boolean startOneWeekBeforeEndMinimum = MomentHelper.isLongEnough(object.getIniDate(), object.getFinalDate(), 7, ChronoUnit.DAYS);

                super.state(startOneWeekBeforeEndMinimum, FINAL_DATE, "developer.training-session.form.error.small-display-period");
            }
        }

        if (!super.getBuffer().getErrors().hasErrors(CREATION_MOMENT) && !super.getBuffer().getErrors().hasErrors(INI_DATE)) {
            final boolean startBeforeCreation = MomentHelper.isAfter(object.getIniDate(), object.getTrainingModule().getCreationMoment());
            super.state(startBeforeCreation, INI_DATE, "developer.training-session.form.error.start-before-creation");

            if (startBeforeCreation) {
                final boolean createOneWeekBeforeStartMinimum = MomentHelper.isLongEnough(object.getTrainingModule().getCreationMoment(), object.getIniDate(), 7, ChronoUnit.DAYS);

                super.state(createOneWeekBeforeStartMinimum, INI_DATE, "developer.training-session.form.error.start-before-creation");
            }
        }
    }
}
```

```

    if (!super.getBuffer().getErrors().hasErrors(CREATION_MOMENT) && !super.getBuffer().getErrors().hasErrors(FINAL_DATE)) {
        final boolean endBeforeCreation = MomentHelper.isAfter(object.getFinalDate(), object.getTrainingModule().getCreationMoment());
        super.state(endBeforeCreation, FINAL_DATE, "developer.training-session.form.error.end-before-creation");
    }

    if (endBeforeCreation) {
        final boolean createOneWeekBeforeEndMinimum = MomentHelper.isLongEnough(object.getTrainingModule().getCreationMoment(), object.getFinalDate(), 7, ChronoUnit.DAYS);
        super.state(createOneWeekBeforeEndMinimum, FINAL_DATE, "developer.training-session.form.error.end-before-creation");
    }
}

if (!super.getBuffer().getErrors().hasErrors("code")) {
    final int trainingSessionId = super.getRequest().getData("id", int.class);
    final boolean duplicatedCode = this.repository.findAllTrainingSessions().stream().filter(e -> e.getId() != trainingSessionId).anyMatch(e -> e.getCode().equals(object.getCode()));
    super.state(duplicatedCode, "code", "developer.training-session.form.error.duplicated-code");
}

Date minDate;
Date maxDate;

minDate = MomentHelper.parse("2000-01-01 00:00", "yyyy-MM-dd HH:mm");
maxDate = MomentHelper.parse("2200-12-31 23:59", "yyyy-MM-dd HH:mm");

if (!super.getBuffer().getErrors().hasErrors(INI_DATE))
    super.state(MomentHelper.isAfterOrEqual(object.getIniDate(), minDate), INI_DATE, "developer.training-session.form.error.before-min-date");

if (!super.getBuffer().getErrors().hasErrors(INI_DATE))
    super.state(MomentHelper.isBeforeOrEqual(object.getIniDate(), maxDate), INI_DATE, "developer.training-session.form.error.after-max-date");

if (!super.getBuffer().getErrors().hasErrors(INI_DATE))
    super.state(MomentHelper.isBeforeOrEqual(object.getIniDate(), MomentHelper.deltaFromMoment(maxDate, -7, ChronoUnit.DAYS)), INI_DATE, "developer.training-session.form.error.no-room-for-");

if (!super.getBuffer().getErrors().hasErrors(FINAL_DATE))
    super.state(MomentHelper.isAfterOrEqual(object.getFinalDate(), minDate), FINAL_DATE, "developer.training-session.form.error.before-min-date");

if (!super.getBuffer().getErrors().hasErrors(FINAL_DATE))
    super.state(MomentHelper.isBeforeOrEqual(object.getFinalDate(), maxDate), FINAL_DATE, "developer.training-session.form.error.after-max-date");

if (!super.getBuffer().getErrors().hasErrors(FINAL_DATE))
    super.state(MomentHelper.isAfterOrEqual(object.getFinalDate(), MomentHelper.deltaFromMoment(minDate, 7, ChronoUnit.DAYS)), FINAL_DATE, "developer.training-session.form.error.no-room-for-");
}

@Override
public void perform(final TrainingSession object) {
    assert object != null;

    this.repository.save(object);
}

@Override
public void unbind(final TrainingSession object) {
    assert object != null;

    Dataset dataset = super.unbind(object, "code", "iniDate", "finalDate", "location", "instructor", "contactEmail", "link", "isDraftMode");
    dataset.put("masterId", object.getTrainingModule().getId());
    dataset.put("draftMode", object.getTrainingModule().getDraftMode());
    super.getResponse().addData(dataset);
}
}

```

>  DeveloperTrainingSessionUpdateService.java	 96,7 %	471	16	487
--	--	-----	----	-----

En esta característica observamos que todas las líneas de la clase han sido cubiertas por completo salvo dos de ellas en el método “validate”. La primera línea comprueba si “creationMoment” e “iniDate” contienen algún error. Esta línea se ejecuta parcialmente debido a que se ha probado en los casos de prueba negativos con fechas iniciales incorrectas, por lo que dicho atributo contendrá valores incorrectos y en algunos casos no entrará a dicha condición. Con la segunda línea en amarillo del método “validate” ocurre lo mismo, esta vez comprobando “creationMoment” y “finalDate”. Por ende, la última condición en amarillo (ifEndBeforeCreation), se ejecutó parcialmente ya que, si previamente se validó que la fecha inicial debe ser anterior a la final, y que la inicial debe ser posterior al momento de creación del módulo de entrenamiento, aunque la fecha final se establezca como anterior al momento de creación del modulo de entrenamiento, el sistema nos indicará que esto no es posible ya que la fecha final es anterior a la inicial. Ha conseguido pasar todas las pruebas con éxito, se ha obtenido un alto porcentaje de cobertura y no se encontró ningún fallo respecto a esta característica durante el desarrollo de los casos de prueba.

Por lo tanto, respecto a la efectividad de los tests a la hora de encontrar fallos o vulnerabilidades, es necesario decir que para este servicio fue baja, ya que no se encontraron errores.

## DeveloperTrainingSessionDeleteService.java:

