

# Testing Report



**Grado en Ingeniería Informática - Ingeniería del Software**

**Diseño y Pruebas II**

**Curso 2023 - 2024**

Código de Grupo: C1-001		
Autores	Correo	Rol
Pedro Pablo Santos Domínguez	<a href="mailto:pedsandom@alum.us.es">pedsandom@alum.us.es</a>	Desarrollador

Repositorio: <<https://github.com/DP2-2023-2024-C1-001/Acme-SF-D04>>

# Índice de Contenidos

<b>1. Resumen ejecutivo.....</b>	<b>3</b>
<b>2. Control de Versiones.....</b>	<b>4</b>
<b>3. Introducción.....</b>	<b>5</b>
<b>4. Contenido.....</b>	<b>7</b>
4.1 Testing funcional.....	7
4.2 Testing de rendimiento.....	25
<b>5. Conclusión.....</b>	<b>29</b>
<b>6. Bibliografía.....</b>	<b>30</b>

# 1. Resumen ejecutivo

Para esta entrega del proyecto Acme-SF, nos hemos centrado en probar las funcionalidades para garantizar la calidad y fiabilidad de las soluciones ofrecidas. Hemos realizado pruebas end-to-end y pruebas de rendimiento para evaluar cómo se comportan los casos de prueba en la práctica. Cubrimos todos los aspectos del software para asegurar que cada componente funcione correctamente.

El Acme Framework facilita la automatización y el seguimiento de los resultados de las pruebas, lo que permite detectar fallos rápidamente y mejorar continuamente el sistema. Las herramientas integradas han garantizado la correcta funcionalidad a lo largo de todo el ciclo de vida del proyecto.

Las pruebas realizadas han demostrado ser efectivas para la detección temprana de errores y la validación de funcionalidades clave.

En resumen, el enfoque en el testing dentro del proyecto Acme-SF es fundamental para garantizar la efectividad de las soluciones, y el Acme Framework proporciona una base sólida para una gestión de pruebas eficiente y confiable.

## 2.Control de Versiones

Fecha	Versión	Descripción
27/05/2024	V1.0	Creación inicial del documento.

### 3. Introducción

El presente documento ofrece un análisis detallado de las pruebas realizadas en el proyecto Acme-SF, que utiliza el Acme Framework para la gestión de operaciones. El enfoque principal es el testing de las funcionalidades implementadas, con especial atención a las pruebas funcionales y de rendimiento.

#### **Pruebas Funcionales**

Este capítulo detalla las pruebas funcionales llevadas a cabo en el proyecto Acme-SF. Estas pruebas se centran en verificar que cada funcionalidad del software cumpla con los requisitos establecidos. Los casos de prueba están organizados por características del sistema, lo que facilita la comprensión y cobertura de las pruebas. Cada caso de prueba incluye una breve descripción de su propósito y procedimiento, así como una evaluación de su efectividad en la detección de errores. Al finalizar este capítulo, se proporciona una visión general de la cobertura de las pruebas funcionales y su contribución a la calidad del software.

#### **Pruebas de Rendimiento**

Esta sección presenta las pruebas de rendimiento realizadas en el proyecto Acme-SF. Estas pruebas se enfocan en evaluar el comportamiento de la aplicación bajo diferentes cargas de trabajo y ayudan a identificar posibles cuellos de botella que puedan afectar su rendimiento. Se incluyen gráficos que muestran el tiempo de respuesta del sistema (wall time) durante las pruebas funcionales en dos computadoras distintas. Además, se proporciona un intervalo de confianza del 95% para estos tiempos de respuesta, lo que ayuda a determinar la fiabilidad de los resultados obtenidos. Asimismo, se realiza un contraste de hipótesis para determinar si hay diferencias significativas en el rendimiento entre las dos computadoras utilizadas en las pruebas. Esta sección ofrece una visión clara del rendimiento del sistema y los factores que pueden influir en él.

#### **Importancia del Testing en el Proyecto Acme-SF**

El testing, tanto funcional como de rendimiento, es crucial para garantizar que el software cumpla con los requisitos del usuario y maneje las cargas de trabajo previstas eficientemente. El uso del Acme Framework ha facilitado la implementación y automatización de las pruebas, permitiendo una detección temprana de errores y mejoras continuas en el sistema.

#### **Estructura**

El documento se organiza en dos capítulos principales: uno dedicado al testing de funcionalidades, donde se detallan los casos de prueba implementados y su efectividad en la detección de errores, y otro centrado en el testing de rendimiento, que incluye gráficos y análisis estadísticos del tiempo de respuesta del sistema en diferentes condiciones. Cada capítulo proporciona una visión específica y detallada de las pruebas realizadas en el proyecto Acme-SF, destacando la importancia del testing para garantizar la calidad del software.

## 4. Contenido

### 4.1 Testing funcional

Para este apartado nos vamos a centrar en las funcionalidades para el rol **Auditor** ofrecidas para las entidades **CodeAudit** y **AuditRecord**, explicando los casos de pruebas que se han utilizado en cada caso. Las funcionalidades que se presentan son análogas para ambas clases y son listar (**List**), mostrar detalles (**Show**), crear (**Create**), actualizar (**Update**), eliminar (**Delete**) y publicar (**Update**). También se analizarán los bugs detectados mientras se realizaba cada caso de prueba.

Además de analizar cada caso de prueba, se va a mostrar la cobertura obtenida, realizando un análisis de las líneas de código que se han probado parcialmente y una explicación a dicho motivo.

#### Entidad CodeAudit

##### Casos de prueba positivos y negativos:

**Listar (List):** Este caso de prueba verifica que la funcionalidad de listar muestre correctamente todas las auditorías de código disponibles para el auditor. Se asegura de que la lista se genere de manera adecuada y que todos los elementos se muestren correctamente en la interfaz de usuario.

**Mostrar Detalles (Show):** Este caso de prueba verifica que la funcionalidad de mostrar detalles despliegue la información completa de una auditoría de código específica cuando se selecciona desde la lista. Se revisa que todos los detalles relevantes se muestren correctamente y que no haya información faltante o incorrecta.

