

Universidad de Sevilla

Escuela Técnica Superior de Ingeniería Informática

## Testing Report



**ACME SOFTWARE FACTORY**

OUR FIRST PROJECT IN D&T

Grado en Ingeniería Informática – Ingeniería del Software

Diseño y Pruebas 2

Curso 2023 – 2024

**Grupo de prácticas: C2-009**

**Autores por orden alfabético:**

Martínez Cano, Juan

	Diseño y Pruebas II Acme-Software-Factory
	<b>Testing Report</b>


## Índice de contenido

1. Resumen ejecutivo .....	3
2. Tabla de revisiones.....	4
3. Introducción .....	5
4. Contenido.....	6
5. Conclusiones .....	13
6. Bibliografía .....	15

	Diseño y Pruebas II Acme-Software-Factory
	<b>Testing Report</b>

## 1. Resumen ejecutivo

En este informe se presenta un análisis detallado de las pruebas y los resultados obtenidos para las funcionalidades correspondientes al módulo Student 5.

	Diseño y Pruebas II Acme-Software-Factory
	<b>Testing Report</b>

## 2. Tabla de revisiones

Fecha	Versión	Descripción
25/05/2024	1.0	Primera versión del documento.
22/06/2024	1.1	Correcciones en el documento.
06/07/2024	1.2	Actualización de los datos.


	Diseño y Pruebas II Acme-Software-Factory
	<b>Testing Report</b>

### 3. Introducción

El documento se divide en dos partes principales: pruebas funcionales y pruebas de rendimiento. El objetivo de este informe es proporcionar un análisis detallado de los casos de prueba y del rendimiento de los mismo.

La primera parte del informe abarca las pruebas funcionales, las cuales se centran en evaluar las funcionalidades de dos entidades específicas: `code_audits` y `audits_records`. Se presentan los casos de prueba realizados, organizados por entidad, con descripciones claras y precisas para cada uno. Esto permite una evaluación concisa y efectiva de las funcionalidades asociadas a `code_audits` y `audits_records`.

En la segunda parte, el documento se centrará en proporcionar gráficos detallados que muestran los tiempos de respuesta del sistema durante las pruebas funcionales mencionadas anteriormente. Se comparará el desempeño del proyecto en dos entornos distintos, proporcionando un análisis más exhaustivo del rendimiento. Esta comparación permitirá identificar las diferencias en eficiencia y capacidad de manejo de carga entre los dos sistemas evaluados.

	Diseño y Pruebas II Acme-Software-Factory
	Testing Report

## 4. Contenido

### Pruebas funcionales.

#### Requirement #6

















Para este requisito, se han implementado las siguientes pruebas:


- **List:** Se ha verificado que el listado de los code audits se realiza correctamente, mostrando todos los elementos esperados.
- **Show:** Se ha comprobado que al acceder a un code audit, se muestran correctamente los datos específicos de ese code audit.
- **Update:** Se ha verificado que las actualizaciones se realizan correctamente. Además, se ha probado que, al actualizar algún campo con valores incorrectos, se muestra el error pertinente. La metodología utilizada para comprobar las restricciones consiste en revisar input por input, probando el límite inferior, el límite inferior +1, el límite superior, el límite superior +1 y algunos valores singulares que también podrían causar fallos.
- **Delete:** Se ha confirmado que se puede eliminar un code audit sin ningún problema.
- **Publish:** Se ha comprobado que se puede publicar un code audit sin impedimentos, verificando también que se apliquen las restricciones adecuadas cuando no se cumplen todos los requisitos necesarios.
- **Create-negative:** Se han verificado todas las restricciones posibles en los campos de creación de un code audit, siguiendo la metodología mencionada, la cual incluye probar el límite inferior, el límite inferior +1, el límite superior, el límite superior +1 y valores singulares que puedan causar errores.
- **Create-positive:** Se ha creado una batería de code audits para comprobar los campos a rellenar con valores máximos y mínimos, así como algunos casos particulares como intentos de script injection. Para cada input, se han seguido los mismos criterios metodológicos, asegurando también la correcta validación de casos positivos.
- **Wrong-role:** Este test comprueba que un usuario anónimo y un cliente no puede realizar las acciones de un auditor.
- **Wrong-user:** Se ha verificado que al iniciar sesión como el auditor2, no se puede acceder a los code audits del auditor1 ni realizar acciones sobre ellos.
- **Wrong-action:** Se ha iniciado sesión como auditor1 y se ha comprobado que no se puede realizar acciones prohibidas como la publicación, actualización y borrado de los code audit ya publicados.

