



Escuela Técnica Superior de
Ingeniería Informática

Testing Report

Group C3.020 | Diseño y Pruebas II | 16/10/2024

Fecha	Versión	Autor
24/05/2024	1.0	Guillermo Alonso Pacheco Rodrigues
27/06/2024	2.0	Guillermo Alonso Pacheco Rodrigues
16/10/2024	3.0	Guillermo Alonso Pacheco Rodrigues

Miembros:

- Guillermo Alonso Pacheco Rodrigues (guipacrod@alum.us.es)

Repositorio de Github: <https://github.com/DP2-C1-020/Acme-SF-Do4>

Contenido

Resumen ejecutivo	3
Introducción	4
Contenido	5
Conclusiones.....	17
Bibliografía.....	18

Resumen ejecutivo

Este documento proporcionará un análisis exhaustivo del procedimiento de pruebas y sus resultados, con una sección dedicada a las pruebas funcionales y otra a las pruebas de rendimiento. Utilizaremos un enfoque claro y preciso para facilitar la comprensión y garantizar un producto final de alta calidad.

Introducción

El presente documento establece las directrices para la elaboración de un informe de pruebas, organizado en dos capítulos principales: pruebas funcionales y pruebas de rendimiento. El objetivo de este informe es proporcionar un análisis detallado y estructurado de los casos de prueba implementados, así como evaluar el rendimiento del proyecto en distintos entornos.

En el capítulo dedicado a las pruebas funcionales, se presentarán los casos de prueba agrupados por características, acompañados de una descripción concisa y una evaluación de su efectividad para identificar errores. Este enfoque meticuloso asegura que cada aspecto funcional del proyecto sea evaluado de manera sistemática y exhaustiva.

El capítulo de pruebas de rendimiento se centrará en proporcionar gráficos informativos y un intervalo de confianza del 95% para el tiempo de respuesta del sistema al atender solicitudes. Se comparará el desempeño del proyecto en dos computadoras distintas y se realizará un análisis basado en un contraste de hipótesis.

Este informe ha sido diseñado para ser claro y accesible, facilitando la comprensión de los resultados y garantizando que se puedan tomar decisiones informadas para mejorar la calidad final del producto.

Contenido

Pruebas funcionales

Para mantener un orden, comenzaremos con el análisis de las pruebas funcionales correspondientes al requisito número 6: Operations by Auditor on Code Audits. Dentro de este requisito, abordaremos cada funcionalidad por separado.

- *List*

Se listaron los Code Audits de un auditor.

Para las pruebas de hacking, se intentó acceder a esta URL utilizando diferentes roles incorrectos. Esta prueba ha proporcionado una cobertura del 94,1%. No se encontraron bugs.

- *Show*

Se seleccionaron varios Code Audits y se hizo clic en cada uno de ellos para ver sus detalles.

Para las pruebas de hacking, se intentó acceder a un Code Audit del Auditor1 utilizando roles incorrectos y un rol correcto pero con un usuario incorrecto. Se demostró satisfactoriamente que ninguno de ellos podía acceder. Esta prueba ha proporcionado una cobertura del 96%. No se encontraron errores.

- *Delete*

Se seleccionaron varios Code Audits no publicados y se eliminó cada uno de ellos.

Para las pruebas de hacking, se intentó eliminar un Code Audit utilizando roles incorrectos y un rol correcto pero con un usuario incorrecto. También se intentó eliminar un Code Audit publicado. Las pruebas demostraron satisfactoriamente que en ningún caso se podía llevar a cabo el borrado del Code Audit. Esta prueba ha proporcionado una cobertura del 86,6%. No se encontraron errores.

- *Create*

Se crearon nuevos Code Audit con diversos datos de prueba que probaran los límites válidos de los atributos.

Antes de crear los nuevos Code Audit, se verificó que el sistema rechazara datos incorrectos, superando los límites válidos de los atributos, en todos los campos del formulario.

