Universidad de Sevilla  
Escuela Técnica Superior de Ingeniería Informática

TESTING REPORT INDIVIDUAL - D04



**C2.027**

**Repositorio**: <https://github.com/josrojrom1/DP2-G27-Acme>

**Autor:**

Alors Romero, Daniel

danalorom1@alum.us.es

**Fecha:** 07/07/2024

**Tabla de contenidos**

[**Resumen**](https://docs.google.com/document/d/17cVhWdu6Ne5DzSSZ5iu594mbz331oXas/edit#heading=h.gjdgxs)[**2**](https://docs.google.com/document/d/17cVhWdu6Ne5DzSSZ5iu594mbz331oXas/edit#heading=h.gjdgxs)

[**Tabla de revisión**](https://docs.google.com/document/d/17cVhWdu6Ne5DzSSZ5iu594mbz331oXas/edit#heading=h.30j0zll)[**2**](https://docs.google.com/document/d/17cVhWdu6Ne5DzSSZ5iu594mbz331oXas/edit#heading=h.30j0zll)

[**Introducción**](https://docs.google.com/document/d/17cVhWdu6Ne5DzSSZ5iu594mbz331oXas/edit#heading=h.1fob9te)[**2**](https://docs.google.com/document/d/17cVhWdu6Ne5DzSSZ5iu594mbz331oXas/edit#heading=h.1fob9te)

[**Contenidos**](https://docs.google.com/document/d/17cVhWdu6Ne5DzSSZ5iu594mbz331oXas/edit#heading=h.3znysh7) **3**

**Capítulo I: Testing funcional 3**

**Capítulo II: Testing de rendimiento 7**

[**Bibliografía**](https://docs.google.com/document/d/17cVhWdu6Ne5DzSSZ5iu594mbz331oXas/edit#heading=h.3dy6vkm) **11**

Resumen

Este documento trata del proceso de testing seguido en los requisitos individuales de entregas anteriores, con el objetivo de llegar a una conclusión de dichos resultados en materia de mejora del código o del mismo proceso de testing. El documento consta de dos capítulos que tratan de los métodos estadísticos realizados para calcular intervalos de confianza y el contraste de hipótesis, alcanzando una valoración final.

# Tabla de revisión

|  |  |  |
| --- | --- | --- |
| **Versión** | **Fecha** | **Descripción** |
| V1 | 19/05/2024 | Creación y desarrollo del documento |
| V2 | 07/07/2024 | Actualización de documentación |

# Introducción

En las siguientes páginas veremos todo el proceso de testing, empezando por el capítulo de Testing Funcional, donde exploraremos la cobertura y el cómo se ha logrado. Para poder seguir con el capítulo del testing de rendimiento, en donde exploraremos los intervalos de confianza en nuestros tiempos de respuesta.

# Capítulo I: Testing funcional

Para este capítulo exploraremos todos los casos de pruebas implementados, agrupados por features. Para cada feature se explicarán las pruebas realizadas junto a sus detecciones de errores y mejoras implementadas.

**Training Module:**

Antes de comenzar describiendo las pruebas, cabe resaltar que, en el create, update y publish se han añadido validaciones que podrían llegar a ser redundantes en pos de una mayor seguridad frente a usuarios malintencionados, aplicando validaciones en el create y update moment, aunque estos se actualizan con la fecha actual del sistema. Junto con el project asociado. Todo ello pensando en la integridad de los datos.

**developer/trainingModule/list:** Para definir estas pruebas procedíamos de la siguiente manera, accedíamos, registrandonos como developer, a nuestras listas de training module, mostrando las listas sin errores, obteniendo así el .safe. Para el hacking se ha hecho get hacking desde un usuario no registrado, intentando acceder a la lista de training modules de un desarrollador, obteniendo un 500, tras esto, desde otro rol, por ejemplo, sponsor, se intentaba acceder de nuevo a la lista de training Module, obteniendo nuevamente un 500.

Tras estas pruebas se ha detectado la gran cobertura de dicho servicio con un 94,9%. Sin errores aparentes a resaltar.