**Crear (Create):** Esta prueba se enfoca en verificar que el auditor pueda crear una nueva auditoría de código de manera exitosa. Se comprueba que todos los campos requeridos se completen correctamente y que la información ingresada se guarde adecuadamente en la base de datos. Se han probado que se cumplan las siguientes restricciones sobre cada campo, mostrando una alerta correspondiente en cada caso:

- **Code:**
  - Debe asegurarse que no sea nulo, para lo que se ha probado enviando el campo vacío.
  - Debe ser único, para lo que se ha probado enviando un código que ya existe, concretamente se ha probado con A-000.
  - Debe seguir el patrón “[A-Z]{1,3}-[0-9]{3}”, para lo que se ha probado enviando otra cadena de texto distinta.
  - Caso positivo: “A-459”, cumple todas las restricciones.

- **Execution:**
  - Debe asegurarse que no sea nulo, para lo que se ha probado enviando el campo vacío.
  - Debe asegurarse que se usa una fecha en pasado, por lo que se ha usado la fecha “2022/07/30 00:01”, que en la simulación temporal del proyecto es una fecha futura.
  - Debe asegurarse que sea un tipo Date, para lo que se ha probado enviando un string en este campo.
  - Caso positivo: “2022/07/23 12:45”, cumple todas las restricciones.
- **Type (CodeAuditType):**
  - Debe asegurarse que no sea nulo, para lo que se ha probado enviando el campo vacío.
  - Caso positivo: “STATIC”, cumple todas las restricciones.
- **Corrective actions:**
  - Debe asegurarse que no sea nulo, para lo que se ha probado enviando el campo vacío.
  - Debe asegurarse que la cadena no sea superior a 100 caracteres, para lo que se ha probado enviando “Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempaosicoqwecunqwwJCNKAJdvcsakvb cdhsbcxsaDNJAD...” en el campo, la cual tiene un total de caracteres mayor a 100.
  - Caso positivo: “Lorem ipsum”, cumple todas las restricciones.
- **Link (opcional):**
  - Debe asegurarse que la cadena no sea inferior a 7 caracteres, ni superior a 255 caracteres, para lo que se ha probado enviando “ftp://”, para el inferior a 7 y “http://www.lorem-ipsum.org/dolor/sit/amet,/consectetur/adipiscing/elit,/sed/do /eiusmod/tempor/incididunt/ut/labore/et/do...” en el campo, la cual tiene un total de 101 caracteres.
  - Caso positivo: “http://example.org/a/b?a&b”, cumple todas las restricciones.
- **Proyecto:**
  - Debe asegurarse que no sea nulo, para lo que se ha probado enviando el campo vacío.
  - Caso positivo: “ABC-1234”, cumple todas las restricciones.

**Actualizar (Update):** En este caso de prueba se verifica que el cliente pueda actualizar la información de una auditoría de código existente. Se comprueba que los cambios realizados se reflejen correctamente en la interfaz de usuario y en la base de datos, sin perder información previa o introducir errores.

- **Code:**

- Debe asegurarse que no sea nulo, para lo que se ha probado enviando el campo vacío.
- Debe ser único, para lo que se ha probado enviando un código que ya existe, concretamente se ha probado con “AAA-000”.
- Debe seguir el patrón “[A-Z]{1,3}-[0-9]{3}”, para lo que se ha probado enviando otra cadena de texto distinta.
- Caso positivo: “AAA-987”, cumple todas las restricciones.

- **Execution:**

- Debe asegurarse que no sea nulo, para lo que se ha probado enviando el campo vacío.
- Debe asegurarse que se usa una fecha en pasado, por lo que se ha usado la fecha “2023/07/24 15:29”, que en la simulación temporal del proyecto es una fecha futura.
- Debe asegurarse que sea un tipo Date, para lo que se ha probado enviando un string en este campo.
- Caso positivo: “2021/07/24 07:39”, cumple todas las restricciones.

- **Type (CodeAuditType):**

- Debe asegurarse que no sea nulo, para lo que se ha probado enviando el campo vacío.
- Caso positivo: “STATIC”, cumple todas las restricciones.

- **Corrective actions:**

- Debe asegurarse que no sea nulo, para lo que se ha probado enviando el campo vacío.
- Debe asegurarse que la cadena no sea superior a 100 caracteres, para lo que se ha probado enviando “Capacitar al personal en el uso adecuado de herramientas dinámicas.Capacitar al personal en el uso adecuado de herramien...” en el campo, la cual tiene un total de caracteres mayor a 100.
- Caso positivo: “Capacitar al personal en el uso adecuado de herramientas dinámicas.”, cumple todas las restricciones.

- **Link (opcional):**

- Debe asegurarse que la cadena no sea inferior a 7 caracteres, ni superior a 255 caracteres, para lo que se ha probado enviando “ftp://”, para el inferior a 7 y “http://www.lorem-ipsum.org/dolor/sit/amet,/consectetur/adipiscing/elit,/sed/do/eiusmod/tempor/incididunt/ut/labore/et/do...” en el campo, la cual tiene un total de 101 caracteres.
- Caso positivo: “”, cumple todas las restricciones, ya que es opcional.



- **Proyecto:**

- Debe asegurarse que no sea nulo, para lo que se ha probado enviando el campo vacío.
- Caso positivo: "ABC-1234", cumple todas las restricciones.

**Eliminar (Delete):** Esta prueba verifica que el auditor pueda eliminar una auditoria de codigo de manera segura y efectiva. Se comprueba que al eliminar un elemento, este desaparezca de la lista y que no haya efectos secundarios no deseados en otras partes del sistema.

**Publicar (Publish):** Por último, se prueba la funcionalidad de publicar, que permite al cliente actualizar el estado de un contrato para indicar que ha sido publicado. Se verifica que esta acción se refleje correctamente en la interfaz de usuario y en la base de datos, y que el estado actualizado sea visible para todos los usuarios autorizados.

- **Code:**

- Debe asegurarse que no sea nulo, para lo que se ha probado enviando el campo vacío.
- Debe ser único, para lo que se ha probado enviando un código que ya existe, concretamente se ha probado con "AAA-000".
- Debe seguir el patrón "[A-Z]{1,3}-[0-9]{3}", para lo que se ha probado enviando otra cadena de texto distinta.
- Caso positivo: "AAA-987", cumple todas las restricciones.

- **Execution:**

- Debe asegurarse que no sea nulo, para lo que se ha probado enviando el campo vacío.
- Debe asegurarse que se usa una fecha en pasado, por lo que se ha usado la fecha "2023/07/24 15:29", que en la simulación temporal del proyecto es una fecha futura.
- Debe asegurarse que sea un tipo Date, para lo que se ha probado enviando un string en este campo.
- Caso positivo: "2021/07/24 07:39", cumple todas las restricciones.