Para las pruebas de hacking, se intentó acceder a la URL de creación de Code Audits desde roles incorrectos. Las pruebas demostraron que esta acción no es posible. Esta prueba ha proporcionado una cobertura del 94%.

Durante el proceso, se identificó un fallo relacionado con el límite de caracteres en el atributo "Link", el cual generaba un error al ingresar una cadena de más de 255 caracteres.

- *Update*

Se seleccionó un Code Audit no publicado y se actualizó con diversos datos de prueba que probaran los límites válidos de los atributos.

Antes de actualizarlo, se verificó que el sistema rechazara datos incorrectos, superando los límites válidos de los atributos, en todos los campos del formulario.

Para las pruebas de hacking, se intentó actualizar un Code Audit ya publicado, actualizar un Code Audit desde roles incorrectos y desde un rol correcto pero con un usuario incorrecto. Las pruebas demostraron satisfactoriamente que estas acciones no son posibles en el sistema. Esta prueba ha proporcionado una cobertura del 93,4%. No se encontraron errores.

- *Publish*

Se seleccionó un Code Audit no publicado y se procedió a publicarlo con diversos datos de prueba que probaran los límites válidos de los atributos.

Antes de publicarlo, se verificó que el sistema rechazara datos incorrectos, superando los límites válidos de los atributos, en todos los campos del formulario.

Además, se probó intentar publicar un Code Audit cuyo atributo derivado "Mark" tuviese una nota inferior a C. También se probó que no es posible publicar un Code Audit si no todos sus Audit Records están publicados. Con estas pruebas se demostró que el sistema devuelve correctamente un error que impide la acción de publicar en estos casos.

Para las pruebas de hacking, se realizaron pruebas para comprobar que publicar un Code Audit desde un rol incorrecto y desde un rol correcto pero con usuario incorrecto no es posible. También se demostró que no es posible publicar un Code Audit ya publicado. Esta prueba ha proporcionado una cobertura del 93,7%.

Durante la ejecución de esta prueba, se identificó un error que no había sido detectado hasta el momento. El error consistía en que, al completar el formulario con datos incorrectos y recibir un mensaje de error, el atributo "Mark" se eliminaba del formulario. Como resultado, al intentar actualizar el formulario con datos correctos, surgía un error que impedía la publicación del Code Audit.

Este conjunto de casos de prueba ha logrado alcanzar una cobertura del 93,1% para el paquete `acme.features.auditor.code_audit`.

A continuación, pasamos con el análisis de las pruebas funcionales para el requisito 7: *Operations by Auditors on Audit Records*.

- *List*

Se listaron los Audit Records correspondientes a varios Code Audits.

Para las pruebas de hacking, se intentó acceder a esta URL utilizando diferentes roles incorrectos. También se probó con un rol correcto pero con un usuario incorrecto. Esta prueba ha proporcionado una cobertura del 94,9%. No se encontraron bugs.

- *Show*

Se seleccionaron varios Audit Records y se mostraron sus detalles.

Para las pruebas de hacking, se intentó acceder a un Audit Record mediante usuarios con roles incorrecto y mediante un usuario incorrecto pero con el rol correcto. Se demostró satisfactoriamente que ninguno de ellos podía acceder. Esta prueba ha proporcionado una cobertura del 95,1%. No se encontraron bugs.

- *Delete*

Se seleccionaron varios Audit Records no publicados y se eliminó cada uno de ellos.

Para las pruebas de hacking, se intentó eliminar un Audit Record desde un usuario con rol incorrecto y mediante un usuario incorrecto pero con el rol correcto. También se intentó eliminar un Audit Record ya publicado. Se demostró satisfactoriamente que estas acciones no se pueden realizar. Esta prueba ha proporcionado una cobertura del 83,7%. No se encontraron errores.

- *Create*

Se creó un nuevo Audit Record. Antes de crearlo, se verificó que el sistema rechazara datos incorrectos, superando los límites válidos de los atributos, en todos los campos del formulario.