Estos son solo algunos ejemplos, ya que se han comprobado muchas más cosas para asegurar el correcto funcionamiento y seguridad del sistema.

	Diseño y Pruebas II Acme-Software-Factory
	<b>Testing Report</b>

El cuanto al coverage, nos queda un total del 94,4% de la funcionalidad cubierta por los test.

▼  acme.features.auditor.codeAudit	 94,4 %
>  AuditorCodeAuditController.java	 100,0 %
>  AuditorCodeAuditShowService.java	 97,2 %
>  AuditorCodeAuditListService.java	 95,6 %
>  AuditorCodeAuditUpdateService.java	 95,2 %
>  AuditorCodeAuditPublishService.java	 95,1 %
>  AuditorCodeAuditCreateService.java	 94,7 %
>  AuditorCodeAuditDeleteService.java	 84,8 %

	Diseño y Pruebas II Acme-Software-Factory
	Testing Report

Como podemos observar, la funcionalidad número 6 queda prácticamente cubierta, con la excepción del delete, ya que en este caso no se ejecuta nunca la función de unbind, quedandose así sin cubrir por las pruebas. Se han probado prácticamente todas las líneas de código posibles con el objetivo de asegurar una buena ejecución del código.

#### Requirement #7

Para este requisito, se han implementado las siguientes pruebas:









- **List:** Se ha comprobado que el listado de los audit records se realiza de manera correcta, mostrando todos los elementos esperados.
- **Show:** Se ha verificado que al acceder a un audit record, se muestran correctamente los datos específicos de ese audit record.
- **Update:** Se ha comprobado que las actualizaciones se realizan correctamente. Además, se ha probado que al actualizar algún campo con valores incorrectos, se muestra el error pertinente. La metodología utilizada para comprobar las restricciones consiste en revisar input por input, probando el límite inferior, el límite inferior +1, el límite superior, el límite superior +1 y algunos valores singulares que también podrían causar fallos.
- **Delete:** Se ha confirmado que se puede eliminar un audit record sin ningún problema.
- **Publish:** Se ha verificado que se puede publicar un audit record sin impedimentos, comprobando también que se apliquen las restricciones adecuadas cuando no se cumplen todos los requisitos necesarios.
- **Create-negative:** Se han verificado todas las restricciones posibles en los campos de creación de un audit record, siguiendo la metodología mencionada, que incluye probar el límite inferior, el límite inferior +1, el límite superior, el límite superior +1 y valores singulares que puedan causar errores.
- **Create-positive:** Se ha creado una batería de audit records para comprobar los campos a rellenar con valores máximos y mínimos, así como algunos casos particulares como intentos de script injection. Para cada input, se han seguido los mismos criterios metodológicos, asegurando también la correcta validación de casos positivos.
- **Wrong-role:** Este test comprueba que un usuario anónimo y un cliente no pueden realizar las acciones de un auditor sobre los audit record.
- **Wrong-user:** Se ha verificado que al iniciar sesión como el auditor2, no se puede acceder a los audit records del auditor1 ni realizar acciones sobre ellos.
- **Wrong-action:** Se ha iniciado sesión como auditor1 y se ha comprobado que no se puede realizar acciones prohibidas como la publicación, actualización y borrado de los audit records ya publicados, así como la creación de los mismos sobre code audits ya publicados.



	Diseño y Pruebas II Acme-Software-Factory
	<b>Testing Report</b>

Estos son solo algunos ejemplos, ya que se han comprobado muchas más cosas para asegurar el correcto funcionamiento y seguridad del sistema.

En cuanto al coverage, se ha realizado una cobertura del 95,1% de la funcionalidad gracias a los test.

