Universidad de Sevilla  
Escuela Técnica Superior de Ingeniería Informática

**Testing Report**



Grado en Ingeniería Informática – Ingeniería del Software  
Diseño y Pruebas 2.

Curso 2023 – 2024

| **Fecha** | **Versión** |
| --- | --- |
| 20/05/2024 | v1.0.0 |

| **Grupo: C1.011** | | |
| --- | --- | --- |
| **Autores por orden alfabético** | **Rol** | **Descripción del rol** |
| Abouri, Mohamed - Y7156458E | Developer | Persona encargada de desarrollar el código. |
| Cabello Ranea, Rafael -32094496C | Operador | Encargado de las tareas de campo, de las instalaciones y del mantenimiento de los sistemas de la empresa. |
| Calderón Rodríguez, Francisco Javier - 78233219F | Project Manager | Persona encargada de tomar decisiones de diseño y vigilar el correcto desarrollo |
| Delgado Pallares, David - 29519510E | Tester | Persona encargada de realizar pruebas sobre el código. |
| Ramírez Lara, Iván- 77852382G | Developer | Persona encargada de desarrollar el código. |

**Control de Versiones**

| **Fecha** | **Versión** | **Descripción** |
| --- | --- | --- |
| 20/05/2024 | v1.0.0 | Desarrollo de la primera versión. |
|  |  |  |
|  |  |  |
|  |  |  |

**Índice de contenido**

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

2[.](#_heading=h.gjdgxs) Testeo funcional 2

3[. Testeo de re](#_heading=h.30j0zll)ndimiento 10

4. Cobertura 15

1. **Introducción**

Este documento presenta dos capítulos esenciales: pruebas funcionales, con un listado de casos de prueba organizados por característica y su efectividad en detectar errores, y pruebas de rendimiento, con gráficos e intervalos de confianza del 95% para el tiempo de respuesta en dos computadoras, además de un análisis para determinar cuál es más potente. Estos capítulos son cruciales para evaluar la calidad y el rendimiento del software.

1. **Testeo funcional**

En esta sección se detallarán todas las pruebas que hemos realizado para testear el funcionamiento de nuestro software. Se explicará qué es lo que se ha hecho, el resultado esperado, el resultado real (que hace referencia al resultado del test corregido después de encontrar bugs que hicieran que el código estuviera mal) y, por último, la columna de bugs encontrados. Si ha habido bugs, se ha arreglado el test, por lo que el resultado real ya no los contiene.

Descripción de la prueba: Se detallará cada prueba realizada, incluyendo los pasos específicos llevados a cabo.

Resultado esperado: Se especificará cuál era el comportamiento o el resultado que se esperaba obtener con la prueba.

Resultado real: Se presentará el resultado obtenido tras la ejecución del test, ya corregido después de encontrar y solucionar cualquier bug que afectará el código.

Bugs encontrados: Se listarán los bugs detectados durante la prueba y las correcciones realizadas. Una vez solucionados los bugs, se actualizará el resultado real para reflejar el funcionamiento correcto del software.

**List**

| **Entidad** | **Descripción** | **Resultado Esperado** | **Resultado Real** | **Bugs detectados** |
| --- | --- | --- | --- | --- |
| **project safe** | Verificar que un manager puede listar sus proyectos. | El sistema deberá mostrar los proyectos asociados al manager | El sistema ha mostrado los proyectos asociados al manager | Ninguno |
| **project hack** | Verificar que un rol no manager no puede listar proyectos. | El sistema no debe mostrar los proyectos, usando un rol no manager, dando un error de pánico | El sistema ha dado un error 500 de no autorizado en cada rol no manager. | Ninguno |
| **user story safe** | Comprobar que un manager puede listar las historias de usuario de los proyectos. | El sistema deberá mostrar las historias de usuario asociadas a un proyecto del manager | El sistema ha mostrado todas las historias de usuario asociadas a cada uno de los proyectos probados | Ninguno |
| **user story hack** | Comprobar que un rol no manager no puede listar historias de usuario asociadas a un proyecto determinado. | El sistema no debe mostrar las historias de usuario asociado a un proyecto, usando un rol no manager, dando un error de pánico | El sistema ha dado un error 500 de no autorizado en cada rol no manager. También ha dado error cuando se ha intentado acceder a las user stories de un proyecto con un manager diferente. | Ninguno |

