Test Report - D04



AUTOR:

* Ignacio Blanquero Blanco (ignblabla@alum.us.es)

Fecha: Sevilla, 27 de Mayo de 2024

Tabla de contenido

[Resumen del Informe 3](#_Toc167740767)

[Historial de Versiones 4](#_Toc167740768)

[Introducción 5](#_Toc167740769)

[Testing functional 6](#_Toc167740770)

[DeveloperTrainingModuleCreateService.java 7](#_Toc167740771)

[DeveloperTrainingModuleDeleteService.java 10](#_Toc167740772)

[DeveloperTrainingModuleListMineService.java 12](#_Toc167740773)

[DeveloperTrainingModulePublishService.java 14](#_Toc167740774)

[DeveloperTrainingModuleShowService.java 16](#_Toc167740775)

[DeveloperTrainingModuleUpdateService.java 17](#_Toc167740776)

[DeveloperTrainingSessionCreateService.java 19](#_Toc167740777)

[DeveloperTrainingSessionDeleteService.java 22](#_Toc167740778)

[DeveloperTrainingSessionListService.java 23](#_Toc167740779)

[DeveloperTrainingSessionPublishService.java 25](#_Toc167740780)

[DeveloperTrainingSessionShowService.java 27](#_Toc167740781)

[DeveloperTrainingSessionUpdateService.java 28](#_Toc167740782)

[Tests de hacking 30](#_Toc167740783)

[Rendimiento del testing 31](#_Toc167740784)

[Performance-testing sin índices 31](#_Toc167740785)

[Performance-testing con índices 33](#_Toc167740786)

[Comparación sin índices – con índices 35](#_Toc167740787)

[Conclusiones 36](#_Toc167740788)

[Bibliografía 37](#_Toc167740789)

# Resumen del Informe

El informe de testing desarrollado proporciona un análisis detallado de las pruebas realizadas a diversos servicios del rol *developer*, en concreto a los servicios de las entidades *trainingModule* y *trainingSession* asociadas a el rol. En el documento se incluyen las distintas pruebas funcionales que se han llevado a cabo, así como las pruebas de hacking y una evaluación del rendimiento de estas.

En definitiva, se resumen las principales observaciones y conclusiones derivadas del proceso de testing destacando las fortalezas y debilidades del proyecto.

# Historial de Versiones

|  |  |  |  |
| --- | --- | --- | --- |
| **Versión** | **Contenidos** | **Fecha** | **Contribuyente** |
| V0.0 | Estructura principal | 25/05/2024 | Ignacio Blanquero Blanco |
| V1.0 | Completado y finalización del documento | 27/05/2024 | Ignacio Blanquero Blanco |
|  |  |  |  |

# Introducción

Este informe detalla el progreso y los resultados de un conjunto de pruebas de software realizadas en el proyecto desarrollado por el equipo Acme-SF. El propósito principal es evaluar tanto la funcionalidad como el rendimiento del sistema bajo tests. Para ello, se han ejecutado diversas pruebas funcionales organizadas por características, cada una diseñada para detectar fallos y garantizar que las funciones del sistema operen según lo previsto. Además, se lleva a cabo un análisis de rendimiento, mediante dos formas distintas, una con índices y otra sin ellos.

# Testing functional

En el proceso de pruebas de la entidad *“TrainingModule.java”*, se ha logrado alcanzar un 94.2% de cobertura. Este indicador es muy positivo, pues destaca la exhaustividad de las pruebas realizadas. Dicha cobertura se traduce en que, de un total de 1294 instrucciones, 1219 han sido cubiertas y 75 no lo han sido.



Aunque una cobertura del 94.2% es un excelente indicador, es importante recordar que el objetivo principal no es alcanzar el 100% de cobertura, sino asegurar que las áreas más críticas y susceptibles a errores del código estén adecuadamente cubiertas. Este objetivo ha sido cumplido con éxito, ya que ninguna de las áreas críticas corresponde a las 75 líneas que no fueron cubiertas.

En cuanto a la entidad *“TrainingSession.java”*, se ha logrado alcanzar una cobertura del 93.8%, siendo un indicador es muy positivo. Dicha cobertura se traduce en que, de un total de 1204 instrucciones, 1129 han sido cubiertas y 75 no lo han sido.



