Universidad de Sevilla  
Escuela Técnica Superior de Ingeniería Informática



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

Curso 2023 – 2024

**Informe de testing**

**Grupo:** C1.014

**Número de estudiante dentro del grupo:** 1

**Repositorio**: <https://github.com/alvaroChico2408/Acme-SF-D04>

**Fecha**: 27/05/2024

|  |  |
| --- | --- |
| **Alumno** | **Correo** |
| Linares Barrera, Jaime | jailinbar@alum.us.es |

**Historial de versiones**

|  |  |  |
| --- | --- | --- |
| **Versión** | **Fecha** | **Descripción de los cambios** |
| V1.0 | 25/05/2024 | Creación del informe |
| V1.1 | 26/05/2024 | Elaboración y revisión del informe |
|  |  |  |

**Índice de contenido**

[**1.** **Resumen ejecutivo** 2](#_Toc167735368)

[**2.** **Introducción** 2](#_Toc167735369)

[**3.** **Contenido** 3](#_Toc167735370)

[**3.1.** **Testing funcional** 3](#_Toc167735371)

[**3.1.1.** **Funcionalidad para los managers sobre los proyectos** 3](#_Toc167735372)

[**3.1.1.1.** **List-Mine** 3](#_Toc167735373)

[**3.1.1.2.** **Show** 4](#_Toc167735374)

[**3.1.1.3.** **Create** 4](#_Toc167735375)

[**3.1.1.4.** **Update** 5](#_Toc167735376)

[**3.1.1.5.** **Delete** 5](#_Toc167735377)

[**3.1.1.6.** **Publish** 6](#_Toc167735378)

[**3.1.2.** **Funcionalidad para los managers sobre las historias de usuario** 6](#_Toc167735379)

[**3.1.2.1.** **List-Mine** 6](#_Toc167735380)

[**3.1.2.2.** **List** 7](#_Toc167735381)

[**3.1.2.3.** **Show** 7](#_Toc167735382)

[**3.1.2.4.** **Create** 8](#_Toc167735383)

[**3.1.2.5.** **Update** 8](#_Toc167735384)

[**3.1.2.6.** **Delete** 9](#_Toc167735385)

[**3.1.2.7.** **Publish** 9](#_Toc167735386)

[**3.1.3.** **Funcionalidad para los managers sobre la relación entre proyectos e historias de usuario** 10](#_Toc167735387)

[**3.1.3.1.** **Create** 10](#_Toc167735388)

[**3.1.3.2.** **Delete** 11](#_Toc167735389)

[**3.2.** **Testing de rendimiento** 12](#_Toc167735390)

[**3.2.1.** **Sin índices – Ordenador estudiante 1** 12](#_Toc167735391)

[**3.2.2.** **Con índices – Ordenador estudiante 1** 13](#_Toc167735392)

[**3.2.3.** **Con índices – Ordenador estudiante 4** 15](#_Toc167735393)

[**4.** **Conclusiones** 17](#_Toc167735394)

[**Bibliografía** 18](#_Toc167735395)

# **Resumen ejecutivo**

Este informe presenta los resultados detallados de las pruebas funcionales y de rendimiento realizadas en el proyecto. En el apartado de pruebas funcionales, se describen los casos de prueba implementados. Cada caso de prueba incluye una descripción clara y una evaluación de su efectividad en la detección de errores. En el capítulo de pruebas de rendimiento, se presentan gráficos detallados y un intervalo de confianza del 95% para el tiempo de respuesta del sistema al procesar solicitudes en dos ordenadores diferentes. Además, se incluye un análisis comparativo con un nivel de confianza del 95% para determinar estadísticamente cuál de los dos ordenadores es más potente. Este informe utiliza un enfoque sistemático y riguroso, asegurando que los resultados sean precisos y útiles para la evaluación del proyecto.

# **Introducción**

El presente documento tiene como objetivo detallar los procedimientos y resultados de las pruebas funcionales y de rendimiento llevadas a cabo en el proyecto. En primer lugar, se aborda el apartado de pruebas funcionales, donde se describen los casos de prueba diseñados para verificar las diversas características del sistema. Cada uno de los casos de prueba incluye una descripción detallada y una evaluación de su efectividad en la detección de errores. Esta sección proporciona una visión clara y estructurada del desempeño del sistema bajo diferentes condiciones de prueba, permitiendo identificar y corregir fallos de manera efectiva.