▼	acme.features.auditor.auditRecord		95,1 %
>	AuditorAuditRecordController.java		100,0 %
>	AuditorAuditRecordListService.java		97,0 %
>	AuditorAuditRecordShowService.java		96,3 %
>	AuditorAuditRecordPublishService.java		96,2 %
>	AuditorAuditRecordUpdateService.java		96,1 %
>	AuditorAuditRecordCreateService.java		95,9 %
>	AuditorAuditRecordDeleteService.java		83,7 %

Como podemos observar, la funcionalidad número 7 queda prácticamente cubierta, con la excepción del delete, ya que en este caso no se ejecuta nunca la función de unbind, quedandose así sin cubrir por las pruebas. Se han probado prácticamente todas las líneas de código posibles con el objetivo de asegurar una buena ejecución del código.

### Pruebas de rendimiento.

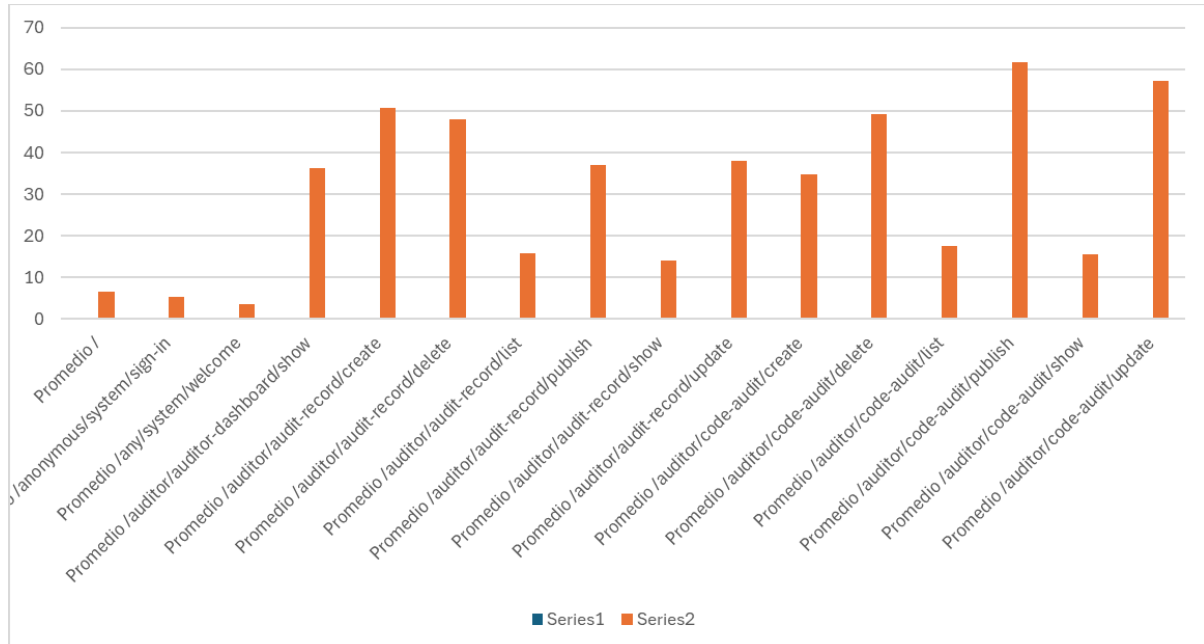
Rendimiento con pc personal

Se ha realizado un estudio del rendimiento de las pruebas anteriores, se ha generado esta gráfica para observar claramente cuáles son las pruebas más ineficientes.



Diseño y Pruebas II  
Acme-Software-Factory  
**Testing Report**

request-path	response-status	time
Promedio /		6.54034512
Promedio /anonymous/system/sign-in		5.28543171
Promedio /any/system/welcome		3.46010493
Promedio /auditor/auditor-dashboard/show		36.3512
Promedio /auditor/audit-record/create		50.7530716
Promedio /auditor/audit-record/delete		47.8989402
Promedio /auditor/audit-record/list		15.6539702
Promedio /auditor/audit-record/publish		36.9554365
Promedio /auditor/audit-record/show		14.146912
Promedio /auditor/audit-record/update		37.8902046
Promedio /auditor/code-audit/create		34.8221978
Promedio /auditor/code-audit/delete		49.2259798
Promedio /auditor/code-audit/list		17.408842
Promedio /auditor/code-audit/publish		61.6741659
Promedio /auditor/code-audit/show		15.5785
Promedio /auditor/code-audit/update		57.2772116
Promedio general		30.3165517