**List-All**

| **user story safe** | Comprobar que un manager puede listar las historias de usuario asociadas a este. | El sistema deberá mostrar las historias de usuario asociadas al manager. | El sistema ha mostrado todas las historias de usuario asociadas al manager | Ninguno |
| --- | --- | --- | --- | --- |
| **user story hack** | Comprobar que un rol no manager no puede listar las historias de usuario. | El sistema no debe mostrar las historias de usuario asociadas al manager. Dando error de pánico | El sistema ha dado un error 500 de no autorizado en cada rol no manager. También ha dado error cuando se ha intentado acceder a las user stories de un proyecto con un manager diferente. | Al hacer por primera vez la prueba si se podía ver las historias de usuario de otro manager, por lo que se podía hacer **get hacking**.  Ahora ya está corregido, así que da error 500. |

**Show**

| **Entidad** | **Descripción** | **Resultado Esperado** | **Resultado Real** | **Bugs detectados** |
| --- | --- | --- | --- | --- |
| **project safe** | Verificar que un manager puede ver toda la información de un proyecto. | El sistema deberá mostrar la información de los proyectos asociados al manager | El sistema ha mostrado la información de los proyectos asociados al manager | Ninguno |
| **project hack** | Verificar que un rol no manager no puede ver la información de los proyectos. | El sistema no debe mostrar la información de proyectos, usando un rol no manager, dando un error de pánico | El sistema ha dado un error 500 de no autorizado en cada rol no manager. | Ninguno |
| **user story safe** | Comprobar que un manager puede ver la información asociada a las historias de usuario. | El sistema deberá mostrar la información de las user stories asociados al manager. | El sistema ha mostrado toda la información de las historias de usuario elegidas y que están asociadas al manager | Ninguno |
| **user story hack** | Comprobar que un rol no manager no puede ver la información asociada a las historias de usuario. | El sistema no debe mostrar las historias de usuario, usando a un rol no manager, dando un error de pánico | El sistema ha dado un error 500 de no autorizado en cada rol no manager. Al intentar acceder a la información de una historia de usuario. Tambíen lo ha dado si la historia de usuario no pertenece a el manager utilizado | Ninguno |

**Create**

| **Entidad** | **Descripción** | **Resultado Esperado** | **Resultado Real** | **Bugs detectados** |
| --- | --- | --- | --- | --- |
| **project safe** | Verificar que un manager puede crear proyectos de forma correcta. | El sistema deberá permitir crear proyectos válidos. | El sistema ha dejado crear proyectos con datos válidos. | Ninguno |
| **project hack** | Verificar que un rol no manager no puede acceder a la creación de proyectos. | El sistema debe de lanzar error de pánico si un rol no manager intenta acceder a la URL o si se hace una petición sin ser manager. | El sistema ha dado un error 500 de no autorizado en cada rol no manager. | Ninguno |
| **user story safe** | Comprobar que un manager puede crear historia de usuario de forma correcta. | El sistema deberá permitir crear historias de usuarios válidas. | El sistema ha permitido crear historias de usuario válidas. | Ninguno |
| **user story hack** | Revisar que un rol no manager no puede acceder a la creación de historias de usuario. | El sistema debe de lanzar error de pánico si un rol no manager intenta acceder a la URL o si se hace una petición sin ser manager. | El sistema ha dado un error 500 de no autorizado en cada rol no manager. Al intentar acceder a la información de una historia de usuario. También lo ha dado si la historia de usuario no pertenece al manager utilizado | Ninguno |
| **project-user-story safe** | Revisar que un manager puede añadir historias de usuario a un proyecto. | El sistema deberá permitir añadir historias de usuario a proyectos no publicado. | El sistema ha permitido añadir historias de usuario a proyectos no publicado. | Ninguno |
| **project-user-story hack** | Revisar que un rol no manager no puede añadir historias de usuario a un proyecto y que un manager no puede asignar historias de usuarios a proyectos de otro manager. | El sistema debe de lanzar error de pánico si un rol no manager intenta acceder a la URL o si se hace una petición sin ser manager. | El sistema no permite acceder a la url y da error 500 si se intenta asignar una historia de usuario a un proyecto de otro manager. | Se descubrió que el sistema era débil antes post hacking ya que se podía añadir una historia de usuario a un proyecto de otro manager. |