**developer/trainingModule/show:** Para definir estas pruebas procedíamos de la siguiente manera, accedíamos, registrandonos como developer, a las listas de training module, para luego, acceder a los forms de cualquier training module, mostrando el form sin errores, obteniendo así el .safe. para el hacking, hemos intentado acceder mediante un usuario no registrado a un show de training module, por medio de la url, obteniendo un 500. Después desde otro rol, por ejemplo, sponsor, hemos intentado acceder a un training module por medio de la url, obteniendo nuevamente un 500. También hemos intentado acceder desde otro usuario del mismo rol al show de un training module, para ello una vez registrado como developer2 hemos intentado acceder a un form del developer1 mediante su url, obteniendo un 500.

Tras todo ello se ha detectado una gran cobertura en el servicio con un 97%. 

**developer/trainingModule/delete:** Para definir estas pruebas procedíamos de la siguiente manera, accedemos, registrandonos como developer, al form de un training module, para luego pulsar el botón delete con la acción de borrado, este proceso se ha realizado para un training module sin hijos publicados y para uno con hijos publicados, en ambos siendo posible su eliminación, y en caso del borrado de la clase padres las clases hijas, publicadas o sin publicar, también se borraban. Para el hacking, hemos intentado borrar un training module que no le pertenezca a ese developer, obteniendo un 500. Seguido de intentar borrar un un training module ya publicado, obteniendo otro 500. Tras ello, probamos también a intentar borrar un training module que no existe, volviendo a obtener un 500. Y para finalizar intentamos borrar un training module desde otro rol distinto a developer.

Para la realización del hacking se usó “inspeccionar” sobre el botón delete para poder cambiar el id en los casos del mismo rol, como la propia dirección producida por el botón en la prueba de distinto rol.

Tras esto, cabe resaltar que el servicio no tiene gran cobertura debido a que, al no tener validaciones, el unbind no se realizará. 

**developer/trainingModule/create:** Para definir estas pruebas procedíamos de la siguiente manera, accedemos, registrandonos como developer, a la lista de los training modules, una vez en ella accedemos al botón para poder crear un training module, empezamos enviándo el form vacío, tras obtener las validaciones de los campos nulos, se han probado todas las validaciones mediante los datos facilitados en el Sample-Data (Excel), más invenciones propias que se vieron pertinentes para probar todo lo posible. Tras ver todas las validaciones, se fue creando un training module para los datos límites de cada apartado de la entidad, es decir, el límite superior, inferior, otro en el intervalo, entre otros. Tras todo ello, se realizó la prueba de hacking, para la cual, se intentó pasar la url del create para un usuario no registrado y para un usuario con un rol distinto a developer, obteniendo en ambos casos un 500.

Tras todo ello se obtuvo una gran cobertura, del 93%. 

**developer/trainingModule/update:** Para definir estas pruebas procedimos de la siguiente manera, accedemos, registrandonos como developer, al form de un training module no publicado, una vez en él accedemos al botón para poder actualizar un training module, empezamos enviándo el form vacío, obteniendo las validaciones de los campos nulos, tras esto, se han probado todas las validaciones mediante los datos facilitados en el Sample-Data (Excel), más invenciones propias que se vieron pertinentes para probar todo lo posible. Tras aplicar todas las validaciones, se fue actualizando un training module para los datos límites de cada apartado de la entidad, es decir, el límite superior, inferior, otro en el intervalo, entre otros. Tras todo ello, se realizó la prueba de hacking, para la cual, se intentó actualizar un training module ya publicado, tras esto, también se intentó actualizar un training module con un id inventada, junto con intentar actualizar un training module de otro desarrollador, por último, intentamos que un sponsor actualizase un training module, por lo cual dio un error 500, igual que en el resto de los casos. Para la realización del hacking se usó “inspeccionar” sobre el botón delete para poder cambiar el id en los casos del mismo rol, como la propia dirección producida por el botón en la prueba de distinto rol.

Tras todo ello se obtuvo una cobertura del 93,9%. 