Como podemos observar, las peticiones más ineficientes es claramente el publish y update de code-audit

	Diseño y Pruebas II Acme-Software-Factory
	Testing Report

También se ha calculado el intervalo de confianza del 95 % para el tiempo que tarda en atender las solicitudes anteriores mi portátil. Nos quedaría:

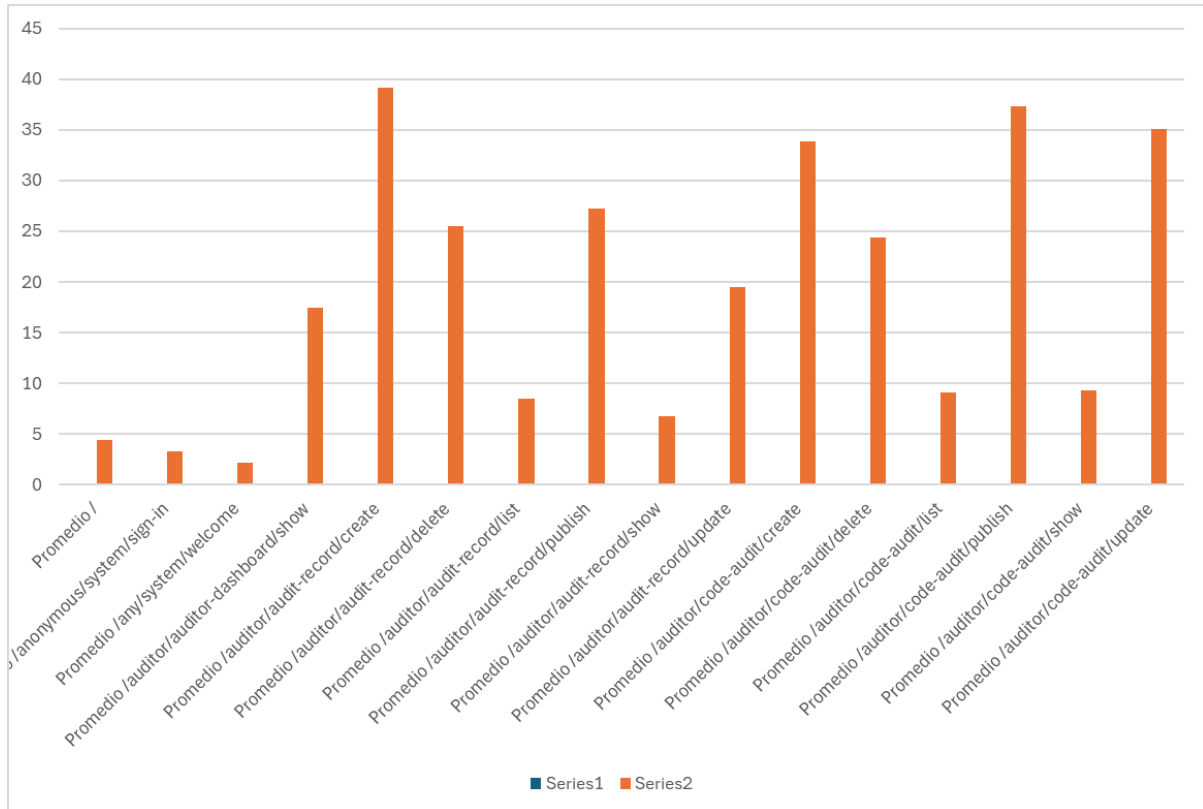
Interval (ms)	28.2674305	32.3656729
Interval (s)	0.02826743	0.03236567

Los intervalos de confianza del 95% para el tiempo de ejecución de un proceso están entre 28.2674 ms y 32.3657 ms, lo que indica que podemos estar 95% seguros de que el tiempo promedio de ejecución se encuentra dentro de este rango.