En segundo lugar, el capítulo de pruebas de rendimiento se centra en el análisis del tiempo de respuesta del sistema al procesar solicitudes durante las pruebas funcionales. Se incluyen gráficos detallados que ilustran los tiempos de respuesta en dos ordenadores diferentes, junto con un intervalo de confianza del 95% para estos tiempos. Además, se lleva a cabo un contraste de hipótesis con un nivel de confianza del 95% para determinar cuál de los dos ordenadores es estadísticamente más potente. Este análisis no solo proporciona información sobre el rendimiento del sistema, sino que también ofrece una base sólida para la comparación de hardware.

El enfoque utilizado en este informe es metódico y sistemático, utilizando herramientas de planificación y Eclipse para garantizar la precisión y efectividad en la detección de errores durante las pruebas funcionales.

# **Contenido**

# **Testing funcional**

En esta sección vamos a explicar los casos de pruebas (positivos, negativos y de hacking) que hemos implementado para cada funcionalidad del sistema. Además vamos a ofrecer una explicación de como de efectivo es para detectar errores, ayudados por la cobertura que nos aporta la herramienta Eclipse la cual nos indica el porcentaje de instrucciones que ha sido ejecutadas.

# **Funcionalidad para los managers sobre los proyectos**

Vamos a comenzar por las pruebas establecidas para las funcionalidades de los manager sobre los proyectos.

# **List-Mine**

Esta funcionalidad consiste en que un manager pueda ver la lista de sus proyectos.

Para realizar las pruebas positivas y negativas de esta funcionalidad nos hemos logueado como un manager y hemos listado sus proyecto y después nos hemos logueado como otra manager y hemos listado todos sus proyectos.

Para realizar las pruebas de hacking, con el rol anónimo hemos introducido al url de mostrar nuestros proyecto (…/manager/Project/list-mine) comprobando que nos saltaba un error y los mismo pero ahora con otro rol (sponsor en mi caso).

Tras realizar estas pruebas hemos obtenido en el servicio un 93.1% de cobertura lo cual nos asegura que la funcionalidad no esconde ningún posible fallo en esas líneas ya que ha ejecutado todas las instrucciones posibles. Decimos que han sido todas las posibles ya que las líneas que salen en amarillo son líneas que no pueden llegar a ejecutarse de otra manera como es el caso de la siguiente línea que se da en todos los test y no volveremos a nombrar más.



A partir de ahora, le haremos referencia con un (1).

# **Show**

Esta funcionalidad consiste en que un manager pueda ver las características/atributos de uno de sus proyectos.

Para realizar las pruebas positivas y negativas de esta funcionalidad nos hemos logueado como manager, hemos mostrado el listado de sus proyectos y hemos ido pulsando en algún proyecto comprobando que aparecía la información correcta. Después nos hemos logueado con otro manager y hemos hecho lo mismo.

Para realizar las pruebas de hacking, hemos intentado acceder a la información de un proyecto con un rol distinto al del manager (anónimo y sponsor) y luego hemos intentado acceder como un manager pero con el usuario incorrecto, es decir, al proyecto de un manager distinto al que nos hemos logueado. En todos los casos han saltado los errores correspondientes.

Tras realizar estas pruebas hemos obtenido un 96.8% de cobertura asegurándonos de esta manera que no van a existir errores en la líneas ejecutadas ya que las únicas líneas que salen en amarillo es la ya comentada en apartados anteriores y la siguiente:

En este caso también sabemos que sale en amarillo debido a que hay un caso que nunca se da y es el caso en el que la primera condición sea true y la segunda false y esto se debe a que la primera condición no solo comprueba el rol si no que también comprueba que ese rol tenga la misma id que el que pasamos por parámetros, es decir, que la segunda condición es inservible, podría eliminarse sin ningún problema.

Este último caso también se da en las demás funcionalidades, por lo que únicamente lo nombraremos en los siguientes casos sin entrar en más detalles. Hacemos referencia a él con un (2).

# **Create**

Esta funcionalidad consiste en que un manager puede crear un proyecto estableciendo las características de su proyecto.

