Universidad de Sevilla  
Escuela Técnica Superior de Ingeniería Informática

TESTING REPORT INDIVIDUAL - D04



**C1.027**

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

**Autor:**

Devós Bono, Agustín

agudevbon@alum.us.es

**Fecha:** 22/05/2024

**Tabla de contenidos**

[**Resumen**](#_heading=h.gjdgxs) **2**

[**Tabla de revisión**](#_heading=h.30j0zll) **2**

[**Introducción**](#_heading=h.1fob9te) **2**

[**Contenidos**](#_heading=h.3znysh7) **3**

**Capítulo I: Testing funcional 3**

**Capítulo II: Testing de rendimiento 4**

[**Bibliografía**](#_heading=h.3dy6vkm) **4**

# Resumen

Este documento se detalla el proceso de testing seguido sobre 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 | 22/05/2024 | Creación y desarrollo del documento |

# Introducción

A continuación, veremos separados en dos capítulos, los métodos estadísticos realizados para analizar los resultados tras el testing formal de los requisitos de todas las entregas del proyecto.

# Contenidos

# Capítulo I: Testing funcional

Este capítulo mostrará el listado con los casos de prueba implementados, agrupados por *features*. Para cada funcionalidad a probar se explicarán los casos positivos y negativos probados en los archivos .safe y los intentos de hackeo realizados en los archivos .hack.

**Entidad Project**

**manager/project/create:** Para probar esta funcionalidad se han probado todos los casos positivos y negativos de cada campo del form. En primer lugar, se ha enviado el formulario vacío y después se han probado todos los casos negativos de cada campo, tras esto, se ha creado un objeto distinto para cada posible valor válido de cada campo. Para la prueba de hacking se ha intentado acceder al formulario desde un rol distinto a manager obteniendo el correspondiente error 500.

**manager/project/delete:** Para realizar esta prueba, se ha probado a eliminar un proyecto no publicado sin historias de usuario asignadas y otro proyecto no publicado con historias de usuario asignadas, para comprobar el correcto funcionamiento del borrado de las asignaciones de la entidad intermedia. Para la prueba de hacking se ha intentado eliminar con el mismo usuario, un proyecto inexistente y un proyecto ya publicado. Con manager2 se ha intentado eliminar un proyecto de manager1 y, por último, con el rol Sponsor se ha intentado eliminar un proyecto. Todas estas pruebas han devuelto correctamente un error 500 not authorised.

* **Explicación lineas rojas coverage:** En el service ManagerProjectDeleteService.java se implementa un unbind, sin embargo como este servicio no tiene validaciones nunca se ejecuta dicho unbind por lo que dicho método no está cubierto por los tests.

**manager/project/list:** Para realizar esta prueba, se ha accedido al listado de proyectos un rol manager comprobando que se lista adecuadamente, completando la prueba positiva de dicha *feature*. Para la prueba de hacking, se ha intentado listar los proyectos de un manager desde el rol sponsor, recibiendo el correspondiente error de acceso.

**manager/project/publish:** Para comprobar el correcto comportamiento de esta funcionalidad y sus restricciones se han realizado las siguientes pruebas:

* Publicar satisfactoriamente un proyecto con una historia de usuario asignada y publicada.
* Publicar satisfactoriamente un proyecto con varias historias de usuario asignadas y publicadas.
* Publicar un proyecto con la indicación de errores fatales activa y recibir el correspondiente aviso de, no posible.
* Publicar un proyecto con varias historias de usuario asignadas y no publicadas, y recibir el correspondiente aviso de, no posible.
* Publicar un proyecto sin historias de usuario asignadas, y recibir el correspondiente aviso de, no posible.

Para la prueba de hacking se ha intentado publicar con el mismo usuario, un proyecto inexistente y un proyecto ya publicado. Con manager2 se ha intentado publicar un proyecto de manager1 y, por último, con el rol Sponsor se ha intentado publicar un proyecto. Todas estas pruebas han devuelto correctamente un error 500 not authorised.

* **Explicación lineas amarillas coverage:** En el service ManagerProjectPublishService.java se implementan ciertas validaciones para mostrar mensajes de error si las condiciones de publicación no son las correctas, de esta forma el usuario recibe feedback en la pantalla en lugar de recibir un erro 500. Es por esto que las lineas:

if (!super.getBuffer().getErrors().hasErrors(“draftMode”))

if (!super.getBuffer().getErrors().hasErrors(“indication”))

no están probadas por todos los caminos posibles del if, ya que los atributos draftMode e indication tienen valor por defecto por lo que no pueden tener errores en su campo.