**developer/trainingModule/publish:** Para definir estas pruebas procedimos de la siguiente manera, accedemos, registrandonos como developer, al form de un training module no publicado (y que pueda ser publicado), una vez en él accedemos al botón para poder publicar un training module, empezamos enviando el form vacío, obteniendo las validaciones de los campos nulos, tras esto, se han probado todas las validaciones mediante los datos facilitados en el Sample-Data (Excel), más invenciones propias que se vieron pertinentes para probar todo lo posible las validaciones, para acto seguido publicar el training module con datos correctos. Tras todo ello, se realizó la prueba de hacking, para la cual, se intentó publicar un training module ya publicado, tras esto, también se intentó publicar un training module con un id inventada, junto con intentar publicar un training module de otro desarrollador, por último, intentamos que un sponsor publicase un training module, por lo cual dio un error 500, igual que en el resto de los casos. Para la realización del hacking se usó “inspeccionar” sobre el botón delete para poder cambiar el id en los casos del mismo rol, como la propia dirección producida por el botón en la prueba de distinto rol.

Tras todo ello se obtuvo una cobertura del 94,8%. 

**Training Sessions:**

**developer/trainingSessions/list:** Para definir estas pruebas procedíamos de la siguiente manera, accedíamos, registrandonos como developer, a nuestras listas de training sessions, mostrando las listas sin errores, obteniendo así el .safe. Para el hacking se ha hecho get hacking desde un usuario no registrado, intentando acceder a la lista de training sessions de un desarrollador, obteniendo un 500, tras esto, desde otro rol, por ejemplo, sponsor, se intentaba acceder de nuevo a la lista de training sessions, obteniendo nuevamente un 500. Junto a intentar que un developer intentase listar los trainings sessions de un training module del que no es dueño. Tras estas pruebas se ha detectado la gran cobertura de dicho servicio con un 97,3%. 

**developer/trainingSessions/show:** Para definir estas pruebas procedíamos de la siguiente manera, accedíamos, registrandonos como developer, a las listas de training sessions, para luego, acceder a los forms de cualquier training session, mostrando el form sin errores, obteniendo así el .safe. para el hacking, hemos intentado acceder mediante un usuario no registrado a un show de training session, por medio de la url, obteniendo un 500. Después desde otro rol, por ejemplo, sponsor, hemos intentado acceder a un training session por medio de la url, obteniendo nuevamente un 500. También hemos intentado acceder desde otro usuario del mismo rol al show de un training session, para ello una vez registrado como developer2 hemos intentado acceder a un form del developer1 mediante su url, obteniendo un 500.

Tras todo ello se ha detectado una gran cobertura en el servicio con un 96,9%. Sin embargo, cabe resaltar que el authorise no se han cubierto todos los caminos. Pudiendo ser debido a restricciones repetidas para asegurar el no hackeo del sistema.

**developer/trainingSessions/delete:** Para definir estas pruebas procedíamos de la siguiente manera, accedemos, registrandonos como developer, al form de un training sessions, para luego pulsar el botón delete con la acción de borrado, este proceso se ha realizado para un training sessions. Para el hacking, hemos intentado borrar un training session que no le pertenezca a ese developer, obteniendo un 500. Seguido de intentar borrar un un training session ya publicado, obteniendo otro 500. Tras ello, probamos también a intentar borrar un training session que no existe, volviendo a obtener un 500. Y para finalizar intentamos borrar un training session desde otro rol distinto a developer.

Para la realización del hacking se usó “inspeccionar” sobre el botón delete para poder cambiar el id en los casos del mismo rol, como la propia dirección producida por el botón en la prueba de distinto rol.

Tras esto, cabe resaltar que el servicio no tiene gran cobertura debido a que al no tener validaciones, el unbind no se realizará. 

**developer/trainingSessions/create:** Para definir estas pruebas procedíamos de la siguiente manera, accedemos, registrandonos como developer, a la lista de los training sessions, una vez en ella accedemos al botón para poder crear un training session, empezamos enviándo el form vacío, tras obtener las validaciones de los campos nulos, tras esto, se han probado todas las validaciones mediante los datos facilitados en el Sample-Data (Excel), más invenciones propias que se vieron pertinentes para probar todo lo posible. Tras ver todas las validaciones, se fue creando un training session para los datos límites de cada apartado de la entidad, es decir, el límite superior, inferior, otro en el intervalo, entre otros. Tras todo ello, se realizó la prueba de hacking, para la cual, se intentó pasar la url del create para un usuario no registrado y para un usuario con un rol distinto a developer, obteniendo en ambos casos un 500. Finalizando con intentar que otro developer intentase acceder al create del training session de un training module del que no es dueño, obteniendo un 500.