Para realizar las pruebas positivas y negativas, hemos empezado primero por las negativas, mandando el formulario en blanco y luego más tarde tras rellenar todos los campos con valores que pasan los test hemos ido probando todos los fallos distintos posibles que podían darse para cada atributo. Tras probar todos los casos negativos posibles, nos hemos limitado a crear proyecto probando con los rangos de valores posible como se ha explicado en la metodología de clase (probando los extremos de las restricciones, los diferentes valores que puede tomar…), siempre siendo valores permitidos.

Para realizar las pruebas de hacking, hemos probado a introducir la url para crear proyecto con un rol distinto al de manager (anónimo y sponsor), comprobando que no dejaba acceder.

Tras realizar estas pruebas hemos obtenido un 93.2% donde las únicas instrucciones que salen en amarillo son aquellas ya nombradas, las del caso (1). Por tanto, podemos decir que no pueden esconderse errores en el código de nuestro servicio.

# **Update**

Esta funcionalidad consiste en que un manager pueda actualizar uno de sus proyecto siempre y cuando este proyecto no haya sido publicado.

Para realizar las pruebas positivas y negativas, hemos llevado una metodología muy parecida a la llevada en el create solo que en el caso de las positivas en vez de ir creándolos hemos ido actualizando el proyecto.

Para realizar las pruebas de hacking, hemos intentado actualizar un proyecto con un rol diferente al de manager (sponsor y anónimo), después nos hemos logueado con un manager pero con un usuario distinto al del proyecto que hemos intentado actualizar (un proyecto publicado y otro no publicado). Por último, nos hemos logueado con un manager y con el usuario adecuado pero hemos intentado actualizar un proyecto que ya está publicado. En todos los casos han funcionado el sistema como esperábamos, rechazando tales peticiones.

Tras realizar esta pruebas hemos obtenido un 94.3% donde las únicas líneas en amarillo son aquellas ya nombradas la del caso (1) y la del caso (2). Por tanto, podemos decir que no pueden esconderse errores en nuestro código del servicio.

# **Delete**

Esta funcionalidad consiste en que un manager puede borrar sus proyecto siempre y cuando estos no hayan sido publicados.

Para realizar las pruebas positivas y negativas, hemos primero probado a borrar proyectos no publicados con historias de usuarios relacionadas y después a proyectos no publicados sin historias de usuario relacionadas. Es decir, todos los casos son positivos, ya que los casos negativos no están permitidos porque no existe autorización para borrar un proyecto que este publicado (no damos la opción del botón de borrar).

Para realizar las pruebas de hacking han seguido la misma línea que el Update, intentar borrar proyectos con el rol inadecuado, intentar borrarlo con el mismo rol pero usuario inadecuado (tanto publicados como no publicados) e intentar borrar un proyecto publicado con el rol y usuario adecuado. En todos los casos el acceso ha sido denegado.

En este caso, tras realizar las pruebas hemos obtenido un 90.8% donde las únicas líneas que han salido en amarillo son los dos casos ya nombrados ((1) y (2)). Por tanto, no pueden esconderse errores en este código.

# **Publish**

Esta funcionalidad consiste en que un manager puede publicar uno de sus proyectos siempre y cuando tenga al menos una historia de usuario relacionada, que todas sus historias de usuarios estén publicadas y que el proyecto no tenga errores fatales.

Para llevar a cabo las pruebas positivas y negativas, primero hemos intentado publicar proyectos en donde las historias de usuario no estaban publicadas o no tenia historias de usuario o el proyecto tenía errores fatales o cambiando ciertos atributos y probando casos en los que no eran validos (es decir, pruebas negativas). Tras comprobar que en todos estos casos nos han dado los errores correspondientes, hemos procedido a publicar proyectos que sí que podíamos publicar.

Para llevar realizar las pruebas de hacking, hemos intentado realizarlo con roles que no eran de manager, luego con el rol de manager pero con el usuario inadecuado (para proyectos publicados y no publicados) y, por último, con el rol y usuario correcto hemos intentado publicar proyectos que ya estaban publicados. En todos los casos se nos ha denegado el acceso, por tanto, la aplicación ha funcionado como esperábamos.