Rendimiento con pc de Eduardo Bustamante Lucena:

Se ha vuelto a realizar un estudio del rendimiento de las pruebas anteriores, pero esta vez con el ordenador de mi compañero, se ha generado esta gráfica para observar claramente cuáles son las pruebas más ineficientes.

equest-path	response-status	time
promedio /		4.41571429
promedio /anonymous/system/sign-in		3.30162927
promedio /any/system/welcome		2.1077881
promedio /auditor/auditor-dashboard/show		17.4532
promedio /auditor/audit-record/create		39.2034235
promedio /auditor/audit-record/delete		25.54196
promedio /auditor/audit-record/list		8.5007807
promedio /auditor/audit-record/publish		27.2785827
promedio /auditor/audit-record/show		6.71605882
promedio /auditor/audit-record/update		19.4739698
promedio /auditor/code-audit/create		33.8697467
promedio /auditor/code-audit/delete		24.33928
promedio /auditor/code-audit/list		9.04819194
promedio /auditor/code-audit/publish		37.3499951
promedio /auditor/code-audit/show		9.296
promedio /auditor/code-audit/update		35.0616419
promedio general		21.0936354



Como podemos observar, las peticiones más ineficientes son las publish y create de los code audits y audit record respectivamente.

También se ha calculado el intervalo de confianza del 95 % para el tiempo que tarda en atender las solicitudes anteriores. Nos quedaría:

Interval(ms)	19.3109024	22.8763685
Interval(s)	0.0193109	0.02287637

Los intervalos de confianza del 95% para el tiempo de ejecución de un proceso están entre 19.3109 ms y 22.8764 ms, lo que significa que podemos estar 95% seguros de que el tiempo promedio de ejecución se encuentra dentro de este rango.

## Profiling your project

Para identificar los cuellos de botella en el código se utilizó la herramienta VisualVM. Los resultados dieron cuenta de las áreas en las que se encuentran los problemas de rendimiento más importantes.