Es importante destacar que todos los tests incluyen líneas de código resaltadas en amarillo correspondientes a la instrucción *“assert object != null”*. Esta situación es completamente normal y esperada, ya que la cobertura de esta línea indicaría que el sistema ha encontrado un fallo irreversible durante los casos de prueba. Estas líneas están presentes como medida preventiva; sin embargo, es afortunado que no hayan sido ejecutadas durante las pruebas.

Es importante señalar que, en este proyecto, solo se han considerado posibles intentos de hacking que implican la inserción de URLs en la barra de búsqueda. Esto se debe a que no se disponía del conocimiento necesario para implementar otros tipos de pruebas de hacking en el contexto específico de este proyecto.

En caso de que haya fallos en la rama main a la hora de comprobar las pruebas de estas dos entidades, lo podrá ver en la rama llamada Task143T.

## DeveloperTrainingModuleCreateService.java

Para realizar el test correspondiente a la clase *DeveloperTrainingModuleCreateService*, se ha probado a crear distintos módulos de entrenamiento. Para ello se han seguido los siguientes pasos:

* En primer lugar, se ha enviado el formulario vacío para comprobar que saltaran todas las restricciones.
* Para el código del módulo de entrenamiento, se han probado las siguientes combinaciones de caracteres:
  + Orden inverso del código: en vez de seguir el formato "[A-Z]{1,3}-[0-9]{3}", se ha optado por probarlo de esta manera: [0-9]{3}-[A-Z]{1,3}.
  + Letras aleatorias, por ejemplo: aodjflaksfjasl.
  + Caracteres de distintos alfabetos: se han probado los caracteres de otros alfabetos que se encontraban en el Excel proporcionado por los profesores de la asignatura, llamado “*Sample Data”*, además de otros alfabetos los cuales se pueden encontrar en el csv llamado “*training-module.csv”*, como, por ejemplo: 오늘 하루도 행복하고 즐겁게 보내세요 건강하고 행복하세요 y Энэ өдөр хайртай бол та нарын цагаараа амьдралд баяртай болохыг хүсье.
  + Valores de hacking: Al igual que se ha hecho con los caracteres de distintos alfabetos, se han probado los valores que se encontraban en el Excel mencionado anteriormente, así como otros valores que se pueden encontrar en el csv, tales como *<h1>hola</h1> o <script>alert("hola")</script>*.
  + Se ha comprobado que el código no estuviera duplicado, con los valores *A-000* y *A-001*.
  + Se ha comprobado que el código *IBB-241*, el cual no había sido utilizado hasta el momento, es válido.
* Para estudiar el atributo del momento de creación *(“creation-moment”)*, se ha probado lo siguiente:
  + Fecha futura (2022/07/30 00:01 y 2025/01/29 15:00).
  + Año bisiesto en pasado (2020/02/29 00:00 y 2020/02/029 23:59).
  + Año no bisiesto (2019/02/29 23:59).
  + Fecha pasada en el límite (2022/07/29 23:59 y 2022/07/30 00:00).
  + Fecha pasada (2019/01/06 00:00).
* En cuanto a los detalles, se han comprobado los siguientes aspectos:
  + Longitud de la cadena superior a 100 caracteres.
  + Cadena vacía.
  + Cadena con un caracter.
  + Cadena con dos caracteres.
  + Cadena con 50 caracteres (aproximadamente).
  + Cadena con 99 caracteres.
  + Cadena con 100 caracteres.
  + Cadena con 101 caracteres.
  + Cadena escrita en distintos alfabetos. Se han utilizado los valores proporcionados en el Excel llamado “Sample Data”, así como distintos valores que se pueden encontrar en el fichero *“training-module.csv”*.
  + Cadena con código malicioso: Se han probado los valores proporcionados por los profesores de la asignatura, así como los proporcionados en el csv de la entidad TrainingModule.
  + Por último, se ha escrito una cadena de caracteres que cumpliera todas las restricciones para que fuera válida.
* A la hora de probar el nivel de dificultad, he decidido realizar lo siguiente:
  + Dejar el campo de “Nivel de dificultad” en blanco para comprobar que saltara la restricción.
  + Probar con los tres valores enumerados de dicho atributo (BASIC, INTERMEDIATE, ADVANCED).
* Para el momento de la actualización se ha comprobado:
  + Campo vacío.
  + Fecha con un minuto antes que la fecha de creación del módulo de entrenamiento.
  + Fecha igual a la de creación.
  + Fecha en pasado muy anterior a la de creación.
  + Fecha en futuro (2022/07/30 00:01).
  + Fecha actual (2022/07/30 00:00).
  + Fecha pasada en el límite (2022/07/29 23:59).
  + Año bisiesto en pasado (2020/02/29 00:00 y 2020/02/029 23:59).
  + Año no bisiesto (2019/02/29 23:59).
  + Año correcto y válido (2020/07/16 20:35).
* Para comprobar el Link:
  + Campo vacío.
  + Valores aleatorios, por ejemplo ajdfhaksjdalskdjalsd.es.
  + Valores proporcionados en el Excel *“Sample Data”*.
  + Formato válido, por ejemplo <https://www.laliga.com>.
* Para el tiempo total estimado:
  + Campo vacío.
  + Valor 0.
  + Valores negativos: -0.001, -9, -1.
  + Valores válidos: 1, 5, 10.
* Para el proyecto:
  + Proyecto vacío.
  + Proyecto válido que pueda ser editado.