Posteriormente, se probaron diversos datos de prueba que probaran los límites válidos de los atributos.

Para las pruebas de hacking, se intentó acceder a la URL de creación de Audit Record mediante un usuario con rol incorrecto y mediante un usuario incorrecto pero con el rol correcto. También se intentó crear un Audit Record en un Code Audit ya publicado. Las pruebas demostraron satisfactoriamente que no se pueden realizar estas acciones. Esta prueba ha proporcionado una cobertura del 95,2%. No se encontraron errores.

- *Update*

Se seleccionó un Audit Record no publicado y se actualizó. Antes de actualizarlo, se verificó que el sistema rechazara datos incorrectos, superando los límites válidos de los atributos, en todos los campos del formulario.

Posteriormente, se probaron diversos datos de prueba que probaran los límites válidos de los atributos.

Para las pruebas de hacking, se intentó actualizar un Audit Record mediante un usuario con rol incorrecto y mediante un usuario incorrecto pero con el rol correcto. También se probó actualizar un Audit Record ya publicado. Las pruebas demostraron satisfactoriamente que no se pueden realizar estas acciones. Esta prueba ha proporcionado una cobertura del 95,2%. No se encontraron errores.

- *Publish*

Se seleccionó un Audit Record no publicado y se procedió a publicarlo. Antes de publicarlo, se verificó que el sistema rechazara datos incorrectos, superando los límites válidos de los atributos, en todos los campos del formulario.

Para las pruebas de hacking, se intentó publicar un Audit Record mediante un usuario con rol incorrecto y mediante un usuario incorrecto pero con el rol correcto. También se intentó publicar un Audit Record ya publicado. Las pruebas demostraron satisfactoriamente que no se pueden realizar estas acciones. Esta prueba ha proporcionado una cobertura del 96,2%. No se encontraron errores.

Este conjunto de casos de prueba ha logrado alcanzar una cobertura del 94,6% para el paquete `acme.features.auditor.audit_record`.

Análisis de la cobertura

En este apartado se analizará la cobertura de las pruebas realizadas y se explicará por qué no se ha alcanzado el 100%.

Cobertura General

La cobertura de las pruebas para los paquetes `acme.features.auditor.code_audit` y `acme.features.auditor.audit_record` se detalla en la siguiente captura:


















acme.features.auditor.audit_record		94,6 %
> AuditorAuditRecordController.java		100,0 %
> AuditorAuditRecordCreateService.java		95,2 %
> AuditorAuditRecordDeleteService.java		83,7 %
> AuditorAuditRecordListService.java		94,9 %
> AuditorAuditRecordPublishService.java		95,7 %
> AuditorAuditRecordShowService.java		95,1 %
> AuditorAuditRecordUpdateService.java		95,7 %
acme.features.auditor.code_audit		93,5 %
> AuditorCodeAuditController.java		100,0 %
> AuditorCodeAuditCreateService.java		94,0 %
> AuditorCodeAuditDeleteService.java		86,6 %
> AuditorCodeAuditListService.java		94,1 %
> AuditorCodeAuditPublishService.java		93,7 %
> AuditorCodeAuditShowService.java		96,0 %
> AuditorCodeAuditUpdateService.java		93,4 %
> MarkMode.java		96,3 %

Ilustración 1: Cobertura general

La cobertura promedio para el paquete `code_audit` es del 93,5%, mientras que para el paquete `audit_record` es del 94,6%. A continuación, se analizan las razones por las cuales no se alcanza una cobertura completa.