```
@Service
public class DeveloperTrainingSessionDeleteService extends AbstractService<Developer, TrainingSession> {

    // Internal state -----

    @Autowired
    private DeveloperTrainingSessionsRepository repository;

    // AbstractService interface -----

    @Override
    public void authorise() {
        boolean status;
        int masterId;
        TrainingSession trainingSession;
        TrainingModule trainingModule;
        Developer developer;

        masterId = super.getRequest().getData("id", int.class);
        trainingSession = this.repository.findOneTrainingSessionById(masterId);
        trainingModule = trainingSession == null ? null : trainingSession.getTrainingModule();
        developer = trainingModule == null ? null : trainingModule.getDeveloper();
        status = trainingSession != null && trainingSession.getIsDraftMode() && //
            trainingModule != null && //
            trainingModule.getDraftMode() && //
            super.getRequest().getPrincipal().hasRole(developer) && //
            trainingModule.getDeveloper().getId() == super.getRequest().getPrincipal().getActiveRoleId();

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

    @Override
    public void load() {
        TrainingSession object;
        int id;

        id = super.getRequest().getData("id", int.class);
        object = this.repository.findOneTrainingSessionById(id);

        super.getBuffer().addData(object);
    }

    @Override
    public void validate(final TrainingSession object) {
        assert object != null;
    }

    @Override
    public void perform(final TrainingSession object) {
        assert object != null;

        this.repository.delete(object);
    }

    @Override
    public void unbind(final TrainingSession object) {
        assert object != null;

        Dataset dataset;
        TrainingModule trainingModule;

        trainingModule = object.getTrainingModule();

        dataset = super.unbind(object, "code", "iniDate", "finalDate", "location", "instructor", "contactEmail", "link", "isDraftMode");
        dataset.put("draftMode", trainingModule.getDraftMode());
        dataset.put("masterId", trainingModule.getId());

        super.getResponse().addData(dataset);
    }
}
```

En esta característica observamos que todas las líneas de la clase han sido cubiertas por completo, salvo una condición en el método “authorise” y el método unbind. Esta primera línea se ejecutó parcialmente debido a que no fue completamente necesaria a la hora de realizar los test de hacking. Estas líneas se han dejado intactas en el código ya que, mediante el uso de otra URL, podría ser necesaria para garantizar la seguridad de este servicio. Se ha obtenido un alto porcentaje de cobertura por lo que no se ha considerado un gran problema que esa línea no estuviera cubierta al completo, pues no es una línea roja. Ha conseguido pasar todas las pruebas con éxito.

Por lo tanto, respecto a la efectividad de los tests a la hora de encontrar fallos o vulnerabilidades, es necesario decir que para este servicio fue baja, ya que no se encontraron errores.

## DeveloperTrainingSessionPublishService.java:

```
@Service
public class DeveloperTrainingSessionPublishService extends AbstractService<Developer, TrainingSession> {

    // Internal state -----

    @Autowired
    private DeveloperTrainingSessionsRepository repository;

    // AbstractService interface -----

    @Override
    public void authorise() {
        boolean status;
        int trainingSessionId;
        TrainingModule trainingModule;
        TrainingSession trainingSession;

        trainingSessionId = super.getRequest().getData("id", int.class);
        trainingModule = this.repository.findOneTrainingModuleByTrainingSessionId(trainingSessionId);
        trainingSession = this.repository.findOneTrainingSessionById(trainingSessionId);
        status = trainingModule != null && trainingModule.getDraftMode() && trainingSession.getIsDraftMode() && super.getRequest().getPrincipal().hasRole(trainingModule.getDeveloper());

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

    @Override
    public void load() {
        TrainingSession object;
        int id;

        id = super.getRequest().getData("id", int.class);
        object = this.repository.findOneTrainingSessionById(id);

        super.getBuffer().addData(object);
    }
}
```



```

@Override
public void bind(final TrainingSession object) {
    assert object != null;

    super.bind(object, "code", "iniDate", "finalDate", "location", "instructor", "contactEmail", "link");
}