Tras todo ello se obtuvo una gran cobertura, del 95,2%.

**developer/trainingSessions/update:** Para definir estas pruebas procedimos de la siguiente manera, accedemos, registrandonos como developer, al form de un training session no publicado, una vez en él accedemos al botón para poder actualizar un training session, empezamos enviándo el form vacío, obteniendo las validaciones de los campos nulos, tras esto, se han probado todas las validaciones mediante los datos facilitados en el Sample-Data (Excel), más invenciones propias que se vieron pertinentes para probar todo lo posible. Tras aplicar todas las validaciones, se fue actualizando un training session para los datos límites de cada apartado de la entidad, es decir, el límite superior, inferior, otro en el intervalo, entre otros. Tras todo ello, se realizó la prueba de hacking, para la cual, se intentó actualizar un training session ya publicado, tras esto, también se intentó actualizar un training session con un id inventada, junto con intentar actualizar un training session de otro desarrollador, por último, intentamos que un sponsor actualizase un training session, por lo cual dio un error 500, igual que en el resto de los casos. Para la realización del hacking se usó “inspeccionar” sobre el botón delete para poder cambiar el id en los casos del mismo rol, como la propia dirección producida por el botón en la prueba de distinto rol.

Tras todo ello se obtuvo una cobertura del 95,2%. Cabe resaltar que hay bucles que como no se cumplen todos los caminos.

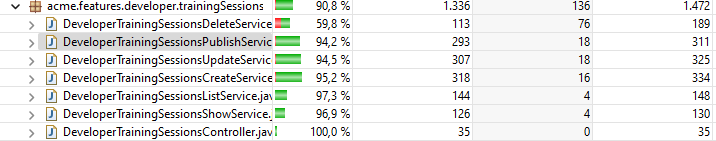
**developer/trainingSessions/publish:** Para definir estas pruebas procedimos de la siguiente manera, accedemos, registrandonos como developer, al form de un training session no publicado (y que pueda ser publicado), una vez en él accedemos al botón para poder publicar un training session, empezamos enviando el form vacío, obteniendo las validaciones de los campos nulos, tras esto, se han probado todas las validaciones mediante los datos facilitados en el Sample-Data (Excel), más invenciones propias que se vieron pertinentes para probar todo lo posible las validaciones, para acto seguido publicar el training session con datos correctos. Tras todo ello, se realizó la prueba de hacking, para la cual, se intentó publicar un training session ya publicado, tras esto, también se intentó publicar un training session con un id inventada, junto con intentar publicar un training session de otro desarrollador, por último, intentamos que un sponsor publicase un training session, por lo cual dio un error 500, igual que en el resto de los casos. Para la realización del hacking se usó “inspeccionar” sobre el botón delete para poder cambiar el id en los casos del mismo rol, como la propia dirección producida por el botón en la prueba de distinto rol.

Tras todo ello se obtuvo una cobertura del 94,2%.

La cobertura global queda tal que:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente



La cobertura podría mejorar comprobando exhaustivamente cada bucle y si sobra o no código en dichos bucles, aparte de lo mencionado al inicio de las validaciones contra el hacking en el training module. Junto a todo eso el unbind del delete pesa bastante en la cobertura total.

# Capítulo II: Testing de rendimiento

En este capítulo se proporcionarán gráficos adecuados y un intervalo de confianza del 95 % para el tiempo que tarda el proyecto en atender las solicitudes de las pruebas funcionales en dos equipos diferentes, junto a la optimización de código mediante el uso del @index en las entidades, además de un contraste de hipótesis de confianza del 95 % con respecto a cuál es el equipo más potente y que proporcione mayor rendimiento.