A continuación, se muestran varias capturas de pantalla correspondientes a la ejecución de los tests de la clase *DeveloperTrainingModuleCreateService.java:*

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

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.

Por último, cabe destacar que la cobertura de dicho test es del 93.5%, habiendo 229 instrucciones cubiertas y 16 sin cubrir.



## DeveloperTrainingModuleDeleteService.java

Para realizar el test correspondiente a la clase *DeveloperTrainingModuleDeleteService*, se ha optado por comprobar los siguientes escenarios:

* Módulo de entrenamiento que puede ser editado, es decir, cuya propiedad “draftMode” es true, puede ser eliminado.
* Módulo de entrenamiento que no puede ser editado (propiedad “draftMode es false), no puede ser eliminado.

A continuación, se muestran varias capturas de pantalla correspondientes a la clase Java *DeveloperTrainingModuleDeleteService.java.*

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto, Correo electrónico

Descripción generada automáticamente

En esta característica, observamos que todas las líneas de la clase han sido completamente cubiertas, excepto la línea que declara la variable status en el método authorise(). Esta falta de cobertura se debe a la consideración de posibles intentos de hacking, aunque no se ha encontrado la URL necesaria para realizar este tipo de hacking en el conjunto de pruebas, o no se ha podido acceder a ella debido a la ausencia de identificadores en las URLs.

Los tipos de tests que cubrirían esta línea amarilla serían “delete-right-role-wrong-user” y “delete-right-role-right-user-wrong-action”. Esto indica que el usuario debe tener el rol de desarrollador y que el trainingModule debe estar en modo borrador, ya que un intento de hacking malicioso podría implicar borrar un módulo de entrenamiento que está publicado.

Por último, cabe destacar que la cobertura de dicho test es del 91.3%, habiendo 136 instrucciones cubiertas y 13 sin cubrir.



## DeveloperTrainingModuleListMineService.java

A la hora de realizar el test correspondiente a la clase Java *DeveloperTrainingModuleListMineService*, se ha optado por realizar las siguientes acciones:

* Iniciar sesión con el usuario *developer1* y listar todos los módulos de entrenamiento de dicho usuario.
* Iniciar sesión con el usuario *developer2* y listar todos los módulos de entrenamiento de dicho usuario.

A continuación, se muestran varias capturas de pantalla correspondientes a la clase Java *DeveloperTrainingModuleListMineService.java.*

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

En esta primera 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 y no se encontró ningún fallo respecto a esta característica durante el desarrollo de los casos de prueba.

Por último, cabe destacar que la cobertura de dicho test es del 97.1%, habiendo 136 instrucciones cubiertas y 4 sin cubrir.



## DeveloperTrainingModulePublishService.java

Para realizar el test de la clase Java *DeveloperTrainingModulePublishService*, se ha optado por seguir la misma metodología que para realizar las pruebas de la clase *DeveloperTrainingModuleCreateService*, siguiendo los mismos pasos mencionados anteriormente para probar todos los atributos.

A continuación, se muestran varias capturas de pantalla correspondientes a la clase Java *DeveloperTrainingModulePublishService.java.*Texto

Descripción generada automáticamente con confianza baja

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Texto

Descripción generada automáticamente con confianza baja

En esta característica, observamos que todas las líneas de la clase han sido completamente cubiertas, excepto la línea que declara la variable status en el método authorise(). Esta falta de cobertura se debe a la consideración de posibles intentos de hacking, aunque no se ha encontrado la URL necesaria para realizar este tipo de hacking en el conjunto de pruebas, o no se ha podido acceder a ella debido a la ausencia de identificadores en las URLs.