@Override
public void validate(final TrainingSession object) {
    assert object != null;

    final String INI_DATE = "iniDate";
    final String FINAL_DATE = "finalDate";
    final String CREATION_MOMENT = "creationMoment";

    if (!super.getBuffer().getErrors().hasErrors(INI_DATE) && !super.getBuffer().getErrors().hasErrors(FINAL_DATE)) {
        final boolean startBeforeEnd = MomentHelper.isAfter(object.getFinalDate(), object.getIniDate());
        super.state(startBeforeEnd, FINAL_DATE, "developer.training-session.form.error.end-before-start");

        if (startBeforeEnd) {
            final boolean startOneWeekBeforeEndMinimum = MomentHelper.isLongEnough(object.getIniDate(), object.getFinalDate(), 7, ChronoUnit.DAYS);
            super.state(startOneWeekBeforeEndMinimum, FINAL_DATE, "developer.training-session.form.error.small-display-period");
        }
    }

    if (!super.getBuffer().getErrors().hasErrors(CREATION_MOMENT) && !super.getBuffer().getErrors().hasErrors(INI_DATE)) {
        final boolean startBeforeCreation = MomentHelper.isAfter(object.getIniDate(), object.getTrainingModule().getCreationMoment());
        super.state(startBeforeCreation, INI_DATE, "developer.training-session.form.error.start-before-creation");

        if (startBeforeCreation) {
            final boolean createOneWeekBeforeStartMinimum = MomentHelper.isLongEnough(object.getTrainingModule().getCreationMoment(), object.getIniDate(), 7, ChronoUnit.DAYS);
            super.state(createOneWeekBeforeStartMinimum, INI_DATE, "developer.training-session.form.error.start-before-creation");
        }
    }

    if (!super.getBuffer().getErrors().hasErrors(CREATION_MOMENT) && !super.getBuffer().getErrors().hasErrors(FINAL_DATE)) {
        final boolean endBeforeCreation = MomentHelper.isAfter(object.getFinalDate(), object.getTrainingModule().getCreationMoment());
        super.state(endBeforeCreation, FINAL_DATE, "developer.training-session.form.error.end-before-creation");

        super.state(endBeforeCreation, FINAL_DATE, "developer.training-session.form.error.end-before-creation");

        if (endBeforeCreation) {
            final boolean createOneWeekBeforeEndMinimum = MomentHelper.isLongEnough(object.getTrainingModule().getCreationMoment(), object.getFinalDate(), 7, ChronoUnit.DAYS);
            super.state(createOneWeekBeforeEndMinimum, FINAL_DATE, "developer.training-session.form.error.end-before-creation");
        }
    }

    if (!super.getBuffer().getErrors().hasErrors("code")) {
        final int trainingSessionId = super.getRequest().getData("id", int.class);
        final boolean duplicatedCode = this.repository.findAllTrainingSessions().stream().filter(e -> e.getId() != trainingSessionId).anyMatch(e -> e.getCode().equals(object.getCode()));

        super.state(!duplicatedCode, "code", "developer.training-session.form.error.duplicated-code");
    }

    Date minDate;
    Date maxDate;

    minDate = MomentHelper.parse("2000-01-01 00:00", "yyyy-MM-dd HH:mm");
    maxDate = MomentHelper.parse("2200-12-31 23:59", "yyyy-MM-dd HH:mm");

    if (!super.getBuffer().getErrors().hasErrors(INI_DATE))
        super.state(MomentHelper.isAfterOrEqual(object.getIniDate(), minDate), INI_DATE, "developer.training-session.form.error.before-min-date");

    if (!super.getBuffer().getErrors().hasErrors(INI_DATE))
        super.state(MomentHelper.isBeforeOrEqual(object.getIniDate(), maxDate), INI_DATE, "developer.training-session.form.error.after-max-date");

    if (!super.getBuffer().getErrors().hasErrors(INI_DATE))
        super.state(MomentHelper.isBeforeOrEqual(object.getIniDate(), MomentHelper.deltaFromMoment(maxDate, -7, ChronoUnit.DAYS)), INI_DATE, "developer.training-session.form.error.no-room-for");

    if (!super.getBuffer().getErrors().hasErrors(FINAL_DATE))
        super.state(MomentHelper.isAfterOrEqual(object.getFinalDate(), minDate), FINAL_DATE, "developer.training-session.form.error.before-min-date");

    if (!super.getBuffer().getErrors().hasErrors(FINAL_DATE))
        super.state(MomentHelper.isBeforeOrEqual(object.getFinalDate(), maxDate), FINAL_DATE, "developer.training-session.form.error.after-max-date");

    if (!super.getBuffer().getErrors().hasErrors(FINAL_DATE))
        super.state(MomentHelper.isAfterOrEqual(object.getFinalDate(), MomentHelper.deltaFromMoment(minDate, 7, ChronoUnit.DAYS)), FINAL_DATE, "developer.training-session.form.error.no-room-f");
}

@Override
public void perform(final TrainingSession object) {
    assert object != null;

    object.setIsDraftMode(false);

    this.repository.save(object);
}

@Override
public void unbind(final TrainingSession object) {
    assert object != null;

    TrainingModule trainingModule;
    trainingModule = object.getTrainingModule();