- **Type (CodeAuditType):**

- Debe asegurarse que no sea nulo, para lo que se ha probado enviando el campo vacío.
- Caso positivo: "STATIC", cumple todas las restricciones.

- **Corrective actions:**
  - Debe asegurarse que no sea nulo, para lo que se ha probado enviando el campo vacío.
  - Debe asegurarse que la cadena no sea superior a 100 caracteres, para lo que se ha probado enviando “Capacitar al personal en el uso adecuado de herramientas dinámicas.Capacitar al personal en el uso adecuado de herramien...” en el campo, la cual tiene un total de caracteres mayor a 100.
  - Caso positivo: “Capacitar al personal en el uso adecuado de herramientas dinámicas.”, cumple todas las restricciones.
- **Link (opcional):**
  - Debe asegurarse que la cadena no sea inferior a 7 caracteres, ni superior a 255 caracteres, para lo que se ha probado enviando “ftp://”, para el inferior a 7 y “http://www.lorem-ipsuam.org/dolor/sit/amet,/consectetur/adipiscing/elit,/sed/do/eiusmod/tempor/incididunt/ut/labore/et/do...” en el campo, la cual tiene un total de 101 caracteres.
  - Caso positivo: “”, cumple todas las restricciones, ya que es opcional.
- **Proyecto:**
  - Debe asegurarse que no sea nulo, para lo que se ha probado enviando el campo vacío.
  - Caso positivo: “ABC-1234”, cumple todas las restricciones.
- **Mark:**
  - Además, tenemos este atributo derivado que obtiene la moda del atributo mark de la entidad, “**AuditRecord**” y para poder publicar la auditoría de código la moda debe ser “A+”, “A”, “B” o “C”.
  - Para, esto se ha probado dejándolo vacío y siendo la moda “F”, por lo que nos salta un mensaje de error.
  - También, se debe cumplir que todos los registros de auditoría asociados estén publicados.
  - Caso positivo: “B”, se cumplen todas las restricciones necesarias.

## **Casos de prueba de hacking:**

### **Listar (List):**

- Usuario sin registrar: Se intenta acceder a la lista de contratos utilizando la URL correspondiente sin iniciar sesión en el sistema. Se espera que el acceso sea denegado y se muestre un mensaje de error 500.
- Auditor al que no pertenece la lista: Se intenta acceder a la lista de auditorías de código utilizando la URL correspondiente con un auditor que no es el propietario de dicha lista de auditorías de código. Se espera que la lista de auditorías de código mostrada sea la del auditor con el que hemos iniciado sesión y no la del auditor de donde hemos copiado la url, ya que no hay ningún campo en la url que muestre información concreta del usuario.

**Crear (Create):**

- Usuario sin registrar: Se intenta crear una auditoría de código utilizando la URL correspondiente sin iniciar sesión en el sistema. Se espera que el acceso sea denegado y se muestre un mensaje de error 500.
- Auditor que no es dueño de la auditoría de código: Se intenta crear una auditoría de código utilizando la URL correspondiente con un auditor que no es el propietario de la auditoría de código a crear. Se espera que se muestre el formulario para crear una auditoría de código pero dicha auditoría de código se cree con el auditor con el que hemos iniciado sesión como propietario, ya que no hay ningún campo en la url que muestre información concreta del usuario.

**Mostrar detalles (Show):**

- Usuario sin registrar: Se intenta mostrar los detalles de la auditoría de código utilizando la URL correspondiente sin iniciar sesión en el sistema. Se espera que el acceso sea denegado y se muestre un mensaje de error 500.
- Auditor al que no pertenece la auditoría de código: Se intenta mostrar los detalles de la auditoría de código utilizando la URL correspondiente con un auditor que no es el propietario de dicha auditoría de código. Se espera que el acceso sea denegado y se muestre un mensaje de error 500.

**Actualizar (Update):**

- Usuario sin registrar: Se intenta actualizar una auditoría de código utilizando la URL correspondiente sin iniciar sesión en el sistema. Se espera que el acceso sea denegado y se muestre un mensaje de error 500.
- Auditor al que no pertenece la auditoría de código: Se intenta actualizar una auditoría de código utilizando la URL correspondiente con un auditor que no es el propietario de dicha auditoría de código. Se espera que el acceso sea denegado y se muestre un mensaje de error 500.

**Publicar (Publish):**

- Usuario sin registrar: Se intenta publicar una auditoría de código utilizando la URL correspondiente sin iniciar sesión en el sistema. Se espera que el acceso sea denegado y se muestre un mensaje de error 500.
- Auditor al que no pertenece la auditoría de código: Se intenta publicar una auditoría de código utilizando la URL correspondiente con un auditor que no es el propietario de dicha auditoría de código. Se espera que el acceso sea denegado y se muestre un mensaje de error 500.

**Eliminar (Delete):**

- Usuario sin registrar: Se intenta eliminar una auditoría de código utilizando la URL correspondiente sin iniciar sesión en el sistema. Se espera que el acceso sea denegado y se muestre un mensaje de error 500.
- Auditor al que no pertenece la auditoría de código: Se intenta eliminar una auditoría de código utilizando la URL correspondiente con un auditor que no es el propietario de dicho contrato. Se espera que el acceso sea denegado y se muestre un mensaje de error 500.

## Bugs detectados

No hemos encontrado ningún bug, a la hora de probar todos los test de la aplicación.

## Entidad ProgressLog

### Casos de prueba positivos y negativos:

**Listar (List):** Este caso de prueba verifica que la funcionalidad de listar dentro de una auditoría de código muestre correctamente todos los registros de auditoría asociados a esa auditoría de código disponibles para el auditor. Se comprueba que la lista se genere de manera adecuada y que todos los elementos sean mostrados correctamente en la interfaz de usuario.

**Mostrar Detalles (Show):** En este caso de prueba se verifica que la funcionalidad de mostrar detalles despliegue la información completa de un registro de auditoría específico cuando se selecciona desde la lista de registros de auditoría asociados a una auditoría de código. Se revisa que todos los detalles relevantes sean mostrados correctamente y que no haya información faltante o incorrecta.