**manager/project/show:** Para realizar esta prueba, se ha accedido al listado de proyectos para entrar a los detalles de un proyecto publicado y después los detalles de un proyecto no publicado, comprobando que se muestra adecuadamente, completando la prueba positiva de dicha *feature*. Para la prueba de hacking se ha intentado mostrar con el mismo usuario, un proyecto inexistente. Con manager2 se ha intentado mostrar un proyecto de manager1 y, por último, con el rol Sponsor se ha intentado mostrar un proyecto. Todas estas pruebas han devuelto el correspondiente error 500 de no autorizado.

**manager/project/update:** Para probar esta funcionalidad se han probado todos los casos positivos y negativos de cada campo del form. En primer lugar, se ha enviado el formulario vacío y después se han probado todos los casos negativos de cada campo, tras esto, se ha actualizado el proyecto una vez para cada posible valor válido de cada campo. Para la prueba de hacking se ha intentado actualizar con el mismo usuario, un proyecto inexistente y un proyecto ya publicado. Con manager2 se ha intentado actualizar un proyecto de manager1 y, por último, con el rol Sponsor se ha intentado actualizar un proyecto. Todas estas pruebas han devuelto correctamente un error 500 not authorised.

**Entidad userStory**

**manager/user-story/create:** Para probar esta funcionalidad se han probado todos los casos positivos y negativos de cada campo del form. En primer lugar, se ha enviado el formulario vacío y después se han probado todos los casos negativos de cada campo, tras esto, se ha creado un objeto distinto para cada posible valor válido de cada campo. Para la prueba de hacking se ha intentado acceder al formulario desde un rol distinto a manager obteniendo el correspondiente error 500

**manager/user-story/delete:** Para realizar esta prueba, se ha probado a eliminar una historia de usuario no publicada que no esté asignada a ningún proyecto y otra historia de usuario no publicada que esté asignada a algún proyecto, para comprobar el correcto funcionamiento del borrado de las asignaciones de la entidad intermedia. Para la prueba de hacking se ha intentado eliminar con el mismo usuario, una historia de usuario inexistente y una historia de usuario ya publicado. Con manager2 se ha intentado eliminar una historia de usuario de manager1 y, por último, con el rol Sponsor se ha intentado eliminar una historia de usuario. Todas estas pruebas han devuelto correctamente un error 500 not authorised.

* **Explicación lineas rojas coverage:** En el service ManagerUserStoryDeleteService.java se implementa un unbind, sin embargo como este servicio no tiene validaciones nunca se ejecuta dicho unbind por lo que dicho método no está cubierto por los tests.

**manager/user-story/list-by-project:** Para realizar esta prueba, se ha accedido a los detalles de un proyecto para después acceder al listado de historias de usuario asignadas a dicho proyecto comprobando que se lista adecuadamente, completando la prueba positiva de dicha *feature*. Para la prueba de hacking se ha intentado listar con el mismo usuario, las historias de usuario de un proyecto inexistente. Con manager2 se ha intentado listar las historias de usuario de un proyecto de manager1 y, por último, con el rol Sponsor se ha intentado listar las historias de usuario de un proyecto. Todas estas pruebas han devuelto correctamente un error 500 not authorised.

**manager/user-story/list-mine:** Para realizar esta prueba, se ha accedido al listado de historias de usuario de un rol manager comprobando que se lista adecuadamente, completando la prueba positiva de dicha *feature*. Para la prueba de hacking, se ha intentado listar las historias de usuario de un manager desde el rol sponsor, recibiendo el correspondiente error de acceso.

**manager/user-story/publish:** Para realizar esta prueba, se ha probado a publicar una historia de usuario no asignada a ningún proyecto y otra asignada a algún proyecto. Para la prueba de hacking se ha intentado publicar con el mismo usuario, una historia de usuario inexistente y una historia de usuario ya publicada. Con manager2 se ha intentado publicar una historia de usuario de manager1 y, por último, con el rol Sponsor se ha intentado publicar una historia de usuario. Todas estas pruebas han devuelto correctamente un error 500 not authorised.

**manager/user-story/show:** Para realizar esta prueba, se ha accedido al listado de historias de usuario para entrar a los detalles de una historia de usuario publicada y después los detalles de una historia de usuario no publicada, comprobando que se muestra adecuadamente, completando la prueba positiva de dicha *feature*. Para la prueba de hacking se ha intentado mostrar con el mismo usuario, una historia de usuario inexistente. Con manager2 se ha intentado mostrar una historia de usuario de manager1 y, por último, con el rol Sponsor se ha intentado mostrar una historia de usuario. Todas estas pruebas han devuelto el correspondiente error 500 de no autorizado.