    Dataset dataset = super.unbind(object, "code", "iniDate", "finalDate", "location", "instructor", "contactEmail", "link", "isDraftMode");
    dataset.put("masterId", object.getTrainingModule().getId());
    dataset.put("draftMode", object.getTrainingModule().getDraftMode());
    dataset.put("trainingModule", trainingModule);

    super.getResponse().addData(dataset);
}
}

```

En esta característica observamos que todas las líneas de la clase han sido cubiertas por completo salvo dos de ellas en el método “validate” (y una en el “authorise”, la cual comentaré más adelante). La primera línea comprueba si “creationMoment” e “iniDate” contienen algún error. Esta línea se ejecuta parcialmente debido a que se ha probado en los casos de prueba negativos con fechas iniciales incorrectas, por lo que dicho atributo contendrá valores incorrectos y en algunos casos no entrará a dicha condición. Con la segunda línea en amarillo del método “validate” ocurre lo mismo, esta vez comprobando “creationMoment” y “finalDate”. Por ende, la última condición en amarillo (ifEndBeforeCreation), se ejecutó parcialmente ya que, si previamente se validó que la fecha inicial debe ser anterior a la final, y que la inicial debe ser posterior al momento de creación del módulo de entrenamiento, aunque la fecha final se establezca como anterior al momento de creación del modulo de entrenamiento, el sistema nos indicará que esto no es posible ya que la fecha final es anterior a la inicial. Por otro lado, una condición en el método “authorise” se ejecutó parcialmente debido a que no fue completamente necesaria a la hora de realizar los test de hacking. Es necesario mencionar que pese a ejecutarse parcialmente dichas líneas, no se pudo “romper” este servicio bajo ninguna prueba de hacking utilizada. Estas líneas se han dejado intactas en el código ya que, mediante el uso de otra URL, podría ser necesaria para garantizar la seguridad de este servicio.

Este servicio ha conseguido pasar todas las pruebas con éxito, se ha obtenido un alto porcentaje de cobertura y no se encontró ningún fallo respecto a esta característica durante el desarrollo de los casos de prueba.

Por lo tanto, respecto a la efectividad de los tests a la hora de encontrar fallos o vulnerabilidades, es necesario decir que para este servicio fue baja, ya que no se encontraron errores.

## DeveloperTrainingSessionController

```
@Controller
public class DeveloperTrainingSessionController extends AbstractController<Developer, TrainingSession> {

    // Internal state -----

    @Autowired
    private DeveloperTrainingSessionListService    listService;

    @Autowired
    private DeveloperTrainingSessionShowService    showService;

    @Autowired
    private DeveloperTrainingSessionDeleteService  deleteService;

    @Autowired
    private DeveloperTrainingSessionCreateService  createService;

    @Autowired
    private DeveloperTrainingSessionUpdateService  updateService;

    @Autowired
    private DeveloperTrainingSessionPublishService publishService;

    // Constructors -----

    @PostConstruct
    protected void initialise() {
        super.addBasicCommand("list", this.listService);
        super.addBasicCommand("show", this.showService);
        super.addBasicCommand("delete", this.deleteService);
        super.addBasicCommand("update", this.updateService);
        super.addBasicCommand("create", this.createService);