Tras realizar todas esta pruebas hemos obtenido un 95.3% de cobertura donde las únicas líneas que han quedado en amarillo son los casos ya nombrado ((1) y (2)). Por tanto, no pueden esconderse errores en nuestros código.

# **Funcionalidad para los managers sobre las historias de usuario**

Vamos a continuar por las pruebas establecidas para las funcionalidades de los managers sobre las historias de usuario.

# **List-Mine**

Esta funcionalidad consiste en que un manager pueda ver todas sus historias de usuario.

Para realizar los casos de prueba positivos y negativos, únicamente nos hemos logueado como un manager y hemos accedido al listado de sus historias de usuario y allí hemos comprobado que se listan todas sus historias de usuario. Luego hemos hecho lo mismo pero ahora con otro manager.

Para realizar los casos de prueba de hacking, únicamente con un rol inadecuado hemos intentado obtener un listado de sus historias de usuario sin éxito, respondiendo la aplicación como esperábamos.

Tras estos casos de pruebas hemos obtenido un 93.1% de cobertura quedando en amarillo las líneas relacionadas con el caso (1) mencionado en anteriores ocasiones.

# **List**

Esta funcionalidad consiste en que dado un proyecto se muestren las historias de usuario (recordar que tanto el proyecto como las historias de usuario han de tener el mismo manager asociado) que están relacionadas con ese proyecto.

Para realizar los casos de prueba positivos y negativos, nos hemos logueado como un manager, hemos accedido a su lista de proyectos, hemos entrado en los detalles (de uno publicado, uno sin publicar, uno con historias de usuario, otro sin historias de usuario…), hemos pulsado en el botón user stories y ahí se nos muestra el listado de las historias de usuario relacionadas con tal proyecto.

Para realizar los casos de prueba de hacking, los hemos llevado a cabo con un rol que no es el adecuado y luego con un rol adecuado pero con el usuario inadecuado (el proyecto no le pertenece a ese manager). En todos los casos no hemos podido acceder por lo que el sistema se comporta tal y como esperamos.

Tras realizar estas pruebas hemos obtenido un 94.7% de cobertura donde las líneas que salen en amarillo es el caso (1) y el siguiente nuevo caso:

Texto, Aplicación

Descripción generada automáticamente

En este caso sabemos que no va a poder ocurrir nunca que la segunda parte de la afirmación sea falsa (siempre será verdadera) ya que sino no tendríamos acceso a tal pantalla porque aparece tal restricción en el método de autorización. Por tanto, esa segunda condición podría eliminarse porque siempre sale true. Todos los demás caminos (que el proyecto sea haya publicado o no) si que se han cubierto.

# **Show**

Esta funcionalidad consiste en que un manager pueda ver las características/atributos de una de sus historias de usuario.

Para realizar las pruebas positivas y negativas de esta funcionalidad nos hemos logueado como manager, hemos mostrado el listado de sus historias de usuario y hemos ido pulsando en alguna historia de usuario comprobando que aparecía la información correcta. También hemos accedido a un proyecto, hemos accedido a sus características y hemos accedido al listado de sus historias de usuario relacionadas. Después nos hemos logueado con otro manager y hemos hecho lo mismo.

Para realizar las pruebas de hacking, hemos intentado acceder a la información de una historia de usuario con un rol distinto al del manager (anónimo y sponsor) y luego hemos intentado acceder como un manager pero con el usuario incorrecto, es decir, a la historia de usuario de un manager distinto al que nos hemos logueado. En todos los casos han saltado los errores correspondientes.

Tras realizar estas pruebas hemos obtenido un 97.1% de cobertura asegurándonos de esta manera que no van a existir errores en la líneas ejecutadas ya que las únicas líneas que salen en amarillo es la ya comentada en apartados anteriores ((1) y (2)).

# **Create**

Esta funcionalidad consiste en que un manager puede crear una historia de usuario estableciendo las características de su proyecto.

Para realizar las pruebas positivas y negativas, hemos empezado primero por las negativas, mandando el formulario en blanco y luego más tarde tras rellenar todos los campos con valores que pasan los test hemos ido probando todos los fallos distintos posibles que podían darse para cada atributo. Tras probar todos los casos negativos posibles, nos hemos limitado a crear proyecto probando con los rangos de valores posible como se ha explicado en la metodología de clase (probando los extremos de las restricciones, los diferentes valores que puede tomar…), siempre siendo valores permitidos.