**manager/user-story/update:** Para probar esta funcionalidad se han probado todos los casos positivos y negativos de cada campo del form. En primer lugar, se ha enviado el formulario vacío y después se han probado todos los casos negativos de cada campo, tras esto, se ha actualizado la historia de usuario una vez para cada posible valor válido de cada campo. Para la prueba de hacking se ha intentado actualizar con el mismo usuario, una historia de usuario inexistente y una historia de usuario ya publicada. Con manager2 se ha intentado actualizar una historia de usuario de manager1 y, por último, con el rol Sponsor se ha intentado actualizar una historia de usuario. Todas estas pruebas han devuelto correctamente un error 500 not authorised.

**Entidad Intermedia projectUserStory**

**manager/project-user-story/create:** Para realizar esta prueba, se ha entrado al listado de relaciones con historias de usuario de un proyecto para intentar, crear una relación con una historia de usuario no publicada y otra relación con una historia de usuario publicada, para comprobar el correcto funcionamiento del creado de las asignaciones de la entidad intermedia. Para la prueba de hacking se ha intentado crear con el mismo usuario, una relación para un proyecto inexistente y una relación para un proyecto ya publicado. Con manager2 se ha intentado crear una relación de manager1 y, por último, con el rol Sponsor se ha intentado entrar al formulario de creación de una relación. Todas estas pruebas han devuelto correctamente un error 500 not authorised.

* **Explicación lineas amarillas coverage:** Para comprobar que no se asigna una historia de usuario a un proyecto de un manager distinto se ha implementado una validación que comprueba dicho caso, sin embargo, no he conseguido realizar esta prueba mediante el navegador, ya que no se podía introducir en el selector una historia de usuario de otro manager y que lo reconociese correctamente. A pesar de esto se ha decidido dejar la validación para evitar este comportamiento mediante un post desde postman.

**manager/project-user-story/delete:** Para realizar esta prueba, se ha entrado al listado de relaciones con historias de usuario de un proyecto para intentar, eliminar una relación con una historia de usuario no publicada y otra relación con una historia de usuario publicada, para comprobar el correcto funcionamiento del borrado de las asignaciones de la entidad intermedia. Para la prueba de hacking se ha intentado eliminar con el mismo usuario, una relación inexistente y una relación de un proyecto ya publicado. Con manager2 se ha intentado eliminar una relación de manager1 y, por último, con el rol Sponsor se ha intentado eliminar una relación. Todas estas pruebas han devuelto correctamente un error 500 not authorised.

* **Explicación lineas rojas coverage:** En el service ManagerProjectUserStoryDeleteService.java se implementa un unbind, sin embargo como este servicio no tiene validaciones nunca se ejecuta dicho unbind por lo que dicho método no está cubierto por los tests.

**manager/project-user-story/list-by-project:** Para realizar esta prueba, se ha accedido a los detalles de un proyecto para después acceder al listado de relaciones con historias de usuario asignadas a dicho proyecto comprobando que se lista adecuadamente, completando la prueba positiva de dicha *feature*. Para la prueba de hacking se ha intentado listar con el mismo usuario, las relaciones de un proyecto inexistente. Con manager2 se ha intentado listar las historias de usuario de un proyecto de manager1 y, por último, con el rol Sponsor se ha intentado listar las historias de usuario de un proyecto. Todas estas pruebas han devuelto correctamente un error 500 not authorised.

**manager/project-user-story/show:** Para realizar esta prueba, se ha accedido al listado relaciones de un proyecto para entrar a los detalles de una relación, comprobando que se muestra adecuadamente, completando la prueba positiva de dicha *feature*. Para la prueba de hacking se ha intentado mostrar con el mismo usuario, una relación inexistente. Con manager2 se ha intentado relación de usuario de manager1 y, por último, con el rol Sponsor se ha intentado mostrar una historia de usuario. Todas estas pruebas han devuelto el correspondiente error 500 de no autorizado.

**Bugs encontrados gracias a los tests:** Gracias a la creación y ejecución de las pruebas se han identificado los siguientes bugs/fallos:

* Publicación de historias de usuario y de proyectos: en la primera implementación de este servicio, se desarrollo el funcionamiento de tal forma que solo publicaba y no actualizaba. Sin embargo, si cambiabas un campo que hiciera saltar una validación y le dabas a publicar, en la base de datos solo se aplicaba el publish. A pesar de esto la validación saltaba en el form y bloqueaba la vista de esta. Para solucionar este bloqueo de la vista se implemento el publish como un update pero que también establezca draftMode a false.
* Validaciones tipo money ProjectUpdateService: las validaciones del tipo Money en el servicio estaban atrasadas y permitían introducir valores de coste no válidos. Tras implementar las validaciones correctas se solventó este problema.