**Publish**

| **Entidad** | **Descripción** | **Resultado Esperado** | **Resultado Real** | **Bugs detectados** |
| --- | --- | --- | --- | --- |
| **project safe** | Verificar que un manager puede publicar proyectos válidos de forma correcta. | El sistema deberá permitir publicar proyectos válidos. | El sistema ha dejado publicar proyectos con datos válidos. | Se descubrió que se podía publicar un proyecto con un código duplicado y esto causaba **error 500**. |
| **project hack** | Verificar que un rol no manager u otro manager no puede acceder a la publicación de proyectos. | El sistema debe de lanzar error de pánico si un rol no manager intenta acceder a la URL, si se hace una petición sin ser manager o si se hace desde un manager diferente | El sistema ha dado un error 500 de no autorizado en cada rol no manager. También ha dado error 500 si un manager intentaba acceder a un proyecto de otro manager | Ninguno |
| **user story safe** | Comprobar que un manager puede publicar una historia de usuario de forma correcta. | El sistema deberá permitir publicar historias de usuarios válidas. | El sistema ha permitido publicar las historias de usuario elegidas. | Ninguno |
| **user story hack** | Comprobar que un rol no manager no puede acceder a la publicación de historias de usuario. | El sistema debe de lanzar error de pánico si un rol no manager intenta acceder a la URL o si se hace una petición sin ser manager. | El sistema ha dado un error 500 de no autorizado en cada rol no manager. Al intentar acceder a la información de una historia de usuario. También lo ha dado si la historia de usuario no pertenece al manager utilizado | Ninguno |

**Delete**

| **Entidad** | **Descripción** | **Resultado Esperado** | **Resultado Real** | **Bugs detectados** |
| --- | --- | --- | --- | --- |
| **project safe** | Verificar que un manager puede borrar proyectos de forma correcta. | El sistema deberá permitir eliminar proyectos que no estén publicados. | El sistema ha dejado eliminar proyectos no publicados. | Se ha detectado un **error 500** al borrar los proyectos. El error es causado por una relación con una entidad implementada por el estudiante 5. |
| **project hack** | Verificar que un rol no manager no puede acceder a la eliminación de proyectos. | El sistema debe de lanzar error de pánico si un rol no manager intenta acceder a la URL o si se hace una petición sin ser manager. | El sistema ha dado un error 500 de no autorizado en cada rol no manager y en manager no autorizados. | Ninguno |
| **user story safe** | Comprobar que un manager puede eliminar historia de usuario de forma correcta. | El sistema deberá permitir eliminar historias de usuarios no publicadas. | El sistema ha permitido eliminar historias de usuario válidas. | Ninguno |
| **user story hack** | Revisar que un rol no manager no puede acceder a la eliminación de historias de usuario. | El sistema debe de lanzar error de pánico si un rol no manager intenta acceder a la URL o si se hace una petición sin ser manager. | El sistema ha dado un error 500 de no autorizado en cada rol no manager. Al intentar acceder a la información de una historia de usuario. También lo ha dado si la historia de usuario no pertenece al manager utilizado | Ninguno |
| **project-user-story safe** | Revisar que un manager puede eliminar historias de usuario de un proyecto. | El sistema deberá permitir eliminar historias de usuario de proyectos no publicado. | El sistema ha permitido eliminar historias de usuario de proyectos no publicado. | Se descubrió que si no se asignaba proyecto el sistema daba **error 500** porque se pedía en él, validate la id de un proyecto inexistente. |
| **project-user-story hack** | Comprobar que un rol no manager no puede eliminar historias de usuario de un proyecto y que un manager no puede asignar historias de usuarios a proyectos de otro manager. | El sistema debe de lanzar error de pánico si un rol no manager intenta acceder a la URL o si se hace una petición sin ser manager. | El sistema no permite acceder a la url y da error 500 si se intenta asignar una historia de usuario a un proyecto de otro manager. | Ninguno |