        super.addCustomCommand("publish", "update", this.publishService);
    }
}
```

Esta clase ha conseguido ser cubierta al 100%, sin ningún problema.

## Rendimiento del testing funcional:

En primer lugar es necesario proporcionar una breve introducción sobre el análisis realizado. Se han llevado a cabo varias pruebas de rendimiento. En una de ellas, no se utilizaron los índices necesarios para optimizar las consultas SQL. En otra prueba, sí se aplicaron los índices mencionados anteriormente. Estas dos se realizaron desde mi ordenador personal. Además, se realizó una tercera prueba en el ordenador de uno de los miembros del grupo (con índices), ya que los resultados pueden variar dependiendo del equipo en el que se ejecuten.

El equipo externo utilizado es el de mi compañero de grupo *pabespnar*, que es el student 5.

### Performance testing sin índices vs con índices

En primer lugar, se irán mostrando los datos obtenidos en este primer análisis del rendimiento con y sin índices, y posteriormente los analizaré.

Before en mi PC (Sin índices)			After en mi PC (Con índices)		
Media	34,79059857		Media	34,13380411	
Error típico	1,630446043		Error típico	1,547430559	
Mediana	13,1438		Mediana	13,2583995	
Moda	3,325999		Moda	2,6792	
Desviación estándar	49,50760207		Desviación estándar	46,98688233	
Varianza de la muestra	2451,002663		Varianza de la muestra	2207,767112	
Curtosis	8,891466623		Curtosis	8,274226708	
Coefficiente de asimetría	2,478740294		Coefficiente de asimetría	2,311976835	
Rango	447,509599		Rango	441,277099	
Mínimo	2,3038		Mínimo	2,3322	
Máximo	449,813399		Máximo	443,609299	
Suma	32076,93188		Suma	31471,36739	
Cuenta	922		Cuenta	922	
Nivel de confianza(95,0%)	3,199820584		Nivel de confianza(95,0%)	3,036899122	
Interval (ms)	31,59077799	37,99041915	Interval (ms)	31,09690498	37,17070323
Interval (s)	0,031590778	0,037990419	Interval (s)	0,031096905	0,037170703

Estos son los resultados obtenidos utilizando la herramienta de "Estadísticas Descriptivas".

Se puede observar que el conjunto de datos "con índices" presenta una media ligeramente más baja (34.13) en comparación con el conjunto "sin índices" (34.79), lo que sugiere que, en promedio, los valores son menores cuando se utilizan los índices.

La media es menor en el conjunto "con índices" principalmente porque el MIR ha sido lo que más se ha reducido, mientras que el resto de las solicitudes se han

disminuido ligeramente o se han quedado igual, y otras han aumentado ligeramente también.

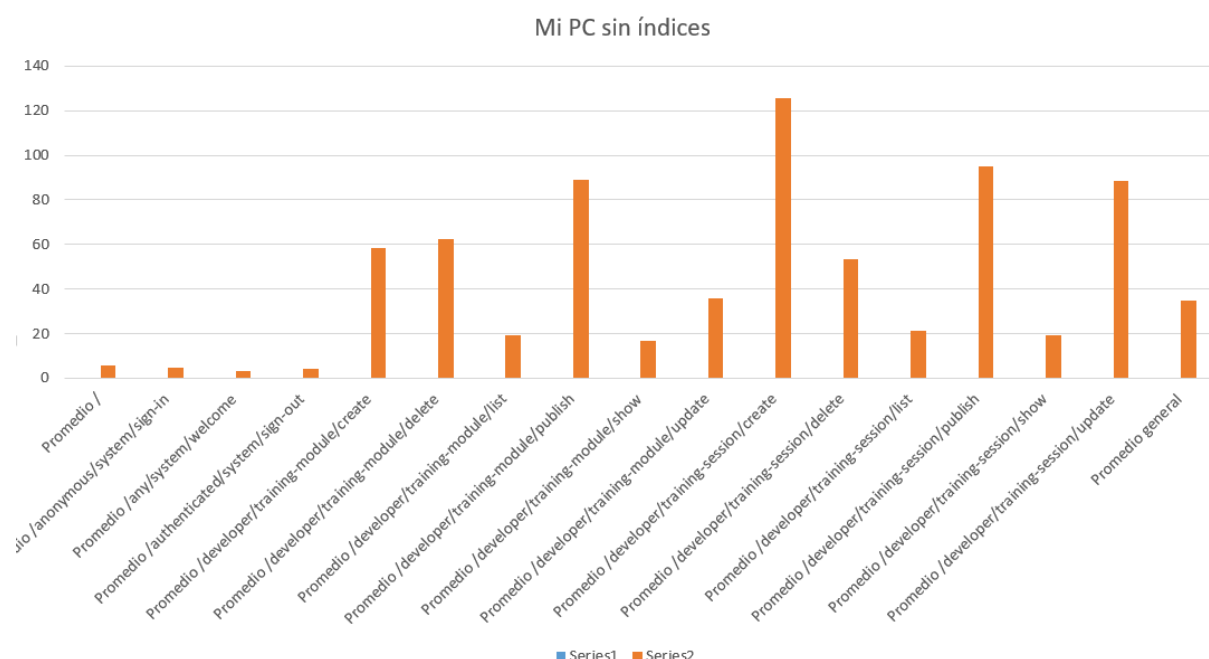
En cuanto a la mediana, esta es mayor en el conjunto "con índices" (13.25) frente al de "sin índices" (13.14), indicando que la mayoría de los valores en el conjunto "con índices" son más altos. En términos de mediana, esto significa que la mayoría de los valores en el conjunto "con índices" son más altos comparados con el conjunto "sin índices".

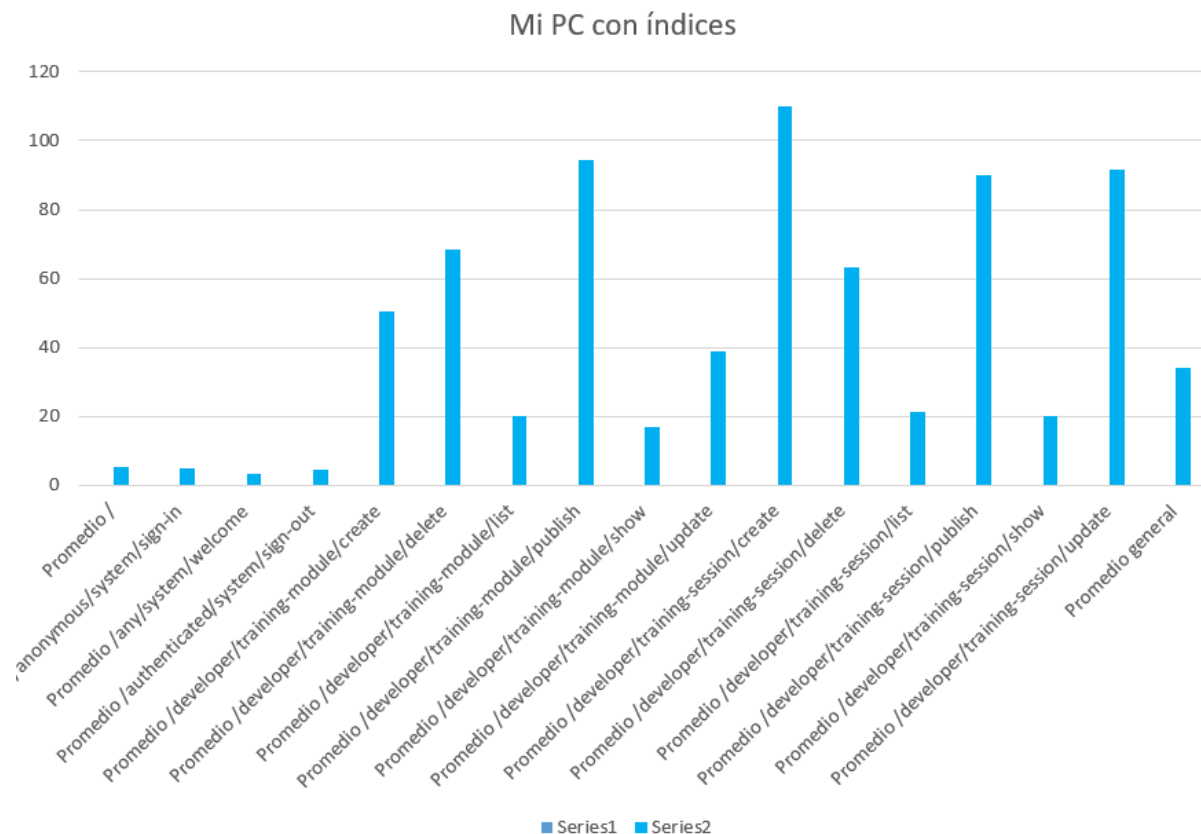
Sin embargo, se puede observar que la implementación de índices no ha marcado una diferencia significativa en el rendimiento general. Tanto la Desviación Estándar como la Varianza son mayores en el conjunto "sin índices", lo que indica una mayor variabilidad en los datos cuando no se utilizan los índices.