Para realizar las pruebas de hacking, hemos probado a introducir la url para crear una historia de usuario con un rol distinto al de manager (anónimo y sponsor), comprobando que no dejaba acceder.

Tras realizar estas pruebas hemos obtenido un 93.8% donde las únicas instrucciones que salen en amarillo son aquellas ya nombradas, las del caso (1). Por tanto, podemos decir que no pueden esconderse errores en el código de nuestro servicio.

# **Update**

Esta funcionalidad consiste en que un manager pueda actualizar una de sus historia de usuario siempre y cuando esta historia de usuario no haya sido publicada.

Para realizar las pruebas positivas y negativas, hemos llevado una metodología muy parecida a la llevada en el create solo que en el caso de las positivas en vez de ir creándolos hemos ido actualizando la historia de usuario.

Para realizar las pruebas de hacking, hemos intentado actualizar un proyecto con un rol diferente al de manager (sponsor y anónimo), después nos hemos logueado con un manager pero con un usuario distinto al de la historia de usuario que hemos intentado actualizar (una historia de usuario publicada y otra no publicada). Por último, nos hemos logueado con un manager y con el usuario adecuado pero hemos intentado actualizar una historia de usuario que ya está publicada. En todos los casos han funcionado el sistema como esperábamos, rechazando tales peticiones.

Tras realizar esta pruebas hemos obtenido un 94.7% donde las únicas líneas en amarillo son aquellas ya nombradas la del caso (1) y la del caso (2). Por tanto, podemos decir que no pueden esconderse errores en nuestro código del servicio.

# **Delete**

Esta funcionalidad consiste en que un manager puede borrar una historia de usuario siempre y cuando estaos no hayan sido publicadas.

Para realizar las pruebas positivas y negativas, hemos primero probado a borrar historias de usuario no publicadas con proyectos o proyectos relacionados y después a historias de usuario no publicados sin proyectos relacionados. Es decir, todos los casos son positivos, ya que los casos negativos no están permitidos porque no existe autorización para borrar una historia de usuario que este publicada (no damos la opción del botón de borrar).

Para realizar las pruebas de hacking han seguido la misma línea que el Update, intentar borrar historias de usuario con el rol inadecuado, intentar borrarlo con el mismo rol pero usuario inadecuado (tanto publicados como no publicados) e intentar borrar una historia de usuario publicada con el rol y usuario adecuado. En todos los casos el acceso ha sido denegado.

En este caso, tras realizar las pruebas hemos obtenido un 90.8% donde las únicas líneas que han salido en amarillo son los dos casos ya nombrados ((1) y (2)). Por tanto, no pueden esconderse errores en este código.

# **Publish**

Esta funcionalidad consiste en que un manager puede publicar una historia de usuario.

Para realizar las pruebas positivas y negativas, hemos primero probado a publicar historias de usuarios modificando campos para que dieran errores (los del spam) y luego hemos pasado a publicar historias de usuario no publicadas con proyectos o proyectos relacionados y después a historias de usuario no publicados sin proyectos relacionados.

Para llevar realizar las pruebas de hacking, hemos intentado realizarlo con roles que no eran de manager, luego con el rol de manager pero con el usuario inadecuado (para historias de usuario publicadas y no publicadas) y, por último, con el rol y usuario correcto hemos intentado publicar historias de usuario que ya estaban publicadas. En todos los casos se nos ha denegado el acceso, por tanto, la aplicación ha funcionado como esperábamos.

Tras realizar todas esta pruebas hemos obtenido un 94.7% de cobertura donde las únicas líneas que han quedado en amarillo son los casos ya nombrado ((1) y (2)). Por tanto, no pueden esconderse errores en nuestros código.

# **Funcionalidad para los managers sobre la relación entre proyectos e historias de usuario**

Concluimos con las pruebas establecidas para las funcionalidades de los managers para añadir o eliminar historias de usuario de un proyecto.

# **Create**

Esta funcionalidad permite a los managers añadir historias de usuario a un proyecto. Para ello solo podrán añadirse historias de usuario del manager de ese proyecto (tiene que ser el mismo del de las historias de usuario) y que esas historias de usuario no hayan estado añadidas antes.