**Update**

| **Entidad** | **Descripción** | **Resultado Esperado** | **Resultado Real** | **Bugs detectados** |
| --- | --- | --- | --- | --- |
| **project safe** | Verificar que un manager puede actualizar proyectos válidos de forma correcta. | El sistema deberá permitir actualizar proyectos válidos. | El sistema ha dejado actualizar proyectos con datos válidos. | Ninguno |
| **project hack** | Verificar que un rol no manager u otro manager no puede acceder a la actualización de proyectos. | El sistema debe de lanzar error de pánico si un rol no manager intenta acceder a la URL, si se hace una petición sin ser manager o si se hace desde un manager diferente | El sistema ha dado un error 500 de no autorizado en cada rol no manager. También ha dado error 500 si un manager intentaba acceder a un proyecto de otro manager | Ninguno |
| **user story safe** | Comprobar que un manager puede actualizar una historia de usuario de forma correcta. | El sistema deberá actualizar historias de usuarios válidas. | El sistema ha permitido actualizar las historias de usuario elegidas. | Ninguno |
| **user story hack** | Comprobar que un rol no manager no puede acceder a la actualización de historias de usuario. | El sistema debe de lanzar error de pánico si un rol no manager intenta acceder a la URL o si se hace una petición sin ser manager. | El sistema ha dado un error 500 de no autorizado en cada rol no manager. Al intentar acceder a la información de una historia de usuario. También lo ha dado si la historia de usuario no pertenece al manager utilizado | Ninguno |

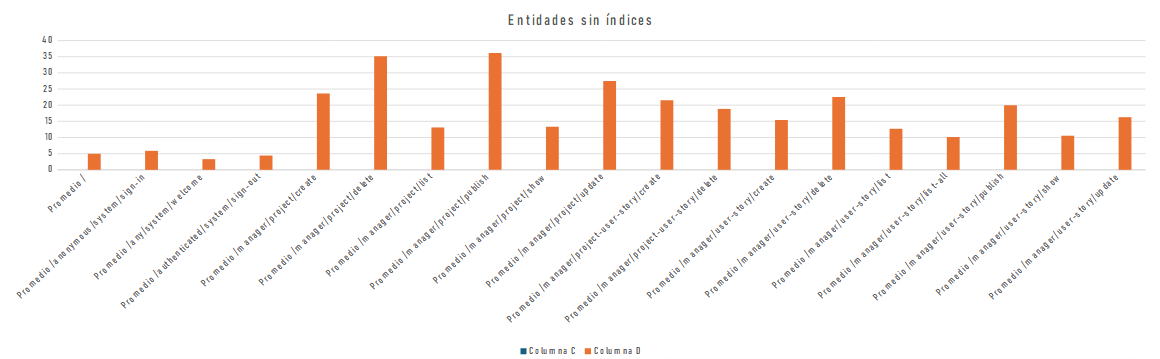
1. **Testeo de rendimiento**

En este capítulo, se evaluará el rendimiento de nuestro proyecto mediante pruebas detalladas que analizan el tiempo de respuesta al atender las solicitudes en nuestras pruebas funcionales. Utilizaremos dos muestras, una de ellas estará compuesta por los resultados obtenidos por hacer realplayer desde coverage, sobre las entidades con índices y sin estos. La segunda muestra se hará de la misma forma, pero desde el apartado de debug.

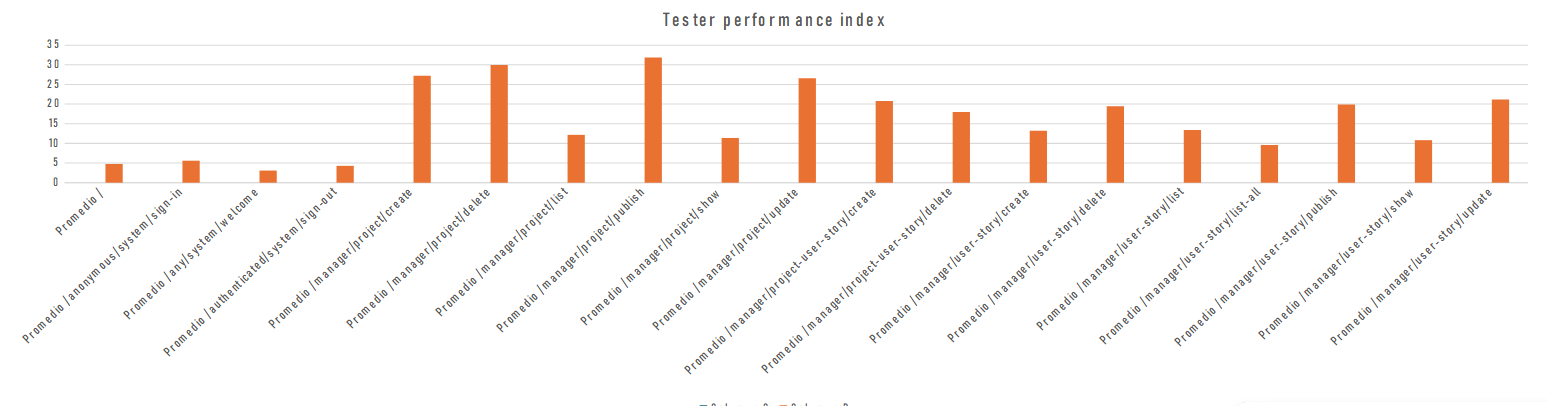
Para realizar el análisis de estas pruebas, presentaremos gráficos que ilustran los resultados obtenidos. Además, calcularemos intervalos de confianza del 95% para los tiempos de respuesta medidos y realizaremos un contraste de hipótesis, también con un intervalo de confianza del 95%, para determinar la diferencia de hacer realplayer sobre entidades con índices o entidades sin estos.