En cuanto a la Curtosis, el conjunto "con índices" tiene una curtosis menor que el conjunto "sin índices", lo que sugiere que la distribución "con índices" es menos puntiaguda y tiene colas menos pesadas.

Respecto a los intervalos, son muy similares, con variaciones más notables en los límites superiores, aunque la diferencia es mínima. Al convertir los intervalos de milisegundos a segundos, se confirma que están comprendidos en menos de un segundo, cumpliendo con los requisitos de esta asignatura.

En resumen, basándonos en estos datos, aún no podemos afirmar que el testeo "con índices" sea mejor que el "sin índices". Se verá si es notable esta diferencia más adelante.





En ambos histogramas se aprecia una forma similar, mostrando una relación comparable entre las barras. Sin embargo, en el histograma "con índices", la mayoría de los valores han disminuido ligeramente (excepto el MIR, que ha tenido la mayor reducción). Por otro lado, algunas barras han aumentado ligeramente en comparación con el histograma "sin índices". De esto, podemos inferir que el tiempo promedio en "con índices" es menor que en "sin índices". A pesar de esto, no podemos concluir que las pruebas "sin índices" sean peores que las pruebas "con índices". Por lo tanto, procederemos a analizar el último cálculo estadístico para obtener una conclusión definitiva.

Prueba z para medias de dos muestras		
	<i>Before (Sin indices)</i>	<i>After (Con índices)</i>
Media	34,79059857	34,13380411
Varianza (conocida)	2451,00266	2207,76711
Observaciones	922	922
Diferencia hipotética de las medias	0	
z	0,292185945	
P(Z<=z) una cola	0,385072229	
Valor crítico de z (una cola)	1,644853627	
Valor crítico de z (dos colas)	0,770144458	
Valor crítico de z (dos colas)	1,959963985	

Este último cálculo estadístico es la Prueba z para medias de dos muestras, con la cual determinaremos si existe una diferencia significativa entre las medias de ambos grupos.

Para este análisis, debemos enfocarnos en el primer "Valor crítico de z (dos colas)", que es 0.770144, y compararlo con Alfa, que es uno menos el nivel de confianza. Dado que el nivel de confianza es del 95%, Alfa es 0.05.

Como el valor crítico (0.770144) no se encuentra dentro del intervalo definido por Alfa ([0.00, 0.05]), **concluimos que no hay una diferencia estadísticamente significativa entre los conjuntos desde una perspectiva global**. Esto indica que, aunque existen diferencias numéricas menores en las estadísticas descriptivas y en los tiempos promedio observados, estas no se traducen en mejoras significativas entre los dos conjuntos de datos.

## Performance testing con índices vs con índices desde el ordenador de pabespnar

Mi PC con índices			PC de pabespnar con índices		
Media	34,13380411		Media	13,96719761	
Error típico	1,547430559		Error típico	0,524910807	
Mediana	13,2583995		Mediana	7,74315	
Moda	2,6792		Moda	2,3557	
Desviación estándar	46,98688233		Desviación estándar	15,93862948	
Varianza de la muestra	2207,767112		Varianza de la muestra	254,0399096	
Curtosis	8,274226708		Curtosis	8,719734232	
Coefficiente de asimetría	2,311976835		Coefficiente de asimetría	2,363897179	