**Crear (Create):** Esta prueba se enfoca en verificar que el auditor pueda crear un nuevo registro de auditoría de manera exitosa. Se comprueba que todos los campos requeridos se completen correctamente y que la información ingresada se guarde adecuadamente en la base de datos. Se han probado que se cumplan las siguientes restricciones sobre cada campo, mostrando una alerta correspondiente en cada caso:

- **Code:**
  - Debe asegurarse que no sea nulo, para lo que se ha probado enviando el campo vacío.
  - Debe ser único, para lo que se ha probado enviando un código que ya existe, concretamente se ha probado con "AU-0099-000".
  - Debe seguir el patrón AU-[0-9]{4}-[0-9]{3}, para lo que se ha probado enviando "AU-000" en este campo.
  - Caso positivo: "AU-0099-039", cumple todas las restricciones.
- **PeriodStart:**
  - Debe asegurarse que no sea nulo, para lo que se ha probado enviando el campo vacío.
  - Debe asegurarse que se usa una fecha posterior a execution de la auditoría de código, por lo que se ha usado la fecha "2009/09/26 15:34", que en la simulación temporal del proyecto es una fecha futura.
  - Debe asegurarse que sea un tipo Date, para lo que se ha probado enviando un string en este campo.
  - Caso positivo: "2022/09/26 15:34", cumple todas las restricciones.
- **PeriodEnd:**
  - Debe asegurarse que no sea nulo, para lo que se ha probado enviando el campo vacío.

- Debe asegurarse que se usa una fecha posterior a “periodStart”, al menos una hora, por lo que se ha usado la fecha “2021/09/26 19:34”, que en la simulación temporal del proyecto es una fecha futura.
- Debe asegurarse que sea un tipo Date, para lo que se ha probado enviando un string en este campo.
- Caso positivo: “2022/09/26 19:34”, cumple todas las restricciones.
- **Mark:**
  - Debe asegurarse que no sea nulo, para lo que se ha probado enviando el campo vacío.
  - También tiene una restricción de valores, que solo acepta “A+”, “A”, “B”, “C”, “F” o “F-”, por lo que hemos probado con “BVAEV”
  - Caso positivo: “B”, se cumplen todas las restricciones necesarias.
- **Link (opcional):**
  - Debe asegurarse que la cadena no sea inferior a 7 caracteres, ni superior a 255 caracteres, para lo que se ha probado enviando “ftp://”, para el inferior a 7 y  
“http://www.lorem-ipsuam.org/dolor/sit/amet,/consectetur/adipiscing/elit,/sed/do/eiusmod/tempor/incididunt/ut/labore/et/do...” en el campo, la cual tiene un total de 101 caracteres.
  - Caso positivo: “”, cumple todas las restricciones, ya que es opcional.

**Actualizar (Update):** En este caso de prueba se verifica que el auditor pueda actualizar la información de un registro de auditoría existente perteneciente a una de sus auditorías de código. Se comprueba que los cambios realizados se reflejen correctamente en la interfaz de usuario y en la base de datos, sin perder información previa o introducir errores.

- **Code:**
  - Debe asegurarse que no sea nulo, para lo que se ha probado enviando el campo vacío.
  - Debe ser único, para lo que se ha probado enviando un código que ya existe, concretamente se ha probado con “AU-0099-000”.
  - Debe seguir el patrón AU-[0-9]{4}-[0-9]{3}, para lo que se ha probado enviando “AU-000” en este campo.
  - Caso positivo: “AU-0099-001”, cumple todas las restricciones.
- **PeriodStart:**
  - Debe asegurarse que no sea nulo, para lo que se ha probado enviando el campo vacío.
  - Debe asegurarse que se usa una fecha posterior a execution de la auditoría de código, por lo que se ha usado la fecha “2009/09/26 15:34”, que en la simulación temporal del proyecto es una fecha futura.
  - Debe asegurarse que sea un tipo Date, para lo que se ha probado enviando un string en este campo.
  - Caso positivo: “2022/09/26 15:34”, cumple todas las restricciones.
- **PeriodEnd:**

- Debe asegurarse que no sea nulo, para lo que se ha probado enviando el campo vacío.
  - Debe asegurarse que se usa una fecha posterior a “periodStart”, al menos una hora, por lo que se ha usado la fecha “2021/09/26 19:34”, que en la simulación temporal del proyecto es una fecha futura.
  - Debe asegurarse que sea un tipo Date, para lo que se ha probado enviando un string en este campo.
  - Caso positivo: “2022/09/26 19:34”, cumple todas las restricciones.
- **Mark:**
    - Debe asegurarse que no sea nulo, para lo que se ha probado enviando el campo vacío.
    - También tiene una restricción de valores, que solo acepta “A+”, “A”, “B”, “C”, “F” o “F-”, por lo que hemos probado con “BVAEV”
    - Caso positivo: “B”, se cumplen todas las restricciones necesarias.
- **Link (opcional):**
    - Debe asegurarse que la cadena no sea inferior a 7 caracteres, ni superior a 255 caracteres, para lo que se ha probado enviando “ftp://”, para el inferior a 7 y  
“http://www.lorem-ipsuam.org/dolor/sit/amet,/consectetur/adipiscing/elit,/sed/do/eiusmod/tempor/incidunt/ut/labore/et/do...” en el campo, la cual tiene un total de 101 caracteres.
    - Caso positivo: “”, cumple todas las restricciones, ya que es opcional.

**Eliminar (Delete):** Esta prueba verifica que el auditor pueda eliminar un registro de auditoría asociado a una de sus auditorías de código de manera segura y efectiva. Se comprueba que al eliminar un elemento, este desaparezca de la lista y que no haya efectos secundarios no deseados en otras partes del sistema.

**Publicar (Publish):** Por último, se prueba la funcionalidad de publicar, que permite al auditor actualizar el estado de un registro de auditoría para indicar que ha sido publicado. Se verifica que esta acción se refleje correctamente en la interfaz de usuario y en la base de datos, y que el estado actualizado sea visible para todos los usuarios autorizados.

- **Code:**
  - Debe asegurarse que no sea nulo, para lo que se ha probado enviando el campo vacío.
  - Debe ser único, para lo que se ha probado enviando un código que ya existe, concretamente se ha probado con “AU-0099-000”.
  - Debe seguir el patrón AU-[0-9]{4}-[0-9]{3}, para lo que se ha probado enviando “AU-000” en este campo.
  - Caso positivo: “AU-4985-095”, cumple todas las restricciones.