**Muestra sobre resultados desde Coverage.**

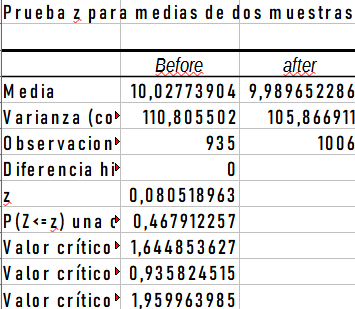
**Gráficas de entidades sin índices:**

****

**Gráficas de entidades con índices:**

****

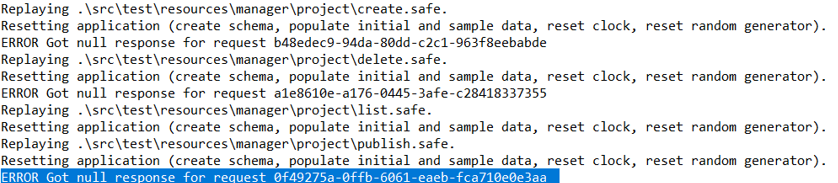
**Comparación de resultados:**

****

En nuestras pruebas de rendimiento, hemos obtenido un valor crítico de z de **0.9358** para un nivel de significancia (α) de **0.95**. Este valor se utiliza para determinar si las diferencias en los tiempos de respuesta entre las dos pruebas son estadísticamente significativas.

De acuerdo con nuestra metodología, si el p-valor se encuentra en el intervalo (α, 1.00], esto indica que los cambios no resultaron en mejoras relevantes; es decir, aunque los tiempos de muestra sean diferentes, globalmente son equivalentes. Dado que nuestro valor crítico de z está en este intervalo, concluimos que no hay una mejora significativa en el rendimiento al comparar los tiempos de respuesta de las pruebas. En resumen, las diferencias observadas no son lo suficientemente grandes como para considerarse estadísticamente relevantes con el nivel de confianza del 95%.