Diseño y Pruebas II  
Acme-Software-Factory  
Testing Report

Name	Self Time (CPU)		Total Time (CPU)
⌚ acme.features.auditor.auditRecord.AuditorAuditRecordCreateService.bind ()	0,0 ms	(-%)	2.242 ms (19,4%)
⌚ acme.features.auditor.codeAudit.AuditorCodeAuditCreateService.bind ()	0,0 ms	(-%)	1.538 ms (13,3%)
⌚ acme.features.auditor.auditRecord.AuditorAuditRecordPublishService.bind ()	0,0 ms	(-%)	1.343 ms (11,6%)
⌚ acme.features.auditor.codeAudit.AuditorCodeAuditUpdateService.bind ()	0,0 ms	(-%)	1.171 ms (10,1%)
⌚ acme.features.auditor.auditRecord.AuditorAuditRecordUpdateService.bind ()	0,0 ms	(-%)	1.166 ms (10,1%)
⌚ acme.features.auditor.codeAudit.AuditorCodeAuditPublishService.bind ()	0,0 ms	(-%)	1.053 ms (9,1%)
⌚ acme.features.auditor.auditRecord.AuditorAuditRecordCreateService.validate ()	0,0 ms	(-%)	409 ms (3,6%)
⌚ acme.features.auditor.auditRecord.AuditorAuditRecordPublishService.validate ()	0,0 ms	(-%)	271 ms (2,3%)
⌚ acme.features.auditor.codeAudit.AuditorCodeAuditListService.load ()	0,0 ms	(-%)	261 ms (2,3%)
⌚ acme.features.auditor.auditRecord.AuditorAuditRecordUpdateService.validate ()	0,0 ms	(-%)	214 ms (1,9%)
⌚ acme.features.auditor.auditRecord.AuditorAuditRecordCreateService.authorise ()	0,0 ms	(-%)	191 ms (1,7%)
⌚ acme.features.auditor.codeAudit.AuditorCodeAuditPublishService.validate ()	0,0 ms	(-%)	175 ms (1,5%)
⌚ acme.features.auditor.auditRecord.AuditorAuditRecordListService.unbind ()	0,0 ms	(-%)	158 ms (1,4%)
⌚ acme.features.auditor.codeAudit.AuditorCodeAuditCreateService.unbind ()	0,0 ms	(-%)	149 ms (1,3%)
⌚ acme.features.auditor.codeAudit.AuditorCodeAuditCreateService.validate ()	0,0 ms	(-%)	133 ms (1,2%)
⌚ acme.features.auditor.auditRecord.AuditorAuditRecordListService.authorise ()	0,0 ms	(-%)	124 ms (1,1%)
⌚ acme.features.auditor.codeAudit.AuditorCodeAuditUpdateService.validate ()	0,0 ms	(-%)	108 ms (0,9%)
⌚ acme.features.auditor.codeAudit.AuditorCodeAuditCreateService.load ()	0,0 ms	(-%)	107 ms (0,9%)
⌚ acme.features.auditor.codeAudit.AuditorCodeAuditUpdateService.unbind ()	0,0 ms	(-%)	92,8 ms (0,8%)
⌚ acme.features.auditor.codeAudit.AuditorCodeAuditShowService.unbind ()	0,0 ms	(-%)	77,0 ms (0,7%)
⌚ acme.features.auditor.auditRecord.AuditorAuditRecordCreateService.load ()	0,0 ms	(-%)	57,4 ms (0,5%)
⌚ acme.features.auditor.codeAudit.AuditorCodeAuditPublishService.unbind ()	0,0 ms	(-%)	41,8 ms (0,4%)
⌚ acme.features.auditor.codeAudit.AuditorCodeAuditUpdateService.authorise ()	0,0 ms	(-%)	34,8 ms (0,3%)
⌚ acme.features.auditor.auditRecord.AuditorAuditRecordDeleteService.bind ()	0,0 ms	(-%)	33,1 ms (0,3%)
⌚ acme.features.auditor.codeAudit.AuditorCodeAuditUpdateService.load ()	0,0 ms	(-%)	32,8 ms (0,3%)
⌚ acme.features.auditor.codeAudit.AuditorCodeAuditPublishService.load ()	0,0 ms	(-%)	32,3 ms (0,3%)
⌚ acme.features.auditor.codeAudit.AuditorCodeAuditDeleteService.bind ()	0,0 ms	(-%)	31,5 ms (0,3%)
⌚ acme.features.auditor.auditRecord.AuditorAuditRecordUpdateService.authorise ()	0,0 ms	(-%)	30,4 ms (0,3%)
⌚ acme.features.auditor.auditRecord.AuditorAuditRecordListService.load ()	0,0 ms	(-%)	27,7 ms (0,2%)
⌚ acme.features.auditor.codeAudit.AuditorCodeAuditShowService.authorise ()	0,0 ms	(-%)	27,5 ms (0,2%)
⌚ acme.features.auditor.auditRecord.AuditorAuditRecordShowService.load ()	0,0 ms	(-%)	26,2 ms (0,2%)

El análisis de rendimiento con VisualVM muestra que los métodos `bind()` y `validate()` de `AuditorAuditRecordCreateService`, `AuditorCodeAuditCreateService`, y `AuditorAuditRecordPublishService` consumen la mayor parte del tiempo de CPU, representando cada uno hasta el 19,4% del total. La optimización de estos métodos debería ser prioritaria para mejorar el rendimiento del sistema, ya que actualmente concentran el uso de recursos. Además, se recomienda un análisis más detallado de sus dependencias internas para identificar oportunidades adicionales de mejora.

## 5. Conclusiones

En conclusión, se ha cubierto casi en su totalidad las funcionalidades requeridas mediante las pruebas mencionadas. Los casos de prueba realizados han demostrado que tanto las entidades **code\_audits** como **audits\_records** funcionan correctamente bajo las condiciones evaluadas. Además, el análisis de rendimiento ha revelado una clara mejora al ejecutar las pruebas en el ordenador de mi compañero, con un notable incremento en la eficiencia y tiempos de respuesta del

	Diseño y Pruebas II Acme-Software-Factory
	<b>Testing Report</b>

sistema. Esta comparación entre diferentes entornos ha permitido identificar oportunidades para optimizar aún más el rendimiento del proyecto. En general, los resultados obtenidos son positivos y proporcionan una base sólida para futuras mejoras y desarrollos.

	Diseño y Pruebas II Acme-Software-Factory
	<b>Testing Report</b>

## 6. Bibliografía

<https://ev.us.es/>