Los tipos de tests que cubrirían esta línea amarilla serían “publish-right-role-wrong-user” y “publis-right-role-right-user-wrong-action”. Esto indica que el usuario debe tener el rol de desarrollador y que el trainingModule debe estar en modo borrador, ya que un intento de hacking malicioso podría implicar publicar un módulo de entrenamiento que está publicado.

La cobertura de dicho test es del 93.7%, habiendo 284 instrucciones cubiertas y 19 sin cubrir.



## DeveloperTrainingModuleShowService.java

Para realizar el test correspondiente a la clase Java *DeveloperTrainingModuleShowService*, he optado por:

* Mostrar un módulo de entrenamiento publicado con el usuario developer1.
* Mostrar un módulo de entrenamiento no publicado con el usuario developer1.
* Mostrar un módulo de entrenamiento publicado con el usuario developer2.
* Mostrar un módulo de entrenamiento no publicado con el usuario developer2.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Texto

Descripción generada automáticamente

La cobertura de dicho test es del 97.2%, habiendo 138 instrucciones cubiertas y 4 sin cubrir.



## DeveloperTrainingModuleUpdateService.java

Para realizar el test de la clase Java *DeveloperTrainingModuleUpdateService*, se ha optado por seguir la misma metodología que para realizar las pruebas de la clase *DeveloperTrainingModuleCreateService*, siguiendo los mismos pasos mencionados anteriormente para probar todos los atributos.

A continuación, se muestran varias capturas de pantalla correspondientes a la clase Java *DeveloperTrainingModuleUpdateService.java.*

Texto

Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

En esta característica, observamos que todas las líneas de la clase han sido completamente cubiertas, excepto la línea que declara la variable status en el método authorise(). Esta falta de cobertura se debe a la consideración de posibles intentos de hacking, aunque no se ha encontrado la URL necesaria para realizar este tipo de hacking en el conjunto de pruebas, o no se ha podido acceder a ella debido a la ausencia de identificadores en las URLs.

Los tipos de tests que cubrirían esta línea amarilla serían “update-right-role-wrong-user” y “update-right-role-right-user-wrong-action”. Esto indica que el usuario debe tener el rol de desarrollador y que el trainingModule debe estar en modo borrador, ya que un intento de hacking malicioso podría implicar actualizar un módulo de entrenamiento que está publicado.

La cobertura de dicho test es del 93.2%, habiendo 260 instrucciones cubiertas y 19 sin cubrir.



## DeveloperTrainingSessionCreateService.java

Para realizar el test correspondiente a la clase *DeveloperTrainingSessionCreateService*, se ha probado a crear distintas sesiones de entrenamiento. Para ello se han seguido los siguientes pasos:

* En primer lugar, se ha enviado el formulario vacío para comprobar que saltaran todas las restricciones.
* Para el código de la sesión de entrenamiento, se han probado las siguientes combinaciones de caracteres:
  + Orden inverso del código: en vez de seguir el formato "TS-[A-Z]{1,3}-[0-9]{3}", se ha optado por probarlo de esta manera: "TS-[0-9]{3}-[A-Z]{1,3}".
  + Letras aleatorias, por ejemplo: aodjflaksfjasl.
  + Caracteres de distintos alfabetos: se han probado los caracteres de otros alfabetos que se encontraban en el Excel proporcionado por los profesores de la asignatura, llamado “*Sample Data”*, además de otros alfabetos los cuales se pueden encontrar en el csv llamado “*training-session.csv”*, como, por ejemplo: 오늘 하루도 행복하고 즐겁게 보내세요 건강하고 행복하세요 y Энэ өдөр хайртай бол та нарын цагаараа амьдралд баяртай болохыг хүсье.
  + Valores de hacking: Al igual que se ha hecho con los caracteres de distintos alfabetos, se han probado los valores que se encontraban en el Excel mencionado anteriormente, así como otros valores que se pueden encontrar en el csv, tales como *<h1>hola</h1> o <script>alert("hola")</script>*.
  + Se ha comprobado que el código no estuviera duplicado, con los valores *TS-M-000* y *TS-PQ-674*.
  + Se ha comprobado que el código *TS-IBB-241*, el cual no había sido utilizado hasta el momento, es válido.
* Para estudiar el atributo del comienzo del período *(“*startPeriodDate*”)*, se ha probado lo siguiente:
  + Fecha futura (2022/07/30 00:01 y 2025/01/29 15:00).
  + Año bisiesto en pasado (2020/02/29 00:00 y 2020/02/029 23:59).
  + Año no bisiesto (2019/02/29 23:59).
  + Fecha pasada en el límite (2022/07/29 23:59 y 2022/07/30 00:00).
  + Fecha pasada (2019/01/06 00:00).
  + Fecha un minuto antes que la creación del TrainingModule.
  + Misma fecha de creación del TrainingModule.
  + Un minuto después de la fecha de creación del TrainingModule.
  + Fecha válida.
* Para estudiar el atributo de finalización del período *(“*finishPeriodDate*”)*, se ha probado lo siguiente:
  + Una semana justa después del comienzo de la sesión de entrenamiento.
  + Un día después del comienzo de la sesión de entrenamiento.
  + Año bisiesto.
  + Fecha válida.
  + Seis días, veintitrés horas y 59 minutos después de la fecha de inicio de la sesión.
* En cuanto a la localización y al instructor, se han comprobado los siguientes aspectos:
  + Longitud de la cadena superior a 75 caracteres.
  + Cadena vacía.
  + Cadena con un caracter.
  + Cadena con dos caracteres.
  + Cadena con 50 caracteres (aproximadamente).
  + Cadena con 74 caracteres.
  + Cadena con 75 caracteres.
  + Cadena con 76 caracteres.
  + Cadena escrita en distintos alfabetos. Se han utilizado los valores proporcionados en el Excel llamado “Sample Data”, así como distintos valores que se pueden encontrar en el fichero *“training-module.csv”*.
  + Cadena con código malicioso: Se han probado los valores proporcionados por los profesores de la asignatura, así como los proporcionados en el csv de la entidad TrainingModule.
  + Por último, se ha escrito una cadena de caracteres que cumpliera todas las restricciones para que fuera válida.
* A la hora de probar el email, he decidido realizar lo siguiente:
  + Se han utilizado los diferentes correos proporcionados en el Excel.
  + Se ha probado un valor vacío.
  + Formato inválido, por ejemplo adjflakdjflakjdfalskdalsdgmail.com.
* Para comprobar el Link:
  + Campo vacío.
  + Valores aleatorios, por ejemplo ajdfhaksjdalskdjalsd.es.
  + Valores proporcionados en el Excel *“Sample Data”*.
  + Formato válido, por ejemplo <https://www.laliga.com>.

A continuación, se muestran varias capturas de pantalla correspondientes a la ejecución de los tests de la clase *DeveloperTrainingSessionCreateService.java:*

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

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, menos la línea que se encarga de la declaración de la variable status en el método authorise(). Esto es debido a que se ha considerado que se podría hacer algún tipo de hacking para ello, pero no se ha encontrado la url necesaria para realizar este hacking en el conjunto de pruebas. El tipo de test que cubriría esta línea amarilla sería *“create-right-role-wrong-user”*, por eso indica que debe tener el rol desarrollador y que esté en modo borrador, pues un intento de hacking malicioso podría ser intentar crear una sesión de entrenamiento que está publicado.

La cobertura de dicho test es del 94.0%, habiendo 250 instrucciones cubiertas y 16 sin cubrir.



## DeveloperTrainingSessionDeleteService.java

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente



## DeveloperTrainingSessionListService.java

A la hora de realizar el test correspondiente a la clase Java *DeveloperTrainingSessionListService*, se ha optado por realizar las siguientes acciones:

* Iniciar sesión con el usuario *developer1* y listar todas las sesiones de entrenamiento de dicho usuario.
* Iniciar sesión con el usuario *developer2* y listar todas las sesiones de entrenamiento de dicho usuario.

A continuación, se muestran varias capturas de pantalla correspondientes a la clase Java *DeveloperTrainingSessionListService.java.*

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Texto

Descripción generada automáticamente con confianza media

Todas las líneas de la clase han sido completamente cubiertas, excepto la línea que declara la variable status en el método authorise(). Esto se debe a la consideración de un posible hacking en esa parte del código, pero no se ha encontrado la URL necesaria para realizar dicho hacking en el conjunto de pruebas.