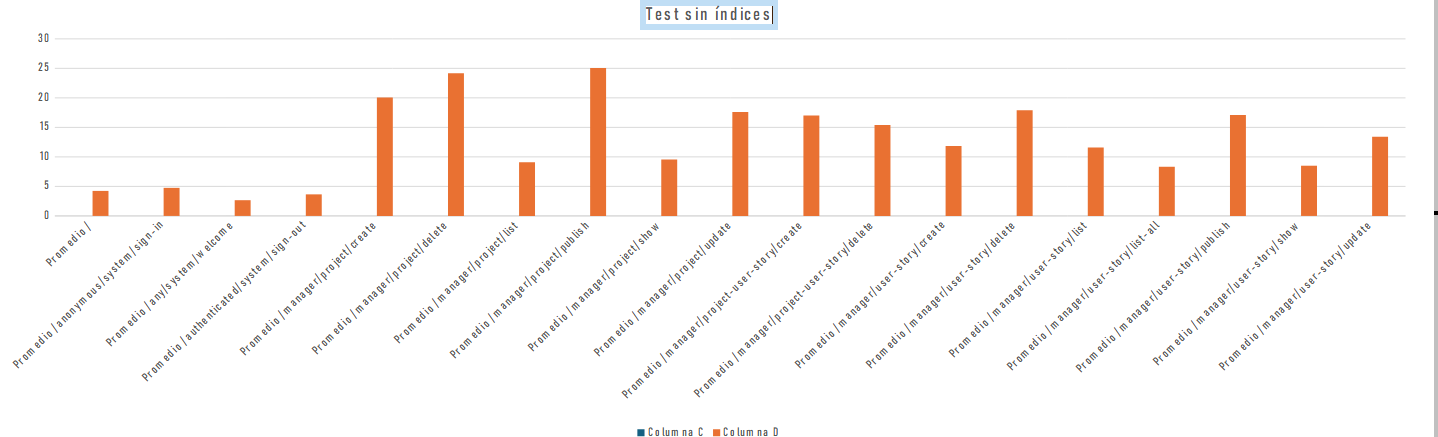
**Advertencia:**

Los resultados de esta prueba han sido afectados por un bug que ocurre en el framework, ya que hay veces que cuando se ejecuta el realplayer desde coverage, en consola se puede apreciar que algunos test tienen errores del tipo **“Error Got null response for request …”.** Esto no ocurre siempre, pues si se hace desde debug no pasa, incluso también cuando se hace desde coverage hay veces que no ocurre.

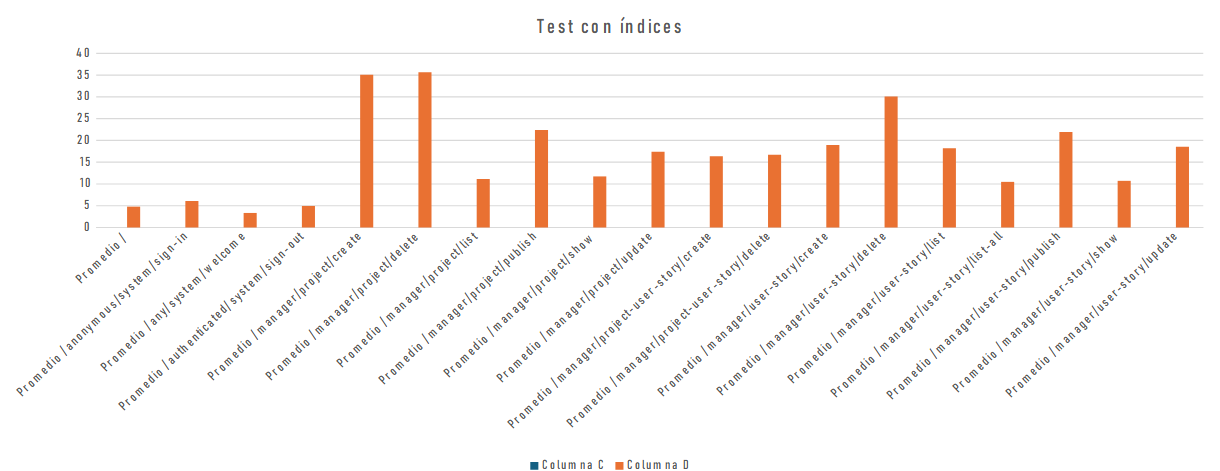
Es por todo esto mencionado anteriormente que he decidido de realizar otra prueba, pero esta vez ejecutando el replayer desde debug, puesto que en este no ocurre este tipo de errores.

**Muestra sobre resultados desde debug.**

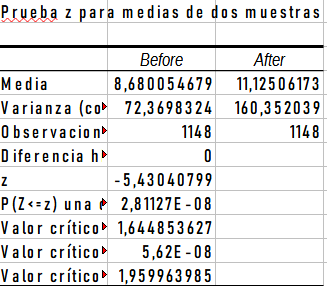
**Gráficas de entidades sin índices:**

****

**Gráficas de entidades con índices:**

****

**Comparación de resultados:**

****

En nuestras pruebas de rendimiento, hemos obtenido un valor crítico de z de **0.0018785**  para un nivel de significancia (α) de **0.95**. Este valor se utiliza para determinar si las diferencias en los tiempos de respuesta entre las dos pruebas son estadísticamente significativas.

De acuerdo con nuestra metodología, si el p-valor se encuentra en el intervalo (α, 1.00], esto indica que los cambios resultaron en mejoras relevantes; es decir, los tiempos de muestra son notablemente diferentes. Dado que nuestro valor crítico de z está en este intervalo, concluimos que hay una mejora significativa en el rendimiento al comparar los tiempos de respuesta de las pruebas. En resumen, las diferencias observadas son notables, debido a que los test con indices son más lentos que sin estos.