**Intervalo de confianza**

Para este apartado se han recolectado 868 datos de los distintos tests (filas del excel-clean) y se ha calculado el **intervalo de confianza con un nivel del 95%** en dos máquinas distintas. Los recursos hardware de dichas máquinas pueden contemplarse en las siguientes tablas.

|  |  |  |
| --- | --- | --- |
| COMPONENTES | PC 1 | PC 2 |
| CPU | Intel Core i5-8250U | Intel Core i7-13620H |
| RAM | 8 GB DDR3 | 32 GB |
| MEMORIA | 120 GB SSD | 500GB M.2 |
| GRÁFICA | Nvidia GeForce MX130 | Intel IrisXe Graphics |

Para los test ejecutados en el PC1, se obtuvo resultados siguiendo la metodología mediante un excel, al cual se le pasaron los datos sacados del test.trace tras aplicar el replayer. Obteniendo:

Tabla

Descripción generada automáticamente

Como puede observarse son tiempos bastante buenos. Sin embargo, aquí entrarían los requerimientos del cliente, teniendo que hacer análisis mucho más exhaustivos, debido a que, si no se cumple que los intervalos de tiempo sean menores a los exigidos por el cliente, se debería analizar y corregir el rendimiento del proyecto. Para facilitar el saber donde tarda más en dar respuesta se ha realizado una gráfica para visualizar el tiempo de respuesta de cada feature:

Gráfico, Gráfico de barras

Descripción generada automáticamente

Para analizar el gráfico primero hay que destacar que posee algunas columnas que se usan para realizar las pruebas, como puede ser el uso de registrarte en la aplicación o listar de sponsor, pero que no nos sirven para analizar el rendimiento de training module o training sessions, por tanto, se descartan.

Cómo se puede observar en el gráfico casi todas las features tienen un tiempo de respuesta bastante parecido, sin embargo, la que más tarda en responder es sin duda el crear un training module, siendo esta el MIR (most inefficient request). Y por contra el que menos tarda en ejecutar es el mostrar los detalles de un training session.

Para los test ejecutados en el PC2, se obtuvo resultados siguiendo la metodología mediante un excel, al cual se le pasaron los datos sacados del test.trace tras aplicar el replayer. Antes de realizar dichos test, se añadió los índices con el @Table para obtener una mayor eficiencia. Se obtuvo:

Tabla

Descripción generada automáticamente

Quedando ambos juntos tal que:

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamente

Como puede observarse hay una mejora bastante notable del rendimiento, esto debido a la mejora en el hardware principalmente, junto a una mejora notable en el software, gracias a los index.

**Contraste de hipótesis**

El objetivo de este contraste de hipótesis es saber si aplicando los cambios del cambio de hardware junto a los índices en las tablas tiene un cambio realmente significativo o no. Para ello implementamos en el excel una prueba de Z-test sobre las columnas de los tiempos, para así comparar realmente el impacto del cambio. El valor alfa utilizado ha sido de 0,05, a dicho alpha se le compara al valor crítico de z (dos colas) que en mi caso es 3,75652E-08 (0.0000000375652). Para saber si el cambio ha sido significativo, el valor p-value deberá estar entre [0.00,alfa), este es nuestro caso. Por tanto, el cambio ha resultado en una mejora del rendimiento. Este ejercicio ayuda a darse cuenta de lo importante que es tener un buen equipo hardware, junto a las pequeñas mejoras que se pueden hacer a nivel software que afectan de manera sustancial a nuestros proyectos. Tras esto cabe resaltar que la diferencias de equipos hardware es claramente notable para que en este ejercicio se notase dicho cambio, con tan solo añadir los index puede no resultar en un cambio tan notable como este. Si nuestro valor hubiese estado por encima de alfa, (alfa,1.00], los cambios realizados no serían mejoras significativas. Y si el valor hubiese estado muy cercano a alfa, nuestras pruebas no serían concluyentes. Si nuestro cliente tras estas pruebas hubiese exigido mejorar más el rendimiento, entonces deberíamos mejorar el MIR junto a aplicar más mejoras y volver a realizar este proceso hasta que nuestro p-value sea el exigido por el cliente.

Tabla

Descripción generada automáticamente

Los cálculos detallados del análisis estadístico de este capítulo pueden comprobarse en los documentos adjunto a esta entrega llamados:

* ***Tester-performance-clean.xlsx***
* ***tester-performance-clean-optimizado.xlsx***
* ***Test-Comparados-Z-test.xlsx***

# Bibliografía

-Documento 08 Annexes, de los contenidos de la plataforma virtual de esta asignatura.

-S02 Performance testing, transparencias del módulo de testing L04 de la asignatura