La cobertura de dicho test es del 95.0%, habiendo 151 instrucciones cubiertas y 8 sin cubrir.



## DeveloperTrainingSessionPublishService.java

Para realizar el test de la clase Java *DeveloperTrainingSessionPublishService*, se ha optado por seguir la misma metodología que para realizar las pruebas de la clase *DeveloperTrainingSessionCreateService*, siguiendo los mismos pasos mencionados anteriormente para probar todos los atributos.

A continuación, se muestran varias capturas de pantalla correspondientes a la clase Java *DeveloperTrainingSessionPublishService.java.*

Texto

Descripción generada automáticamente con confianza baja

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

En esta característica, observamos que todas las líneas de la clase han sido completamente cubiertas, excepto la línea que declara la variable status en el método authorise(). Esta falta de cobertura se debe a la consideración de posibles intentos de hacking, aunque no se ha encontrado la URL necesaria para realizar este tipo de hacking en el conjunto de pruebas, o no se ha podido acceder a ella debido a la ausencia de identificadores en las URLs.

Los tipos de tests que cubrirían esta línea amarilla serían “publish-right-role-wrong-user” y “publish-right-role-right-user-wrong-action”. Esto indica que el usuario debe tener el rol de desarrollador y que el trainingSession debe estar en modo borrador, ya que un intento de hacking malicioso podría implicar publicar una sesión de entrenamiento que está publicada.

La cobertura de dicho test es del 93.2%, habiendo 233 instrucciones cubiertas y 17 sin cubrir.



## DeveloperTrainingSessionShowService.java

Para realizar el test correspondiente a la clase Java *DeveloperTrainingSessionShowService*, he optado por:

* Mostrar una sesión de entrenamiento publicada con el usuario developer1.
* Mostrar una sesión de entrenamiento no publicada con el usuario developer1.
* Mostrar una sesión de entrenamiento publicada con el usuario developer2.
* Mostrar una sesión de entrenamiento no publicada con el usuario developer2.

A continuación, se muestran varias capturas de pantalla correspondientes a la clase Java *DeveloperTrainingSessionShowService.java*

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

En esta característica se puede observar el alto porcentaje de cobertura a pesar de la línea en amarillo situada en el método authorise().

La cobertura de dicho test es del 96.5%, habiendo 110 instrucciones cubiertas y 4 sin cubrir.



## DeveloperTrainingSessionUpdateService.java

Para realizar el test de la clase Java *DeveloperTrainingSessionUpdateService*, se ha optado por seguir la misma metodología que para realizar las pruebas de la clase *DeveloperTrainingSessionCreateService*, siguiendo los mismos pasos mencionados anteriormente para probar todos los atributos.

A continuación, se muestran varias capturas de pantalla correspondientes a la clase Java *DeveloperTrainingSessionUpdateService.java.*

Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

En esta característica, observamos que todas las líneas de la clase han sido completamente cubiertas, excepto la línea que declara la variable status en el método authorise(). Esta falta de cobertura se debe a la consideración de posibles intentos de hacking, aunque no se ha encontrado la URL necesaria para realizar este tipo de hacking en el conjunto de pruebas, o no se ha podido acceder a ella debido a la ausencia de identificadores en las URLs.

Los tipos de tests que cubrirían esta línea amarilla serían “update-right-role-wrong-user” y “update-right-role-right-user-wrong-action”. Esto indica que el usuario debe tener el rol de desarrollador y que el trainingSession debe estar en modo borrador, ya que un intento de hacking malicioso podría implicar borrar una sesión de entrenamiento que está publicada.

La cobertura de dicho test es del 93.5%, habiendo 246 instrucciones cubiertas y 17 sin cubrir.



