Universidad de Sevilla  
Escuela Técnica Superior de Ingeniería Informática



**Grado en Ingeniería Informática – Ingeniería del Software**  
**Diseño y Pruebas II**

Curso 2023 – 2024

|  |  |
| --- | --- |
| **Fecha** | **Versión** |
| 27/05/24 | 1.0 |

|  |  |
| --- | --- |
| **Grupo de Prácticas: C1.012** | |
| **Repositorio:** [https://github.com/DP2-C1-012/Acme-SF](mailto:https://github.com/DP2-C1-012/Acme-SF) | |
| **Autores por orden alfabético** | **Correo** |
| Bernal Caunedo, Álvaro Jesús | [alvbercau@alum.us.es](mailto:alvbercau@alum.us.es) |
| Caballero Hernández, Jaime | [jaicabher1@alum.us.es](mailto:jaicabher1@alum.us.es) |
| Casamitjana Benítez, Juan José | [juacasben@alum.us.es](mailto:juacasben@alum.us.es) |
| Herrera Lobo, Nicolás | [nicherlob@alum.us.es](mailto:nicherlob@alum.us.es) |
| Montoya Albitres, Ronald Alexander | [ronmonalb@alum.us.es](mailto:ronmonalb@alum.us.es) |

**Índice de contenido**

[1. Resumen Ejecutivo 2](#_Toc474589782)

[2. Tabla de revisiones 2](#_Toc179714428)

[3. Introducción 2](#_Toc1184956288)

[4. Contenido 2](#_Toc647377246)

[5. Conclusiones 11](#_Toc201131074)

[6. Bibliografía 12](#_Toc1190488549)

# **Resumen Ejecutivo**

# Este informe ofrece un análisis detallado de las pruebas funcionales realizadas sobre las características de los proyectos y sus respectivas historias de usuario. El objetivo principal es asegurar la calidad y seguridad del sistema mediante la identificación y corrección de posibles errores antes de su implementación definitiva. Las pruebas incluyeron verificaciones exhaustivas de la creación, eliminación, listado, publicación, visualización y actualización de proyectos y sus historias de usuario, además de la posibilidad de relacionar estas historias o eliminar dichas relaciones, ya que los proyectos y las historias de usuario están interconectados. Adicionalmente, se llevaron a cabo pruebas de seguridad utilizando técnicas de hacking para garantizar que los datos no puedan ser manipulados por usuarios no autorizados. Los resultados muestran que se detectaron y corrigieron varias validaciones ausentes y que no se encontraron errores en las pruebas de seguridad.

# **Tabla de revisiones**

|  |  |  |
| --- | --- | --- |
| **Fecha** | **Revisión** | **Descripción** |
| 27/05/2024 | 1.0 | Versión inicial del documento |
|  |  |  |

# **Introducción**

Este documento presenta una estructura clara y ordenada, con un resumen ejecutivo al inicio seguido de una tabla de revisiones. La introducción establece el contexto, mientras que el contenido aborda los temas principales. Las conclusiones resumen los hallazgos, y la bibliografía respalda la investigación. Esta disposición facilita la comprensión y la navegación del documento.