Para los casos positivos, hemos añadido historias de usuario (publicadas o no publicadas) a proyectos con historias de usuarios (publicadas o no publicadas) y a proyectos sin historias de usuario. Para los casos negativos hemos intentado añadir una historia de usuario nula dando el error que esperábamos.

Para las pruebas de hacking, hemos intentado añadir historias de usuarios a proyecto estando logueados con un rol inadecuado, con el rol adecuado pero un usuario inadecuado (proyecto publicado y no publicado) y con el rol y usuario adecuado pero estando el proyecto publicado. En todos los casos hemos tenido los errores que teníamos que obtener.

Tras estas pruebas la cobertura ha sido de un 87.3% esto se debe que además de las líneas amarillas que nos habían surgido anteriormente ((1) y (2)) han surgido otras como:

Texto

Descripción generada automáticamente con confianza media

En este caso los proyectos nunca van a tener errores ya que se obtiene directamente de base de datos por ello siempre va a entrar en todos los if. Además, el proyecto siempre va a estar no publicado ya que sino no autoriza a entrar a tal pantalla. También, nunca se cumple que exista una relación entre la historia de usuario que queremos unir con ese proyecto ya que solo mostramos aquellas historias de usuario que podemos relacionar con este proyecto (si ya está relacionada no la mostramos). Por la misma razón que esto anterior el proyecto y la historia de usuario siempre tienen el mismo manager.

Imagen que contiene Texto

Descripción generada automáticamente

En este caso, en el primer if nunca se cumple la condición de que la historia de usuario que mandamos no sea nula por eso nunca se entra en los bloques que aparecen en rojo que podrían eliminarse sin ningún problema.

# **Delete**

Esta funcionalidad permite a los managers eliminar historias de usuario de un proyecto. Para ello solo podrán eliminarse historias de usuario que estén añadidas.

Para los casos positivos, hemos eliminado historias de usuario (publicadas o no publicadas) de proyectos con historias de usuarios (publicadas o no publicadas) y de proyectos sin historias de usuario. Para los casos negativos hemos intentado eliminar una historia de usuario nula dando el error que esperábamos.

Para las pruebas de hacking, hemos intentado eliminar historias de usuarios de proyecto estando logueados con un rol inadecuado, con el rol adecuado pero un usuario inadecuado (proyecto publicado y no publicado) y con el rol y usuario adecuado pero estando el proyecto publicado. En todos los casos hemos tenido los errores que teníamos que obtener.

Tras estas pruebas la cobertura ha sido de un 86.8% esto se debe que además de las líneas amarillas que nos habían surgido anteriormente ((1), (2) y la segunda captura del caso del create) han surgido otras como:

Texto

Descripción generada automáticamente

Como podemos ver el validate queda muy parecido al anterior, lo único que cambia es que solo podemos borrar historias de usuarios que aparezcan y el problema para que no salga en verde es que siempre aparecen ya que solo mostramos historias de usuario que están relacionadas con tal proyecto.

# **Testing de rendimiento**

Nos centramos ahora en el análisis del tiempo de respuesta del sistema al procesar solicitudes durante las pruebas funcionales. Se llevarán acabo comparaciones entre nuestro portátil sin y con los índices para ver si estos han mejorado el rendimiento y además vamos a comparar los tiempos de nuestros test en el ordenador de otro estudiante.

# **Sin índices – Ordenador estudiante 1**

Gráfico, Gráfico en cascada

Descripción generada automáticamenteTras aplicar los procedimientos explicados en las diapositivas hemos obtenido los siguientes tiempos para las diversas funcionalidades. Además mostramos un gráfico donde podemos ver cuales son las funcionalidades que tardan más en ejecutarse.

Tabla

Descripción generada automáticamenteY hemos obtenido de los datos los siguiente resultados:

A continuación, en el siguiente apartado, veremos si los resultados con los índices ya incluidos han mejorado o no las prestaciones.

# **Con índices – Ordenador estudiante 1**

Estos son los tiempos obtenidos para las distintas funcionalidades además de un gráfico para resaltar cuales son las funcionalidades que tardan más en ejecutarse.