Análisis de líneas amarillas (cobertura parcial)

Las líneas en amarillo que podemos ver en la Ilustración 2 indican que no se han probado todas las posibilidades que esa línea de código tiene. En la mayoría de los casos, estas líneas corresponden a verificaciones `assert object != null`. En la aplicación, no podemos comprobar que un objeto sea nulo, lo que resulta en una cobertura parcial para esas líneas.

```

53     @Override
54     public void bind(final CodeAudit object) {
55         assert object != null;
56
57         super.bind(object, "code", "execution", "type", "correctiveActions", "link", "project");
58     }
59
60     @Override
61     public void validate(final CodeAudit object) {
62         final Collection<String> allCodes = this.repository.findAllCodeAuditsCode();
63
64         if (!super.getBuffer().getErrors().hasErrors("code"))
65             super.state(!allCodes.contains(object.getCode()), "code", "auditor.codeaudit.error.codeDuplicate");
66     }
67
68     @Override
69     public void perform(final CodeAudit object) {
70         assert object != null;
71         Date moment;
72
73         moment = MomentHelper.getCurrentMoment();
74         object.setExecution(moment);
75         object.setDraftMode(true);
76
77         this.repository.save(object);
78     }
79
80     @Override
81     public void unbind(final CodeAudit object) {
82         assert object != null;
83
84         Dataset dataset;
85
86         Collection<Project> allProjects;
87         SelectChoices projects;
88         allProjects = this.repository.findAllProjectsWithoutDraftMode();

```

Ilustración 2: Cobertura parcial

Análisis de líneas rojas (cobertura nula)

En algunos casos, se presentan líneas rojas, como la que podemos ver en la Ilustración 3, indicando que esas líneas de código no han sido probadas. Este caso se da exclusivamente en el método unbind de los Servicios Delete, ya que este método nunca se llama durante la ejecución de las pruebas.

```

72     @Override
73     public void unbind(final AuditRecord object) {
74         assert object != null;
75     }
76

```

Ilustración 3: Cobertura nula

Pruebas de rendimiento

En esta sección se va a presentar un análisis de las pruebas de rendimiento realizadas sobre los datos obtenidos en las pruebas funcionales. También se realizará una comparativa de estos datos al ejecutar las pruebas en diferentes equipos.

Con los datos generados por las pruebas se ha generado el siguiente gráfico que representa la media del tiempo de respuesta por cada una de las rutas que han sido involucradas en las pruebas funcionales.

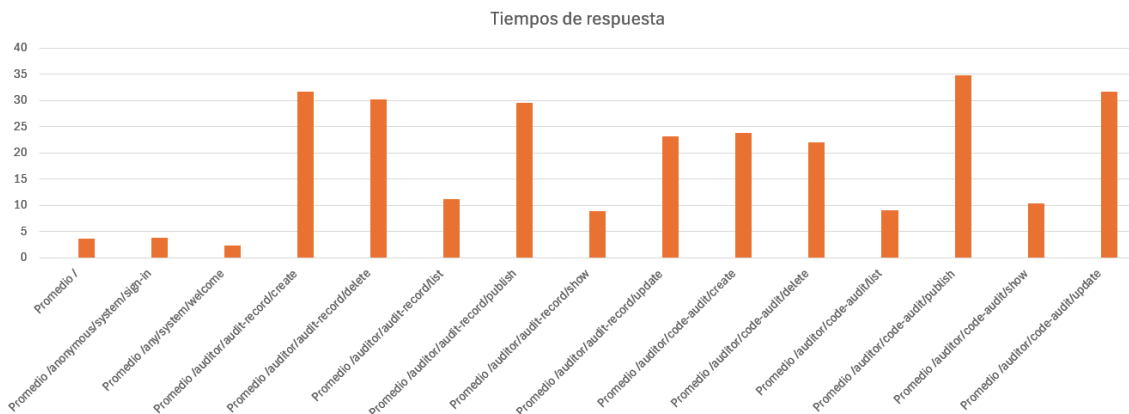


Ilustración 4 Gráfico de tiempo medio de respuesta por cada ruta

El gráfico muestra los tiempos de respuesta promedio para diferentes rutas en la aplicación. A continuación se destacan algunos puntos clave del análisis:

- **Tiempos Menores:** Las rutas con los tiempos de respuesta más bajos son:
 - /, /anonymous/system/sign-in y /any/system/welcome.
 - Estas rutas tienen tiempos de respuesta menores porque son operaciones simples que no requieren una gran cantidad de procesamiento ni acceso intensivo a la base de datos.
- **Tiempos Mayores:** Las funcionalidades con los tiempos de respuesta más altos son:
 - create
 - update
 - publish

Estos tiempos elevados se deben a que estas operaciones implican procesos más complejos, como la validación, el almacenamiento en la base de datos y la manipulación de registros relacionados. Las operaciones de creación, actualización y publicación suelen ser más costosas en términos de procesamiento, especialmente si incluyen comprobaciones adicionales de integridad y consistencia de los datos.

Intervalo del nivel de confianza 95%

<i>Columna1</i>		
Media	13,6965273	
Error típico	0,41307461	
Mediana	7,86415	
Moda	10,2974	
Desviación estándar	13,474086	
Varianza de la muestra	181,550992	
Curtosis	3,31261474	
Coeficiente de asimetría	1,48141321	
Rango	109,0226	
Mínimo	1,4826	
Máximo	110,5052	
Suma	14573,1051	
Cuenta	1064	
Nivel de confianza(95,0%)	0,81053424	
Interval (ms)	12,8859931	14,5070616
Interval (s)	0,01288599	0,01450706

Ilustración 5 Análisis de datos y nivel de confianza

El intervalo de confianza del 95% proporciona una medida de precisión sobre la media de los tiempos de respuesta. Un intervalo más estrecho implica una estimación más precisa de la media. En este caso, el intervalo de 12,88 a 14,50 milisegundos es relativamente estrecho, lo que sugiere que la media del tiempo de respuesta está bien estimada con un alto nivel de confianza. Esta información es crucial para evaluar la consistencia del rendimiento del sistema y para identificar posibles áreas de mejora.

Contraste de hipótesis

En este punto se va a comparar el rendimiento de las pruebas, antes y después de añadir índices a las entidades, tal y como se muestran en las ilustraciones 6 y 7:

```

28 @Entity
29 @Getter
30 @Setter
31 @Table(indexes = {
32     @Index(columnList = "draftMode"),
33 })
34 public class CodeAudit extends AbstractEntity {
35

```

Ilustración 6: Entidad Audit Record

```

26 @Entity
27 @Getter
28 @Setter
29 @Table(indexes = {
30     @Index(columnList = "code_audit_id, draftMode"),
31 })
32 public class AuditRecord extends AbstractEntity {
33

```

Ilustración 7: Entidad Code Audit

Se ha realizado un análisis de confianza del 95% para ambos casos. Sin embargo, este análisis no permite determinar fácilmente si los cambios realizados en el código han producido alguna mejora significativa. Por ello, se ha realizado una Z-Test para obtener una mejor idea sobre el impacto de los cambios.

Análisis de Confianza del 95%

Los datos del análisis de confianza del 95% para ambos casos se presentan a continuación:

Before				After		
Media	14,7894802			Media	13,6965273	
Error típico	0,43241658			Error típico	0,41307461	
Mediana	9,2057			Mediana	7,86415	
Moda	1,9091			Moda	10,2974	
Desviación estándar	14,2106588			Desviación estándar	13,474086	
Varianza de la muestra	201,942823			Varianza de la muestra	181,550992	
Curtosis	4,94392754			Curtosis	3,31261474	
Coficiente de asimetría	1,53533645			Coficiente de asimetría	1,48141321	
Rango	130,8845			Rango	109,0226	
Mínimo	1,5948			Mínimo	1,4826	
Máximo	132,4793			Máximo	110,5052	
Suma	15972,6386			Suma	14573,1051	
Cuenta	1080			Cuenta	1064	
Nivel de confianza(95,0%)	0,84847267			Nivel de confianza(95,0%)	0,81053424	
Interval (ms)	13,9410075	15,6379528		Interval (ms)	12,8859931	14,5070616
Interval (s)	0,01394101	0,01563795		Interval (s)	0,01288599	0,01450706