- **PeriodStart:**
  - Debe asegurarse que no sea nulo, para lo que se ha probado enviando el campo vacío.
  - Debe asegurarse que se usa una fecha posterior a execution de la auditoría de código, por lo que se ha usado la fecha “2009/09/26 15:34”, que en la simulación temporal del proyecto es una fecha futura.
  - Debe asegurarse que sea un tipo Date, para lo que se ha probado enviando un string en este campo.
  - Caso positivo: “2022/09/26 15:34”, cumple todas las restricciones.
- **PeriodEnd:**
  - Debe asegurarse que no sea nulo, para lo que se ha probado enviando el campo vacío.
  - Debe asegurarse que se usa una fecha posterior a “periodStart”, al menos una hora, por lo que se ha usado la fecha “2021/09/26 19:34”, que en la simulación temporal del proyecto es una fecha futura.
  - Debe asegurarse que sea un tipo Date, para lo que se ha probado enviando un string en este campo.
  - Caso positivo: “2022/09/26 19:34”, cumple todas las restricciones.
- **Mark:**
  - Debe asegurarse que no sea nulo, para lo que se ha probado enviando el campo vacío.
  - También tiene una restricción de valores, que solo acepta “A+”, “A”, “B”, “C”, “F” o “F-”, por lo que hemos probado con “BVAEV”
  - Caso positivo: “B”, se cumplen todas las restricciones necesarias.
- **Link (opcional):**
  - Debe asegurarse que la cadena no sea inferior a 7 caracteres, ni superior a 255 caracteres, para lo que se ha probado enviando “ftp://”, para el inferior a 7 y “http://www.lorem-ipsuam.org/dolor/sit/amet,/consectetur/adipiscing/elit,/sed/do/eiusmod/tempor/incididunt/ut/labore/et/do...” en el campo, la cual tiene un total de 101 caracteres.
  - Caso positivo: “”, cumple todas las restricciones, ya que es opcional.

## **Casos de prueba de hacking:**

### **Listar (List):**

- Usuario sin registrar: Se intenta acceder a la lista de registros de auditoría asociada a una auditoría de código utilizando la URL correspondiente sin iniciar sesión en el sistema. Se espera que el acceso sea denegado y se muestre un mensaje de error 500.
- Auditor al que no pertenece la lista: Se intenta acceder a la lista de registros de auditoría asociada a una auditoría de código utilizando la URL correspondiente con un auditor que no es el propietario de dicho auditor. Se espera que el acceso sea denegado y se muestre un mensaje de error 500, ya que el campo masterId en la url

contiene el id de la auditoría de código de la que se quieren listar sus registros de auditoría.

**Crear (Create):**

- Usuario sin registrar: Se intenta crear un registro de auditoría asociado a una auditoría de código utilizando la URL correspondiente sin iniciar sesión en el sistema. Se espera que el acceso sea denegado y se muestre un mensaje de error 500.
- Auditor que no es dueño de la auditoría de código: Se intenta crear un registro de auditoría asociado a una auditoría de código utilizando la URL correspondiente con un auditor que no es el propietario de dicha auditoría de código. Se espera que el acceso sea denegado y se muestre un mensaje de error 500, ya que el campo masterId en la url contiene el id de la auditoría de código de la que se quieren listar sus registros de auditoría.

**Mostrar detalles (Show):**

- Usuario sin registrar: Se intenta mostrar los detalles de un registro de auditoría utilizando la URL correspondiente sin iniciar sesión en el sistema. Se espera que el acceso sea denegado y se muestre un mensaje de error 500.
- Auditor que no es dueño del registro de auditoría: Se intenta mostrar los detalles de un registro de auditoría utilizando la URL correspondiente con un auditor que no es el propietario de dicha auditoría de código. Se espera que el acceso sea denegado y se muestre un mensaje de error 500.

**Actualizar (Update):**

- Usuario sin registrar: Se intenta actualizar un registro de auditoría utilizando la URL correspondiente sin iniciar sesión en el sistema. Se espera que el acceso sea denegado y se muestre un mensaje de error 500.
- Auditor que no es dueño del registro de auditoría: Se intenta actualizar un registro de auditoría utilizando la URL correspondiente con un auditor que no es el propietario de dicha auditoría de código. Se espera que el acceso sea denegado y se muestre un mensaje de error 500.

**Publicar (Publish):**

- Usuario sin registrar: Se intenta publicar un registro de auditoría utilizando la URL correspondiente sin iniciar sesión en el sistema. Se espera que el acceso sea denegado y se muestre un mensaje de error 500.
- Auditor que no es dueño del registro de auditoría: Se intenta publicar un registro de auditoría utilizando la URL correspondiente con un auditor que no es el propietario de dicha auditoría de código. Se espera que el acceso sea denegado y se muestre un mensaje de error 500.

**Eliminar (Delete):**

- Usuario sin registrar: Se intenta eliminar un registro de auditoría utilizando la URL correspondiente sin iniciar sesión en el sistema. Se espera que el acceso sea denegado y se muestre un mensaje de error 500.



- Auditor que no es dueño del registro de auditoría: Se intenta eliminar un registro de auditoría utilizando la URL correspondiente con un auditor que no es el propietario de dicha auditoría de código. Se espera que el acceso sea denegado y se muestre un mensaje de error 500.




## **Bugs detectados**

Para esta entidad no fueron encontrados bugs que comprometieron la estabilidad del sistema durante el testing formal, ya que todos los bugs existentes durante el desarrollo de la misma fueron encontrados y solucionados tras realizarse el testing informal y en revisiones de seguimiento del proyecto con nuestro tutor.

Los bugs encontrados durante el testing informal fueron:

- Cuando tratábamos de realizar una operación de borrado (delete), el problema ocurre al trabajar con delete en la url, ya que el botón funcionaba sin problema. Pero con el testing encontramos que los atributos del deleteService del método bind y unbind, no eran los suyos por lo que daba error 500.