Para comprobar los detalles sobre el tiempo de ejecución y los resultados de las pruebas realizadas, así como los datos originales de los cuales se ha extraído la información de este capítulo, puede consultar el documento adjunto llamado ***tester-performance.xlsx*** enviado junto a la entrega ***D04: testing performance***.

Por otro lado, las pruebas realizadas en modo ***cobertura*** han mostrado un porcentaje de cobertura adecuado a las pruebas realizadas, el cual quedaría resumido en las siguientes capturas.

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente

# 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, además de un contraste de hipótesis de confianza del 95 % con respecto a cuál es el equipo más potente.

**Intervalo de confianza**

Para este apartado se han recolectado 1320 datos de los distintos tests y se ha calculado el **intervalo de confianza con un nivel del 95%** en el mismo dispositivo, pero implementando mejoras de rendimiento en el código.

Tras ejecutar los tests en primer lugar con el código base se consiguió la cobertura indicada anteriormente, podemos observar los resultados en cuanto a tiempo de consulta en la Columna 1 de la imagen inferior. Para tratar de mejorar el tiempo de respuesta en la segunda batería de tests se añadió índices “*@Table”* en las entidades *Project, UserStory* y *ProjectUserStory*

Tabla

Descripción generada automáticamente

Los tests ejecutados en la primera batería de tests mostraron que la función: **crear una nueva relación entre un proyecto y una historia de usuario era la que más tardaba en ejecutarse (MIR)**, tal y como muestra el siguiente gráfico del tiempo (en ms) de cada feature probada.

Para tratar de mejorar el tiempo de respuesta en la segunda batería de tests se añadió índices “*@Table”* en las entidades *Project, UserStory* y *ProjectUserStory*. Vemos a continuación los resultados obtenidos tras aplicar este cambio para la mejora de rendimiento de la aplicación.

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamente

Para ver los resultados y tiempos de ejecución en mayor detalle puede acceder al archivo **tester-performance2.xlsx** adjunto en la entrega

Se puede observar una reducción de la media de los tiempos de carga de más de la mitad gracias a la implementación de índices de @Table en las entidades del estudiante.

Analicemos ahora también el gráfico de los tiempos de respuesta promedio, podemos observar también una reducción de los tiempos de carga muy notable. Tras la implementación de dicho cambio ahora **la consulta más ineficiente (MIR) es el borrado de proyectos.**

Comparación en paralelo de los análisis de datos para ambas pruebas:

Tabla

Descripción generada automáticamente

**Contraste de hipótesis**

Para comparar adecuadamente los intervalos de confianza calculados en ambas máquinas tras ejecutar las pruebas, se ha considerado realizar la prueba Z-Test sobre las columnas de tiempos generados por ambas máquinas (antes y después), quedándonos con el valor del campo de la tabla generada llamado ***Valor crítico de z (dos colas)*** ***= 0.***

Tabla

Descripción generada automáticamente

Dicho valor crítico de z nos indica que nos encontramos en el intervalo *( a - 1.00 ]*, donde, ***a*** *= alpha = 1 - Intervalo de Confianza = 1 - 0.95 =* ***0.05***

Al ser z = 0 tenemos que: *0 =< z < a*

Tras la obtención de estos datos podemos concluir que incluir índices ha reducido de forma muy notable el tiempo de ejecución de los tests. Sorprende la gran diferencia de tiempo de ejecución por lo que añado a continuación una hipótesis del por qué:

Podemos observar en la primera gráfica de barras que las operaciones que más tardan son:

* Borrar proyecto:
* Borrar relación proyecto-historia de usuario
* Crear relación proyecto- historia de usuario
* Borrar historia de usuario

Identificamos que todas estas consultas crean o borran algún objeto de la entidad *ProjectUserStory,* al haber implementado @Index en los dos únicos atributos se ha acelerado toda consulta de creación o borrado de los objetos de esta entidad. Estas 4 consultas eran los picos más altos y con mayor diferencia con respecto al resto de promedios. Al regularse y estandarizarse el promedio de tiempo de las consultas la varianza baja en gran cantidad lo que ha dado lugar, sumado con la bajada de tiempo de consulta base de el resto de entidades tras implementar los índices, ha concluido en una gran implementación de mejora de rendimiento.

Tras los resultados obtenidos, no ha sido necesario realizar un estudio del rendimiento del dispositivo puesto que ya arrojaba buenos valores en el tiempo de ejecución de las pruebas.

Los cálculos detallados del análisis estadístico de este capítulo puede comprobarse en el documento adjunto a esta entrega llamado ***tester-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