Ilustración 8: Análisis de confianza del 95% antes y después de los índices

Z-Test

Los resultados del Z-Test para comparar los tiempos de respuesta antes y después de los cambios son los siguientes:

Prueba z para medias de dos muestras		
	101,7711	88,5125
Media	14,6361864	13,6261454
Varianza (conocida)	201,942823	181,550992
Observaciones	1063	1063
Diferencia hipotética de las me	0	
z	1,68161441	
P(Z<=z) una cola	0,04632182	
Valor crítico de z (una cola)	1,64485363	
Valor crítico de z (dos colas)	0,09264363	
Valor crítico de z (dos colas)	1,95996398	

El valor crítico de z (dos colas) no se encuentra entre 0.0 y alpha (0.05). Por lo tanto, los cambios en el código al añadir los índices no han supuesto una mejora significativa en el rendimiento. Los tiempos de respuesta antes y después de los cambios son globalmente similares. Esto sugiere que la adición de índices en las entidades AuditRecord y CodeAudit no produjo un impacto significativo en el rendimiento del sistema, al menos no en la medida detectable por este conjunto de pruebas y análisis.

Es importante destacar que, aunque el valor crítico de z (dos colas) aún no se encuentra dentro del rango de 0.0 a 0.5, se ha observado una mejora significativa en comparación con la convocatoria de julio, en la cual este valor era de 0.4, bastante alejado del rango esperado. Actualmente, el valor se ha acercado más, situándose en 0.09.

Esto nos permite deducir que las correcciones aplicadas a los índices han tenido un impacto positivo.