Rango	441,277099		Rango	144,0343	
Mínimo	2,3322		Mínimo	1,3981	
Máximo	443,609299		Máximo	145,4324	
Suma	31471,36739		Suma	12877,7562	
Cuenta	922		Cuenta	922	
Nivel de confianza(95,0%)	3,036899122		Nivel de confianza(95,0%)	1,030160067	
Interval (ms)	31,09690498	37,17070323	Interval (ms)	12,93703755	14,99735768
interval (s)	0,031096905	0,037170703	Interval (s)	0,012937038	0,014997358

Estos son los resultados obtenidos utilizando la herramienta de "Estadísticas Descriptivas".

Se puede observar claramente que el conjunto de datos de mi PC presenta una media mucho más alta (34.13) en comparación con el conjunto del PC de pabespnar (13.96), lo que sugiere que los valores son mucho menores en el PC de mi compañero.

En cuanto a la mediana, esta es mayor en el conjunto de datos de mi PC (13.25) frente al del PC de mi compañero (7.74), indicando que la mayoría de los valores en el conjunto de mi PC son más altos. En términos de mediana, esto significa que la mayoría de los valores en el conjunto de mi PC son más altos comparados con el conjunto del PC de mi compañero.

Tanto la Desviación Estándar (46.98) como la Varianza (2207.76) son mayores en el conjunto de datos de mi PC que en el conjunto del PC de mi compañero (Desviación Estándar (15.93), Varianza (254.03)), lo que indica una mayor variabilidad en los datos en mi PC.

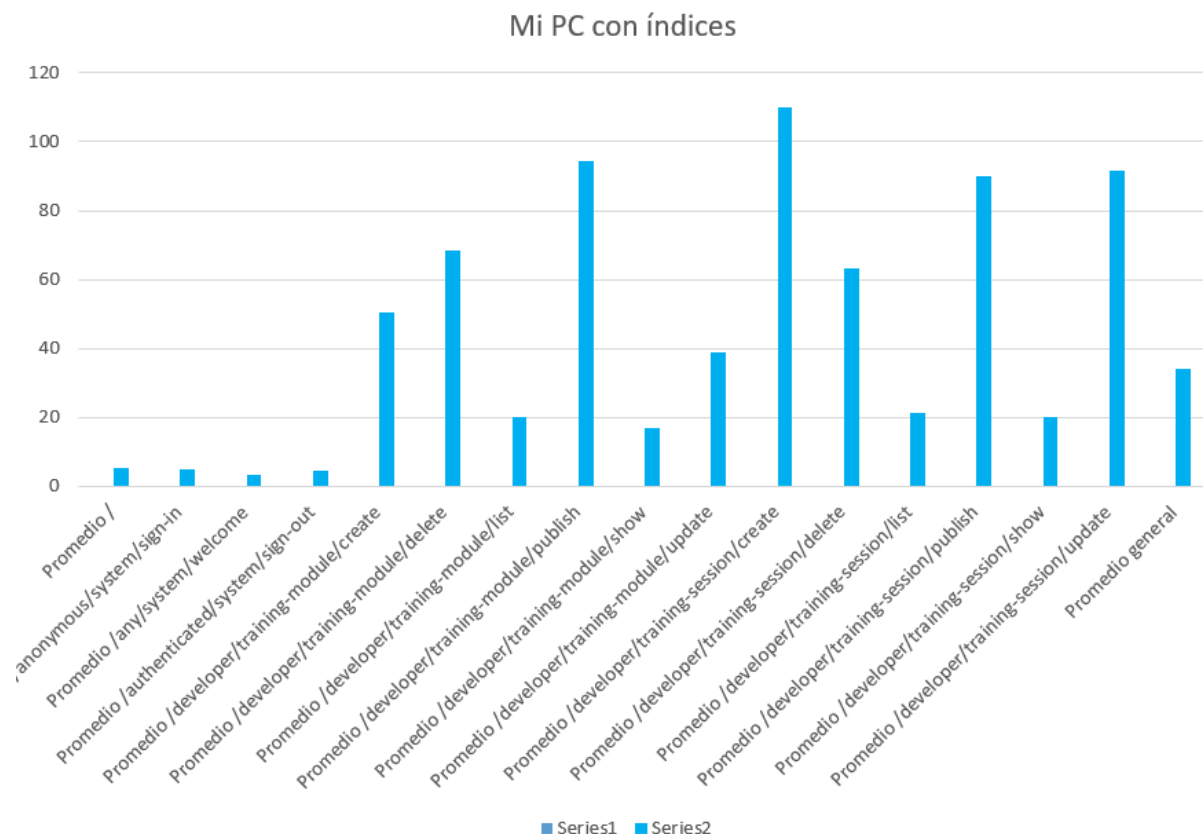
En cuanto a la Curtosis, el conjunto de datos de mi PC tiene una curtosis menor que el conjunto del PC de mi compañero, lo que sugiere que la distribución "con índices" es menos puntiaguda y tiene colas menos pesadas.

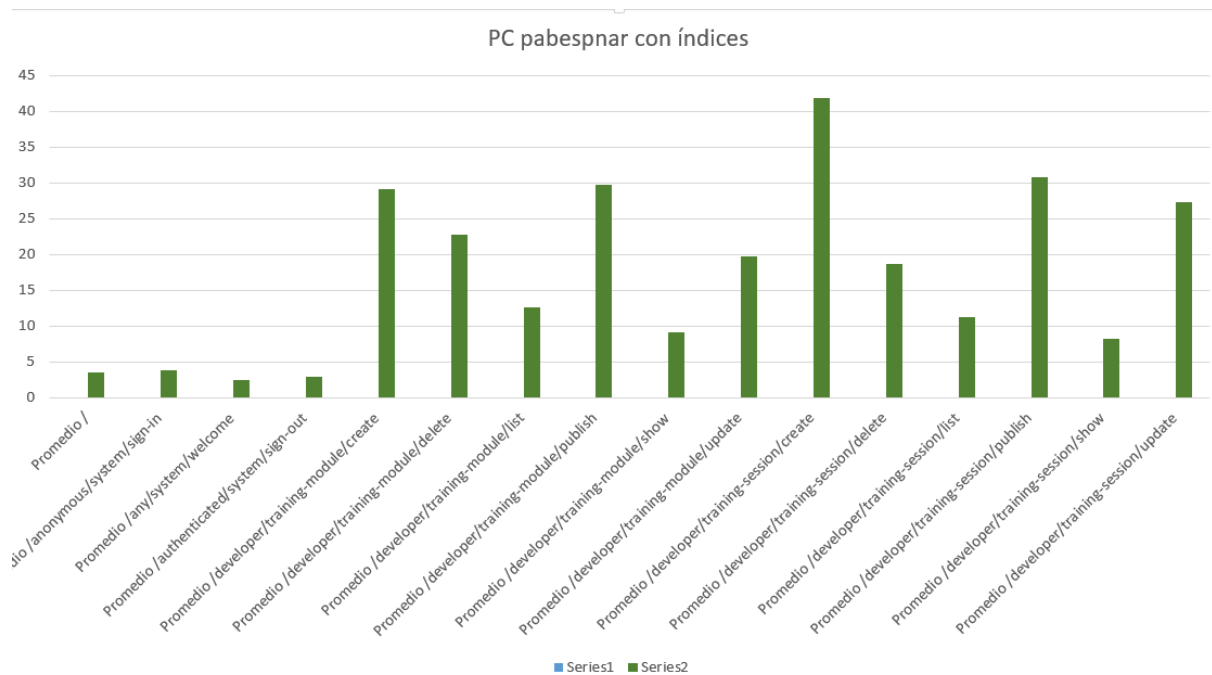
Respecto a los intervalos, vemos que hay bastante diferencia, siendo mucho mejores los proporcionados por el conjunto de datos del PC de mi compañero. No obstante, al convertir los intervalos de milisegundos a segundos, se confirma que



ambos pares están comprendidos en menos de un segundo, cumpliendo con los requisitos de esta asignatura.

En resumen, basándonos en estos datos, podemos comenzar a intuir que el testeo realizado por el PC de mi compañero, posiblemente por tener mejores prestaciones, presenta un rendimiento mucho mejor que el realizado por mi ordenador. Se verá como de notable es esta diferencia más adelante.





Fijándonos en los histogramas, podemos observar la diferencia abismal producida por la ejecución de las pruebas en mi ordenador y en el de mi compañero. Podemos observar como todas las request han disminuido drásticamente, habiendo muy pocas que se hayan mantenido constantes. De esto, podemos inferir que el tiempo promedio en el PC de mi compañero es mucho menor que en mi PC. Aún sabiendo esto, procederemos a analizar el último cálculo estadístico para obtener una conclusión definitiva y que termine de confirmar la superioridad del PC de mi compañero respecto al mío a la hora de realizar las pruebas.

Prueba z para medias de dos muestras		
	<i>before (Mi PC con índices)</i>	<i>after (PC de pabespnar con índices)</i>
Media	34,13380411	13,96719761
Varianza (conocida)	2207,767112	254,0399096
Observaciones	922	922
Diferencia hipotética de las medias	0	
z	12,34159474	
P(Z<=z) una cola	0	