# **Contenido**

* 1. **Pruebas Funcionales**
     1. **Project**
* **Create.safe:** Se comprueban que al crear un proyecto han de pasarse todas las validaciones relacionadas con sus respectivos campos para posteriormente proceder a la creación de varios proyectos probando los limites laterales de todos sus campos.
* **Delete.safe:** Se comprueba que un proyecto es borrado de forma correcta, así como su relación con las historias de usuario con las que previamente estaba relacionado.
* **List-mine.safe:** Se comprueba el listado de los proyectos de un mánager tras haber este iniciado sesión.
* **Publish.safe:** Se vuelven a comprobar que todos los campos de un proyecto están correctos previo a su publicación, probando también con límites laterales. También se comprueba que este proyecto ha de tener historias de usuario y que estas estén publicadas.
* **Show.safe:** Se comprueba la vista de detalles de un proyecto.
* **Update.safe:** Se comprueban al igual que en create y publish que no se puede actualizar un proyecto si este tiene algún error en sus campos o si está publicado, y de nuevo vuelven a probarse los límites laterales.
* **Create.hack:** Mediante GET hacking, se prueba a crear un proyecto sin ser un manager, devolviendo esto un error.
* **Delete.hack:** Mediante POST hacking, se prueba a eliminar un proyecto que no pertenece al manager que tiene iniciada sesión. De la misma forma, se prueba a eliminar un proyecto que está publicado. Ambas dieron error.
* **List-mine.hack:** Mediante GET hacking, se prueba a visualizar los proyectos de un manager sin ser un manager, dando esto un error.
* **Publish.hack:** Mediante POST hacking, se prueba a publicar un proyecto que no pertenece al manager que tiene iniciada sesión, a publicar un proyecto ya publicado y a publicar un proyecto sin historias de usuario asignadas ni publicadas. Todas estas devolvieron error.
* **Show.hack:** Mediante GET hacking, se prueba a acceder a la vista de detalle de un proyecto que pertenece a otro manager, devolviendo esto un error. También se probó sin tener la sesión iniciada, devolviendo también un error.
* **Update.hack:** Mediante POST hacking, se prueba a actualizar un proyecto que no pertenece al manager que tiene sesión iniciada y actualizar un proyecto ya publicado, devolviendo todo esto un error.
  + 1. **ProjectUs**

- **Create.safe**: Se comprueba que puede añadirse una historia de usuario a un proyecto que no esté publicado. Tampoco puede añadirse una historia de usuario a un proyecto al que ya pertenezca.

- **Delete.safe:** Se comprueba que una historia de usuario puede eliminarse sin problema de un proyecto que no esté publicado.

- **Create.hack**: Se comprueba mediante GET y POST hacking que no se puede añadir una historia de usuario a un proyecto que no pertenece al manager que tiene iniciada sesión y viceversa. Seguirá sin poder añadirse a un proyecto que ya esté publicado. Todo esto devuelve sus respectivos errores.

- **Delete.hack**: Se comprueba mediante GET y POST hacking que no se puede borrar una historia de usuario a un proyecto que no pertenece al manager que tiene iniciada sesión y viceversa. Seguirá sin poder borrarse de un proyecto que ya esté publicado. Todo esto devuelve sus respectivos errores.

* + 1. **User Story**
* **Create.safe:** Se comprueban que al crear una historia de usuario han de pasarse todas las validaciones relacionadas con sus respectivos campos para posteriormente proceder a la creación de varios proyectos probando los limites laterales de todos sus campos.
* **Delete.safe:** Se comprueba que una historia de usuario es borrada de forma correcta, así como su relación con los proyectos con los que previamente estaba relacionado.
* **List-all.safe:** Se comprueba el listado de todas las historias de usuario de un manager se realiza de forma correcta.
* **List-mine.safe:** Se comprueba el listado de las historias de usuario de un proyecto en concreto funciona correctamente.
* **Publish.safe:** Se vuelven a comprobar que todos los campos de una historia de usuario están correctos previo a su publicación, probando también con límites laterales.
* **Show.safe:** Se comprueba la vista de detalles de una historia de usuario.
* **Update.safe:** Se comprueban al igual que en create y publish que no se puede actualizar una historia de usuario si este tiene algún error en sus campos o si está publicada, y de nuevo vuelven a probarse los límites laterales.
* **Create.hack:** Mediante GET hacking, se prueba a crear una historia de usuario sin ser un manager, devolviendo esto un error.
* **Delete.hack:** Mediante POST hacking, se prueba a eliminar una historia de usuario que no pertenece al manager que tiene iniciada sesión. De la misma forma, se prueba a eliminar una historia de usuario que está publicado. Ambas dieron error.
* **List-all.hack:** Mediante GET hacking, se prueba a visualizar las historias de usuario de un manager sin ser un manager, dando esto un error.
* **List-mine.hack:** Mediante GET hacking, se prueba a visualizar las historias de usuario de un proyecto que no pertenecen al manager que tiene iniciada sesión.
* **Publish.hack:** Mediante POST hacking, se prueba a publicar una historia de usuario que no pertenece al manager que tiene iniciada sesión y a publicar una historia de usuario ya publicada. Todas estas devolvieron error.
* **Show.hack:** Mediante GET hacking, se prueba a acceder a la vista de detalle de una historia de usuario que pertenece a otro manager, devolviendo esto un error. También se probó sin tener la sesión iniciada, devolviendo también un error.
* **Update.hack:** Mediante POST hacking, se prueba a actualizar una historia d eusuario que no pertenece al manager que tiene sesión iniciada y actualizar una historia de usuario ya publicada, devolviendo todo esto un error.