## Cobertura de tests

▼	acme.features.auditor.codeaudits		93,7 %	1.255	84	1.339
>	AuditorCodeAuditUpdateService.java		93,1 %	269	20	289
>	AuditorCodeAuditShowService.java		96,0 %	144	6	150
>	AuditorCodeAuditPublishService.java		94,3 %	329	20	349
>	AuditorCodeAuditListService.java		93,1 %	54	4	58
>	AuditorCodeAuditDeleteService.java		91,9 %	204	18	222
>	AuditorCodeAuditCreateService.java		93,2 %	220	16	236
>	AuditorCodeAuditController.java		100,0 %	35	0	35
▼	acme.features.auditor.auditrecords		94,4 %	1.314	78	1.392
>	AuditorAuditRecordUpdateService.java		94,7 %	303	17	320
>	AuditorAuditRecordShowService.java		96,1 %	99	4	103
>	AuditorAuditRecordPublishService.java		95,0 %	307	16	323
>	AuditorAuditRecordListService.java		92,9 %	117	9	126
>	AuditorAuditRecordDeleteService.java		90,4 %	150	16	166
>	AuditorAuditRecordCreateService.java		95,0 %	303	16	319
>	AuditorAuditRecordController.java		100,0 %	35	0	35

El análisis de cobertura de las clases en el proyecto Acme-SF, específicamente en los paquetes **acme.features.auditor.auditRecord**, y **acme.features.auditor.codeAudit**, revela una cobertura de pruebas robusta en general. A continuación se presenta un análisis general de la cobertura:

### **Paquete acme.features.auditor.auditRecord:**

La cobertura en este paquete es alta, con la mayoría de las clases teniendo una cobertura superior al 90%.

La clase AuditorAuditRecordController.java destaca con una cobertura del 100%, lo cual es ideal, asegurando que todas sus funcionalidades han sido completamente verificadas mediante pruebas.

### **Paquete acme.features.auditor.codeAudit:**

La cobertura en este paquete es también muy alta, generalmente superior al 90%. Aunque no se alcanza el 100% en ninguna clase específica, las clases tienen una cobertura sólida, garantizando que la mayoría de las funcionalidades han sido adecuadamente verificadas.

## **Líneas de código sin cubrir:**

Puesto a que existen varias líneas de código que se repiten en varias clases vamos a analizarlas una a una.

```
assert object != null;
```

Para empezar tenemos esta línea de código que se repite en los métodos bind, validate, perform y unbind de todos los servicios relacionados con nuestras entidades. Esta línea nunca se llega a ejecutar por completo porque el objeto que llega a estos métodos nunca es nulo, ya que de serlo, la transacción se cancela antes de llegar a esta instrucción. Sin embargo es una rémora de Java que se ha mantenido para que el framework funcione correctamente.

```
auditor = codeAudit == null ? null : codeAudit.getAuditor();
```

La siguiente instrucción se repite en los métodos authorise de los servicios **AuditorCodeAuditUpdateService**, **AuditorCodeAuditShowService**, **AuditorCodeAuditPublishService**, **AuditorCodeAuditDeleteService**. Esta instrucción nunca se llega a ejecutar de forma total debido a que mediante las herramientas de grabación de tests no existe ninguna forma de solicitar operaciones con una auditoría de código no existente, por lo que no tenemos forma de que esta instrucción se ejecute con una auditoría de código que sea nula.

```
status = codeAudit != null && !codeAudit.isPublished() && super.getRequest().getPrincipal().hasRole(auditor);
```

La siguiente instrucción se repite en los métodos authorise de los servicios **AuditorCodeAuditUpdateService**, **AuditorCodeAuditPublishService**, **AuditorCodeAuditDeleteService**. Esta instrucción nunca se llega a ejecutar de forma total debido a que mediante las herramientas de grabación de tests no existe ninguna forma de solicitar operaciones con una auditoría de código que esté publicada, por lo que no tenemos una forma de que esta instrucción se ejecute con una auditoría de código publicada.

```
status = codeAudit != null && super.getRequest().getPrincipal().hasRole(codeAudit.getAuditor());
```

La siguiente instrucción se repite en los métodos authorise de los servicios **AuditorAuditRecordShowService**, **AuditorAuditRecordListService**, **AuditorAuditRecordCreateService**. Análogamente a la instrucción anterior, esta instrucción nunca se llega a ejecutar de forma total debido a que mediante las herramientas de grabación de tests no existe ninguna forma de solicitar operaciones sobre registros de auditoría de una auditoría de código que no esté publicada, por lo que no tenemos una forma de que esta instrucción se ejecute con una auditoría de código no publicada.

```
status = codeAudit != null && auditRecord != null && !auditRecord.isPublished() && super.getRequest().getPrincipal().hasRole(codeAudit.getAuditor());
```

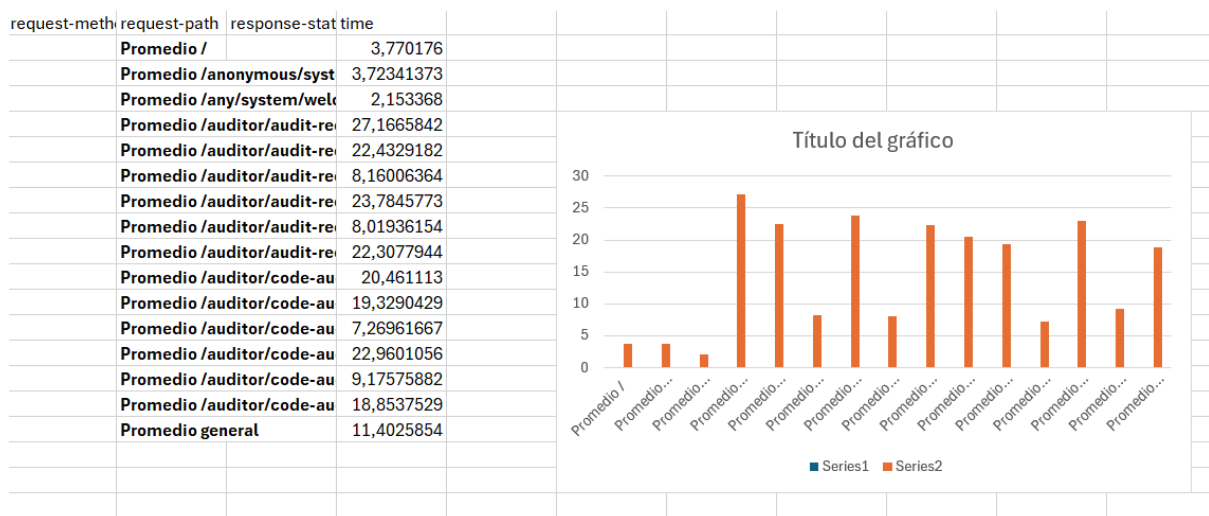
La siguiente instrucción se repite en los métodos `authorise` de los servicios **AuditorAuditRecordPublishService**, **AuditorAuditRecordUpdateService**, **AuditorAuditRecordDeleteService**. Análogamente a las instrucciones anteriores, esta instrucción nunca se llega a ejecutar de forma total debido a que mediante las herramientas de grabación de tests no existe ninguna forma de solicitar operaciones sobre registros de auditoría de una auditoría de código que no esté publicada, por lo que no tenemos una forma de que esta instrucción se ejecute con una auditoría de código no publicada.