Valor crítico de z (una cola)	1,644853627	
Valor crítico de z (dos colas)	0	
Valor crítico de z (dos colas)	1,959963985	

Este último cálculo estadístico es la Prueba z para medias de dos muestras, con la cual determinaremos si existe una diferencia significativa entre las medias de ambos grupos.

Para este análisis, debemos enfocarnos en el primer "Valor crítico de z (dos colas)", que es 0, y compararlo con Alfa, que es uno menos el nivel de confianza. Dado que el nivel de confianza es del 95%, Alfa es 0.05.

Como el valor crítico (0) se encuentra dentro del intervalo definido por Alfa ( $[0.00, 0.05]$ ), lo que indica que se puede hacer una comparación de los tiempos medios.

**Concluimos que hay una diferencia estadísticamente significativa** entre los conjuntos desde una perspectiva global y que las pruebas realizadas desde el ordenador de mi compañero tienen un mejor rendimiento que las realizadas desde el mío.

Esto se traduce en mejoras significativas entre los dos conjuntos de datos.

# Conclusiones

Sin llevar a cabo esta fase de pruebas formales, no podría haber afirmado con una precisión profesional que la calidad de mi producto era elevada. Esta etapa ha sido crucial no solo para detectar y solucionar errores, sino también para confirmar que cada componente de la aplicación opera como se espera bajo diversas condiciones. Este proceso detallado me ha permitido identificar fallos críticos que podrían haber pasado desapercibidos en una revisión más superficial, los cuales, de no haber sido corregidos, podrían haber resultado en problemas graves y potenciales fallos durante el uso por parte del cliente, afectando su experiencia y la reputación del producto.

Gracias a las pruebas formales, he podido garantizar que el producto final es sólido, confiable y capaz de cumplir con las expectativas y necesidades del usuario. Este nivel de análisis no solo mejora la calidad del producto, sino que también genera mayor confianza entre los usuarios y clientes, quienes pueden estar seguros de que están adquiriendo un producto bien desarrollado y probado rigurosamente.

Finalmente, el uso de pruebas en diferentes computadoras, así como las herramientas estadísticas utilizadas sobre los datos del producto, como el Z-Test y el intervalo de confianza, han sido esenciales para validar el rendimiento y la estabilidad de la aplicación. Estas herramientas estadísticas nos han permitido evaluar objetivamente si los cambios realizados han tenido un impacto significativo. El Z-Test, en particular, ha proporcionado una forma precisa de comparar los tiempos de ejecución antes y después de las modificaciones, mientras que el intervalo de confianza nos ha ofrecido una medida de la precisión de nuestras estimaciones. Estas metodologías han asegurado que los resultados obtenidos no son producto del azar, sino que reflejan mejoras o consistencias reales en el rendimiento del producto.

En conclusión, las pruebas formales no son solo una fase más del desarrollo, sino una piedra angular que asegura la calidad del producto, protege la satisfacción del cliente y fortalece la confianza en los productos que desarrollo.

# Bibliografía

Intencionalmente en blanco.