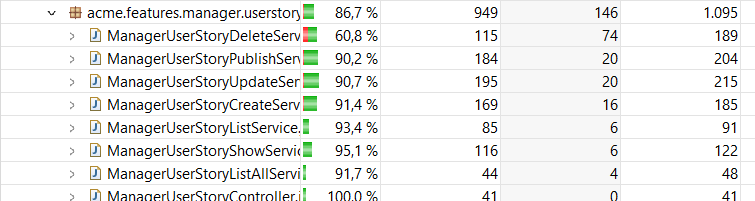
**Conclusión**

Una vez mostrado los resultados, podemos ver que hay una gran diferencia en las muestras cuando ocurren errores como el señalado anteriormente. Es por esto que he querido mostrar los resultados de ambas pruebas, ya que el resultado es el inverso y probablemente la primera prueba no sea válida por estos errores cometidos por el framework.

**Cobertura**

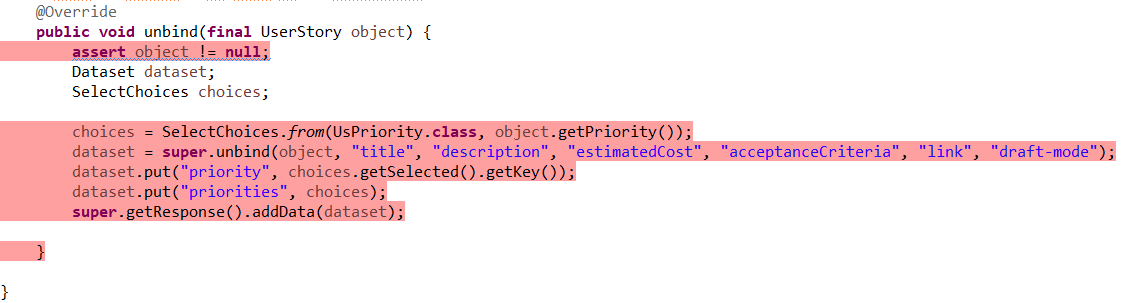
En esta sección, analizaremos la cobertura de código lograda con nuestros tests. La cobertura de código indica qué porcentaje del código fuente ha sido ejecutado durante las pruebas, ayudando a identificar áreas verificadas y posibles errores no detectados.

Discutiremos los métodos de medición, presentaremos los resultados obtenidos y evaluaremos la efectividad de nuestros tests. También identificaremos lagunas en la cobertura y propondremos estrategias para mejorarla, asegurando así mayor confiabilidad y robustez del software.

**Cobertura Historia de usuario.**

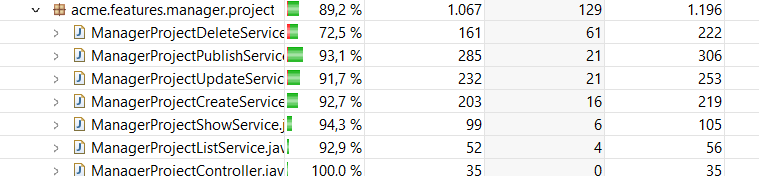
Como se puede apreciar, la cobertura sobre las features de la entidad Historia de usuario es bastante buena, ya que todas las features implementadas tienen una cobertura de más del 90% excepto el delete.

Esto ocurre porque el delete cuenta con el método unbind, pero este no es nunca usado porque no es necesario para esta funcionalidad, si comentamos este código o lo borramos la cobertura ya superaria el 90% de porcentaje, es por esto que no estamos preocupados por que este no llegue al 90% de cobertura.



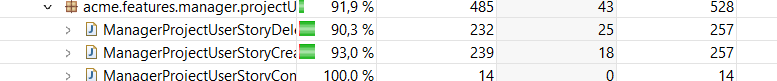
**Cobertura de proyecto.**

Con respecto a esta entidad, ocurre una cosa parecida a la comentada anteriormente, ya que la cobertura de todas las features es de mayor que el 90% exceptuando a delete que por el unbind no supera este porcentaje.



**Cobertura de proyecto-Historia de usuario.**

Con respecto a esta última, todas las features superan el 90% de cobertura, puesto que el delete sí usa el unbind y esto hace que todas estas líneas estén cubiertas.

****