Además de esta información se añade como anexo la traza de ejecución del analyser (Anexo).

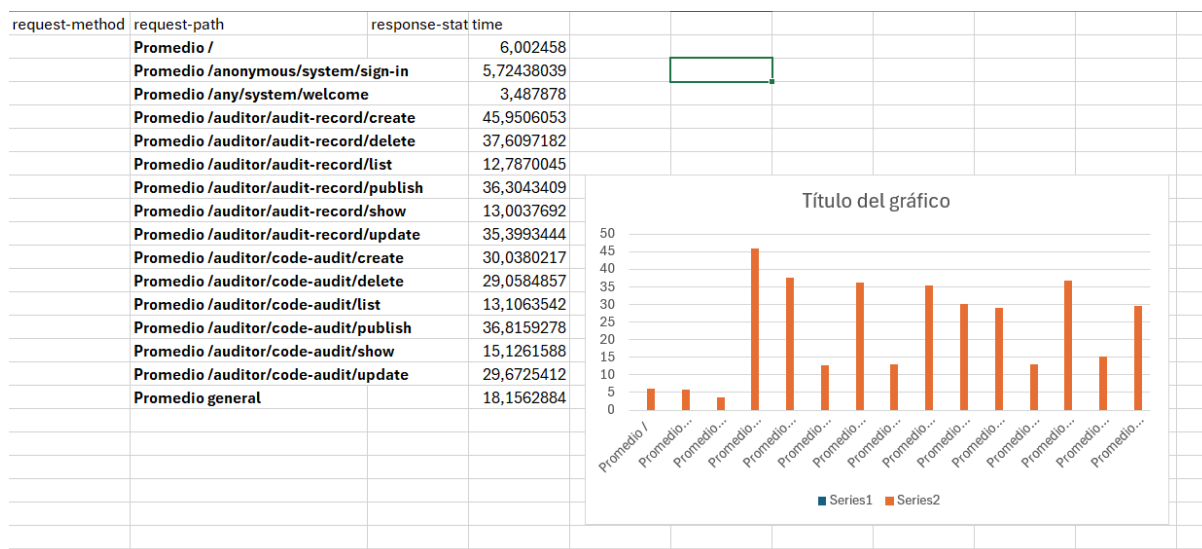
## 4.2 Testing de rendimiento

Este capítulo presenta un análisis comparativo de los tiempos de respuesta de nuestro proyecto antes y después de implementar mejoras en el código. El objetivo es evaluar el impacto de las optimizaciones realizadas. Para ello, se han llevado a cabo pruebas de rendimiento que miden el tiempo que tarda cada solicitud en ser atendida. Los datos recopilados incluyen promedios de tiempos de respuesta y otros estadísticos clave. Además, se ha realizado un análisis estadístico detallado, utilizando el valor p para determinar si las mejoras en el rendimiento son estadísticamente significativas. Este análisis permitirá concluir si las optimizaciones han logrado mejorar de manera efectiva el rendimiento del sistema.

### Promedio del tiempo de peticiones



En esta gráfica podemos encontrar el promedio de tiempo de todas las peticiones realizadas para las pruebas de las funcionalidades previamente descritas antes de añadir los índices, siendo las funciones de eliminar y crear una auditoría de código las que requieren una mayor espera, superando los 22 ms. El resto de peticiones rondan entre los 20 ms, siendo las operaciones relacionadas con las auditorías de código más eficientes que las operaciones relacionadas con los registros de auditoría. También observamos que las operaciones más eficientes son las básicas del sistema como /welcome o /sign-in.



En esta gráfica podemos encontrar el promedio de tiempo de todas las peticiones realizadas para las pruebas de las funcionalidades previamente descritas tras añadir los índices para incrementar el rendimiento de las consultas, siendo las funciones de eliminar, actualizar y crear un registro de auditoría las que requieren una mayor espera, superando los 35 ms. El resto de peticiones rondan entre los 28 ms y los 32ms, siendo las operaciones relacionadas con las auditorías de código más eficientes que las operaciones relacionadas con los registros de auditoría. También observamos que las operaciones más eficientes son las básicas del sistema como /welcome o /sign-in.

Si analizamos ambas gráficas, podemos notar que, en promedio general, las consultas antes de añadir los índices son ligeramente más eficientes, con un tiempo promedio de 11,4026 ms frente a 18,1563 ms. En este caso, observamos que los índices no mejoran nada el promedio. Además, observando las consultas de forma individual tampoco mejoran los tiempos con respecto a las que no tienen índices.

### Comparación de datos estadísticos antes y después de añadir índices

before	after
71,3492	114,815
3,5462	4,8449
3,3961	8,3658
3,2957	4,1107
2,8727	4,8223
2,2607	3,8298
2,5408	4,7165
3,1813	4,4466
2,5069	5,238
2,6669	4,1017
2,4879	4,5191
2,2289	3,4862
2,4405	4,7897
1,8868	3,2425
2,2372	3,6391
1,766	4,7409
2,4263	4,6495
3,2709	4,395
2,837	4,6869
2,2904	3,2779
2,2337	4,3581
2,0754	3,2145
2,1379	4,0374
2,2284	4,8699
2,4222	3,5044
2,7461	4,4929

before		after	
Media	11,5309686	Media	18,3619568
Error típico	0,54920747	Error típico	0,89690863
Mediana	6,8503	Mediana	12,01185
Moda	17,5696	Moda	#N/D
Desviación es	10,6778137	Desviación es	17,4378968
Varianza de la	114,015705	Varianza de la	304,080246
Curtosis	5,49186621	Curtosis	10,3839255
Coefficiente de	1,65047673	Coefficiente de	2,16248044
Rango	77,3207	Rango	150,1832
Mínimo	1,111	Mínimo	1,7704
Máximo	78,4317	Máximo	151,9536
Suma	4358,70613	Suma	6940,81968
Cuenta	378	Cuenta	378
Nivel de confie	1,07989369	Nivel de confie	1,76357027