Tras esto, vamos a ver unas imágenes en la que vamos a observar la cobertura de los tests mencionados anteriormente, Todos ellos tienen una gran cobertura superior al 90% en todas ellas, exceptuando el delete de Project y de User Story ya que en estos, al no utilizarse el método unbind, este no se testea al realizar la acción.

Tabla

Descripción generada automáticamente con confianza baja

Figura 1: Cobertura de project

Interfaz de usuario gráfica

Descripción generada automáticamente

Figura 2: Cobertura de ProyectUs

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Figura 3: Cobertura User Story

* 1. **Pruebas de Rendimiento**
* **Ordenador 1:**

Gráfico, Gráfico de barras

Descripción generada automáticamente

Figura 5: Tiempo promedio de las peticiones del pc 1

Cómo podemos ver en la Figura 2, el tiempo promedio de las peticiones es bastante bueno, a excepción de la acción de actualización de proyectos, que ha tardado severamente más.

Interfaz de usuario gráfica, Texto, Aplicación, Tabla

Descripción generada automáticamente

Figura 6: Tabla de rendimiento del PC1

En el cuadro 1, se muestran las estadísticas de rendimiento, así como el nivel de confianza.

* **Ordenador 2:**

Gráfico, Gráfico de barras

Descripción generada automáticamente

Figura 7: Tiempo promedio de las peticiones del pc2

En la Figura 7 aparece la gráfica de tiempos generados en el segundo ordenador una vez modificado los índices. En comparación a la Figura 5, ha habido una mejora en el tiempo medio de estas peticiones.

Tabla, Excel

Descripción generada automáticamente

Figura 8: rendimiento de pc2

Por último, en la figura 8 aparecen las estadísticas de rendimiento del pc2; cómo podemos ver, el resultado ha alojado grandes modificaciones en muchos de sus apartados. Por otra parte, la moda nos devuelve el valor N/D por motivos desconocidos.

* **Comparación:**

Tabla

Descripción generada automáticamente

Figura 9: Comparativa de rendimiento

En la figura 9 se muestran las estadísticas generadas para comparar ambas ejecuciones. Viendo el valor crítico de z para dos colas, se puede concluir en que los cambios han sido muy significativos ya que alfa era 0.5 y nos ha devuelto un valor de z de 0.006, muy alejado de este. Tras esto deducimos que la implementación de índices en nuestro proyecto ha sido un absoluto éxito.

# **Conclusi ones**

En conclusión, el proceso de pruebas es fundamental para asegurar la calidad y operatividad de cualquier sistema. Mediante el análisis de diversos componentes y escenarios de uso, es posible detectar y corregir errores potenciales antes de que estos impacten al usuario final. Este informe presenta un análisis exhaustivo de las pruebas funcionales realizadas sobre las funcionalidades de proyecto, historia de usuario y su entidad intermedia

Es crucial enfatizar que las pruebas son un proceso continuo que debe revisarse y ajustarse regularmente para garantizar el correcto funcionamiento del sistema. Se recomienda llevar a cabo evaluaciones periódicas para asegurarse de que el sistema siga operando según lo esperado y realizar los ajustes necesarios.

En resumen, este informe de pruebas proporciona una visión clara y detallada de las verificaciones realizadas, garantizando que las funcionalidades probadas funcionen de manera eficiente y segura. Este riguroso proceso de pruebas es vital para la calidad del sistema y la satisfacción del usuario final.

# **Bibliografía**

Intencionalmente en blanco.