Gráfico, Gráfico en cascada

Descripción generada automáticamente

Como podemos observar el promedio general ha sido prácticamente igual (incluso un poco mayor) que sin utilizar los índices lo cual ya puede darnos un indicio de que no han mejorado las prestaciones.

Hemos obtenido los siguientes resultados:

Aplicación, Tabla, Excel

Descripción generada automáticamente

**Comparación entre utilizar y no utilizar índices**

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamenteNos disponemos a comparar los siguiente resultados:

Tabla

Descripción generada automáticamenteTras compararlos hemos obtenido lo siguiente resultados:

Como se observa en *Valor crítico de z (dos colas)***,** el valor está en el intervalo [0, Alpha), lo que indica que las diferencias en los tiempos medios entre las dos máquinas no son debidas al azar y por tanto podemos comparar. Observando ambas medias que el uso de índices no ha conseguido que mejore el rendimiento ya que la media obtenida tras utilizar los índices es mayor que la obtenida sin utilizar los índices.

# **Con índices – Ordenador estudiante 4**

Gráfico

Descripción generada automáticamenteEstos son los tiempos, obtenidos con el ordenador del estudiante 4 (que es de última generación), para las distintas funcionalidades además de un gráfico para resaltar cuales son las funcionalidades que tardan más en ejecutarse.

Como podemos observar el promedio general ha sido bastante menor que con nuestro portátil lo cual ya puede darnos un indicio de que las prestaciones del portátil del compañero son mejores que las nuestras.

Hemos obtenido los siguientes resultados:

Aplicación, Tabla

Descripción generada automáticamente

**Comparación entre los resultados de mi compañero contra los míos**

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamenteNos disponemos a comparar los siguientes resultados:

Tras compararlos hemos obtenido los siguientes resultados:

Tabla

Descripción generada automáticamente

Como se observa en *Valor crítico de z (dos colas)***,** el valor está en el intervalo [0, Alpha), lo que indica que las diferencias en los tiempos medios entre las dos máquinas no son debidas al azar y por tanto podemos comparar. Observando ambas medias podemos observar que el ordenador del compañero ha realizado la tarea más rápido que mi ordenador, lo cual podría ser algo normal teniendo en cuenta las prestaciones que ofrece su ordenador en comparación del mío.

# **Conclusiones**

Este informe presenta los resultados de las pruebas funcionales y de rendimiento realizadas en el proyecto. Las pruebas funcionales detallan diversos casos implementados para verificar las funcionalidades del sistema, evaluando su efectividad en la detección de errores. Las pruebas de rendimiento se enfocan en el análisis del tiempo de respuesta del sistema al procesar solicitudes en dos ordenadores diferentes, utilizando gráficos detallados e intervalos de confianza del 95% para asegurar la precisión de los resultados. Se realizó un análisis comparativo con el mismo nivel de confianza para determinar cuál de los dos ordenadores es más potente.

La sección de pruebas funcionales describe casos de prueba positivos, negativos y de hacking para cada funcionalidad del sistema, utilizando herramientas como Eclipse para asegurar una alta cobertura de código. Por ejemplo, en la funcionalidad "List-Mine" de proyectos para los managers, se probaron accesos correctos e incorrectos, logrando una cobertura del 93.1%. Casos similares se describen para funcionalidades como "Show", "Create", "Update", "Delete" y "Publish", todas con altos porcentajes de cobertura, asegurando que no queden errores ocultos en el código.

Las pruebas de rendimiento compararon el tiempo de respuesta del sistema en dos ordenadores diferentes, con y sin índices. Los resultados mostraron que el uso de índices no mejoró significativamente el rendimiento en el primer ordenador. Sin embargo, al comparar con un ordenador de última generación como es el de uno de los compañeros (estudiante 4), se observó una mejora notable en los tiempos de respuesta, confirmando que el hardware y software más avanzado ofrece mejores prestaciones.

En resumen, el informe confirma la efectividad del sistema en términos de funcionalidad y rendimiento. Las pruebas funcionales aseguran una alta cobertura y detección de errores, mientras que las pruebas de rendimiento destacan la influencia del hardware y software en el desempeño del sistema. Estos resultados proporcionan una base sólida para la evaluación y mejora continua del proyecto.

# **Bibliografía**

Intencionalmente en blanco.