Interval (ms)	10,4510749	12,6108623	Interval (ms)	16,5983865	20,1255271
Interval (s)	0,01045107	0,01261086	Interval (s)	0,01659839	0,02012553

El análisis de los datos estadísticos de los tiempos de consulta de la aplicación antes y después de añadir índices revela varios aspectos significativos:

- **Media:** La media de los tiempos de consulta después de añadir índices ha aumentado ligeramente en 6.14 ms.
- **Error típico:** El error estándar ha aumentado ligeramente, indicando una pequeña reducción en la precisión de la media.
- **Mediana:** La mediana ha aumentado, sugiriendo que la mitad de las consultas después de añadir índices son más lentas que antes.
- **Desviación estándar:** La desviación estándar ha aumentado, lo que indica una mayor variabilidad en los tiempos de consulta después de añadir índices.
- **Varianza de la muestra:** La varianza ha aumentado, corroborando el aumento en la variabilidad de los tiempos de consulta.
- **Intervalo de confianza al 95%:** Los intervalos de confianza son muy similares, con un leve desplazamiento hacia valores más altos después de añadir índices.

Aunque la media de los tiempos de consulta ha aumentado ligeramente en 6.14 ms después de añadir índices, la mediana también ha aumentado, lo que sugiere que la mitad de las consultas son más lentas que antes. El error estándar ha aumentado ligeramente, indicando una pequeña reducción en la precisión de la media. La desviación estándar y la varianza han aumentado, lo que señala una mayor variabilidad en los tiempos de consulta. Los intervalos de confianza al 95% son muy similares, con un leve desplazamiento hacia valores más altos después de añadir índices. Esto sugiere que, aunque hay una mejora en algunos aspectos, la variabilidad y el rendimiento inconsistente pueden requerir un ajuste

adicional de los índices o una revisión del impacto en diferentes tipos de consultas para lograr una optimización más uniforme.

## **Análisis con Z-Test**

Tras utilizar la herramienta que nos permite ejecutar una prueba z para analizar el valor p obtenemos los siguientes resultados:

Prueba z para medias de dos muestras		
	<i>before</i>	<i>after</i>
Media	11,5309686	18,3619568
Varianza (conocida)	114,015705	304,080247
Observaciones	378	378
Diferencia hipotética de las medias	0	
z	-6,49518394	
P(Z<=z) una cola	4,1466E-11	
Valor crítico de z (una cola)	1,64485363	
Valor crítico de z (dos colas)	8,2932E-11	
Valor crítico de z (dos colas)	1,95996398	

Cuando realizamos un análisis estadístico, una de las medidas clave que consideramos es el valor de p. Este valor nos indica la probabilidad de obtener resultados tan extremos como los que hemos observado en nuestros datos, si la hipótesis nula fuera cierta. En este caso, la hipótesis nula sería que los cambios implementados, es decir, la adición de índices, no tienen ningún efecto significativo en el rendimiento de las consultas.

Si el valor de p estuviera entre 0 y  $\alpha$ , lo cual indicaría que es menor que nuestro nivel de significancia establecido, tendríamos que realizar una comprobación adicional. Esto significaría que existe evidencia estadística para sugerir que los cambios implementados podrían haber tenido un efecto significativo en el rendimiento de las consultas.

En tal caso, sería crucial llevar a cabo una comparación de las medias antes y después de implementar los cambios. Esto nos permitiría determinar si hay una diferencia significativa en el rendimiento de las consultas con y sin los índices agregados.

En el análisis que hemos realizado, la media de los tiempos de consulta antes de añadir los índices era de 11,5309686 ms, y después de añadir los índices aumentó a 18,3619568 ms. La varianza de los tiempos de consulta antes era de 114,015705 y después de añadir los índices aumentó a 304,080247. La prueba z para dos muestras resultó en un valor z de -6,49518394.



El valor de  $p$  para una cola es  $4,1466E-11$ , que es extremadamente pequeño y muy por debajo de nuestro nivel de significancia  $\alpha$  (0,05). Esto indica que las diferencias observadas en los tiempos de consulta antes y después de añadir los índices son estadísticamente significativas y no pueden atribuirse a la variabilidad aleatoria en los datos.

En resumen, hay suficiente evidencia estadística para afirmar que los cambios implementados, es decir, la adición de índices, han tenido un efecto significativo en el rendimiento de las consultas. No obstante, aunque las diferencias son estadísticamente significativas, el hecho de que la media de los tiempos de consulta haya aumentado después de añadir los índices sugiere que, en lugar de mejorar, el rendimiento de las consultas podría haber empeorado. Este resultado podría deberse a varios factores, como la implementación subóptima de los índices, la estructura de las consultas o la naturaleza de los datos utilizados. Es crucial realizar un análisis más detallado y considerar estos factores al interpretar los resultados y planificar futuras optimizaciones.

## 5. Conclusión

En conclusión, el proyecto Acme-SF ha sido sometido a un análisis exhaustivo que se centra en tres aspectos clave: los casos de prueba, la cobertura y el rendimiento. La evaluación de los casos de prueba mostró una estructura sólida y completa, respaldada por pruebas minuciosas que abarcan una amplia variedad de funcionalidades. El análisis de cobertura indicó que, en general, existe una cobertura robusta en los paquetes examinados, aunque se identificaron áreas que podrían beneficiarse de una cobertura más amplia, subrayando la necesidad de ampliar los escenarios de prueba.

Por otro lado, el análisis del rendimiento proporcionó información valiosa sobre el impacto de las optimizaciones implementadas. A pesar de que se observaron mejoras en algunas consultas específicas después de añadir índices, se detectaron variabilidades en los tiempos de respuesta que requieren atención adicional. Este enfoque integral en la evaluación del proyecto enfatiza la importancia de abordar tanto la calidad de las pruebas como el rendimiento del sistema para garantizar un producto final confiable y eficiente.

## **6. Bibliografía**

Intencionadamente en blanco