## Tests de hacking

A la hora de realizar los tests de hacking para los distintos servicios de la entidad *trainingModule*, no se han podido realizar todas las categorías que se pedían, ya que había pruebas cuyo resultado era *“AssertionError”*, por lo que solo he decidió poner los errores que sí que he considerado son los importantes y los que tienen que aparecer, que son los errores ERROR Status: 500 Internal Server Error. ERROR Exception: acme.client.helpers.Assert.state(Assert.java:45): **Access is not authorised.**

Es por esto por lo que, en la carpeta, se encuentran los siguientes tests que se muestran a continuación:

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

En cuanto a la entidad *trainingSession* y sus tests, estos han seguido la misma metodología a la hora de hacerse, considerando aquellos en los que la salida era la mencionada anteriormente. En la carpeta de tests de la entidad *trainingSession*, se pueden encontrar los siguientes tests de hacking:

Imagen que contiene Aplicación

Descripción generada automáticamente



# Rendimiento del testing

Se han realizado varios “*performance-testing*”, uno de ellos sin los índices correspondientes para agilizar las querys SQL, otro de ellos con los índices.

## Performance-testing sin índices

Antes de haber introducido los índices, el resultado obtenido en el fichero tester.trace y pasado a Excel ha sido:

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Una vez promediado cada una de las peticiones y el tiempo que se ha tardado en realizar cada una para analizar el rendimiento, el resultado ha sido:

Tabla

Descripción generada automáticamente

Como podemos observar, la petición más ineficiente es, con diferencia, la de listar los módulos de entrenamiento. Se ha deducido que la causa de esta ineficiencia puede estar en el hecho de que, para probar todas las demás características de la aplicación, es necesario acceder primero a la lista de módulos de entrenamiento, lo cual acumula una gran cantidad de peticiones en ella, afectando negativamente su rendimiento.

Una posible solución podría ser implementar otras pestañas para que el usuario pueda acceder directamente a características específicas sin tener que pasar por la lista de módulos de entrenamiento. Sin embargo, esto resultaría en una interfaz de usuario mucho menos intuitiva. Dado que uno de los requisitos fundamentales del cliente es que la interfaz sea lo más sencilla posible, se ha decidido mantener el diseño actual a pesar de este inconveniente.

## Performance-testing con índices

A continuación, analizamos el segundo caso, en el que ya se habían introducido los índices.

Estos índices han sido:





Una vez introducidos los índices, el resultado obtenido en el fichero tester.trace y pasado a Excel ha sido:

Interfaz de usuario gráfica

Descripción generada automáticamente

Una vez promediado cada una de las peticiones y el tiempo que se ha tardado en realizar cada una para analizar el rendimiento, el resultado ha sido:

Tabla, Excel

Descripción generada automáticamente

Como podemos observar, la petición más ineficiente vuelve a ser la de listar los módulos de entrenamiento. Sin embargo, realizando una comparación de las dos imágenes, se observa como pasa de tener un valor cercano a 1600 a uno ligeramente superior a 1200.

Por lo tanto, la introducción de índices es favorable para el rendimiento del sistema, si bien este sigue siendo algo ineficiente para realizar esta función, debido a que para probar todas las demás características de la aplicación, es necesario acceder primero a la lista de módulos de entrenamiento, lo cual acumula una gran cantidad de peticiones en ella, afectando negativamente su rendimiento.

## Comparación sin índices – con índices

Al realizar la comparación entre los dos casos, se han obtenido los siguientes resultados:

Tabla

Descripción generada automáticamente

Analizando los intervalos, determinamos que el intervalo de confianza del 95% sín índices sería [87.88114377, 118.9486279] milisegundos, mientras que con índices sería [87.2216056, 118.154206] milisegundos.

El objetivo de todo esto es determinar si los cambios han mejorado el MIR o no. Para ello se debe revisar si la diferencia entre las medias es cero o no está en un intervalo de confianza del 95%.

Para simplificar este proceso de comparación se realizará una prueba z, en este caso se han obtenido los siguientes resultados:

Tabla

Descripción generada automáticamente

Para analizar esta prueba es necesario fijarse en el valor de la celda “Valor crítico de z(dos colas)”. Como podemos comprobar, este valor (0) está entre [0.00,Alpha], lo que indica que se puede hacer una comparación de los tiempos medios.

# Conclusiones

Después de revisar los tests realizados, se observa que Ignacio Blanquero Blanco ha logrado una cobertura bastante alta y aceptable de las entidades que debía abordar como estudiante tres. Además, se ha calculado un índice del 95% para el conjunto de casos de prueba sin utilizar los índices recomendados, otro conjunto que sí los utilizó.

Por consiguiente, podemos afirmar que la calidad del testing realizado y la amplitud de cobertura alcanzada han sido bastante satisfactorias.

# Bibliografía

Si no hay bibliografía, escribir “En blanco a propósito”.