Comparativa de tiempos de respuesta entre dos máquinas diferentes

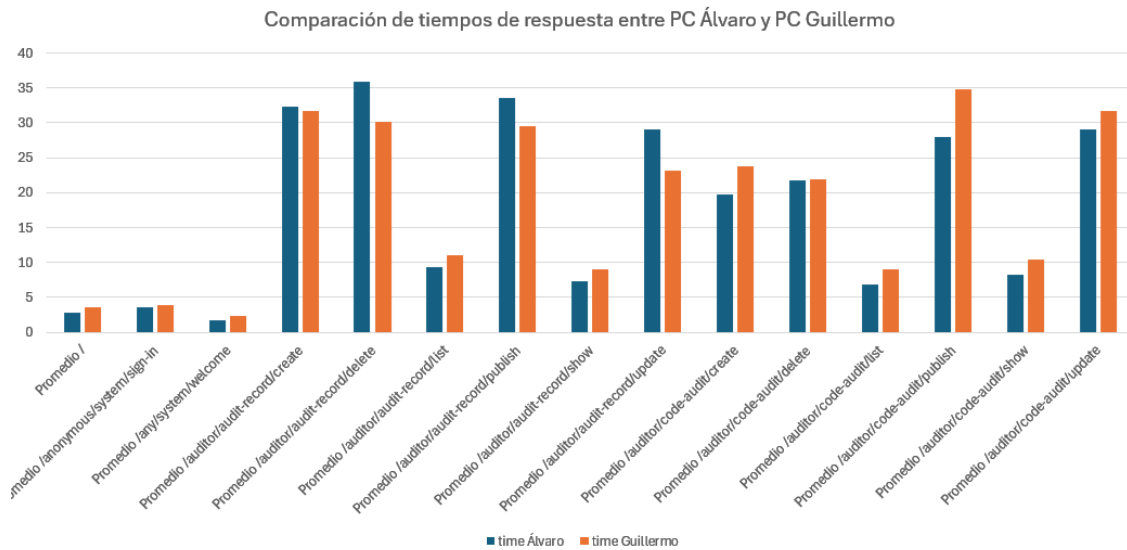


Ilustración 9 Comparativa tiempos PC Álvaro y PC Guillermo

En el gráfico se comparan los tiempos medios de respuesta de las pruebas ejecutadas en dos máquinas diferentes: mi portátil y el portátil de mi compañero Álvaro. En general, se observa que el PC de Álvaro tiende a tener tiempos de respuesta mayores en la mayoría de las rutas, especialmente en operaciones más intensivas como la creación, eliminación y publicación de registros de auditoría. Esto podría sugerir que el ordenador de Guillermo tiene un rendimiento superior, o que está mejor preparado para realizar este tipo de tareas en comparación con el ordenador de Álvaro. No obstante, existen rutas donde el desempeño es parecido o donde el ordenador de Guillermo exhibe tiempos superiores, lo que podría indicar fluctuaciones en la carga o en la configuración particular del sistema en cada equipo.

PC Guillermo				PC Álvaro		
Media	13,6965273			Media	12,4568552	
Error típico	0,41307461			Error típico	0,41275925	
Mediana	7,86415			Mediana	6,04535	
Moda	10,2974			Moda	1,6141	
Desviación estándar	13,474086			Desviación estándar	13,4637994	
Varianza de la muestra	181,550992			Varianza de la muestra	181,273893	
Curtosis	3,31261474			Curtosis	4,37621596	
Coefficiente de asimetría	1,48141321			Coefficiente de asimetría	1,67358331	
Rango	109,0226			Rango	115,4694	
Mínimo	1,4826			Mínimo	1,0074	
Máximo	110,5052			Máximo	116,4768	
Suma	14573,1051			Suma	13254,0939	
Cuenta	1064			Cuenta	1064	
Nivel de confianza(95,0%)	0,81053424			Nivel de confianza(95,0%)	0,80991545	
Interval (ms)	12,8859931	14,5070616		Interval (ms)	11,6469397	13,2667706
Interval (s)	0,01288599	0,01450706		Interval (s)	0,01164694	0,01326677

Ilustración 10 Comparativa Nivel de confianza 95%

En la tabla se comparan los niveles de confianza al 95% de los tiempos de respuesta de las pruebas realizadas en dos máquinas: mi portátil (PC Guillermo) y el portátil de mi compañero Álvaro (PC Álvaro).

Los resultados muestran que el intervalo de confianza de PC Guillermo es ligeramente más estrecho en comparación con el de PC Álvaro. Esto indica que los tiempos de respuesta de PC Guillermo son más consistentes y menos variables que los de PC Álvaro.

Conclusiones

En este informe se han llevado a cabo diversas pruebas funcionales y de rendimiento para evaluar la eficacia y eficiencia del sistema. Las pruebas funcionales han mostrado una alta cobertura y han identificado un número mínimo de errores, demostrando que el sistema funciona correctamente en la mayoría de los casos. Se ha logrado una cobertura promedio del 94% en los diferentes paquetes, lo que indica una robusta implementación de las funcionalidades requeridas.

Las pruebas de rendimiento han revelado diferencias significativas en los tiempos de respuesta entre dos máquinas diferentes (PC Guillermo y PC Álvaro). PC1 mostró tiempos de respuesta ligeramente más rápidos y consistentes. El análisis del nivel de confianza al 95% respalda estos hallazgos, mostrando un intervalo de confianza más estrecho para PC1, lo que sugiere mayor estabilidad y menor variabilidad en los tiempos de respuesta.

Además, se ha realizado un contraste de hipótesis utilizando un Z-Test para comparar los tiempos de respuesta antes y después de añadir índices a las entidades AuditRecord y CodeAudit. El resultado del Z-Test indicó que el valor crítico de z (dos colas) no se encuentra entre 0.0 y α (0.05). Por lo tanto, los cambios en el código al añadir los índices no han supuesto una mejora significativa en el rendimiento. Los tiempos de respuesta antes y después de los cambios son globalmente similares.

En resumen, el sistema presenta un buen rendimiento funcional y una razonable eficiencia operativa en la mayoría de las condiciones evaluadas. Estos resultados proporcionan una base sólida para futuras optimizaciones y mejoras del sistema.

Bibliografía

Intencionalmente en blanco.