

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:

Bustamante Lucena, Eduardo

	Diseño y Pruebas II Acme-Software-Factory
	Testing Report

Índice de contenido

1. Resumen ejecutivo	3
2. Tabla de revisiones.....	4
3. Introducción	5
4. Contenido.....	6
Pruebas funcionales.....	6
Requirement 6	6
Requirement 7	7
Pruebas de rendimiento	9
Profiling your project.....	11
5. Conclusiones	12

	Diseño y Pruebas II Acme-Software-Factory
	Testing Report

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 2.

	Diseño y Pruebas II Acme-Software-Factory
	Testing Report

2. Tabla de revisiones

Fecha	Versión	Descripción
26/05/2024	1.0	Primera versión del documento.
20/06/2024	1.1	Correcciones debido a los errores identificados en el First Call.


	Diseño y Pruebas II Acme-Software-Factory
	Testing Report

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: contracts y progress_log. 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 dichas entidades.

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.

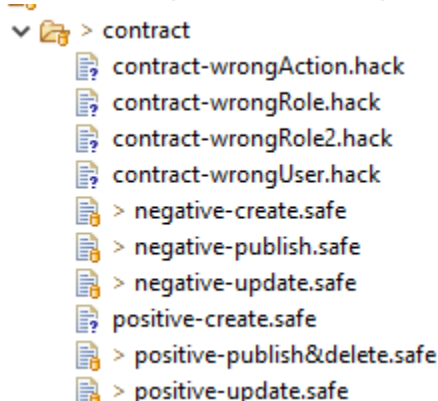
	Diseño y Pruebas II Acme-Software-Factory
	Testing Report

4. Contenido

Pruebas funcionales

Requirement 6

Para este requisito se han implementado las siguientes pruebas:



Contract-wrongAction.hack: Se realiza un registro con el client1 y se comprueba que no podemos borrar, actualizar o publicar un contrato ya publicado.

Contract-wrongRole.hack y Contract-wrongRole2.hack: Se comprueba que siendo un usuario anónimo no podemos realizar ninguna operación relacionada con los contratos (crear, publicar, actualizar y eliminar).

Contract-wrongUser.hack: En este caso nos registraremos como “client2” y se comprueba que no se pueden realizar operaciones sobre los contratos pertenecientes al “client1”.

Negative-create.safe: Se ha seguido la metodología explicada para comprobar todas las posibilidades negativas existentes en el formulario de creación de un contrato teniendo en cuenta las restricciones de cada campo.

Negative-publish.safe: Se ha seguido la metodología explicada para comprobar todas las posibilidades negativas existentes en el formulario de publicación de un contrato teniendo en cuenta las restricciones de cada campo.

Negative-update.safe: Se ha seguido la metodología explicada para comprobar todas las posibilidades negativas existentes en el formulario de actualización de un contrato teniendo en cuenta las restricciones de cada campo.

















Positive-create.safe: Se ha seguido la metodología explicada para comprobar todas las posibilidades positivas existentes en el formulario de creación de un contrato teniendo en cuenta las restricciones de cada campo.

Positive-publish&delete.safe: Se comprueba que se puede eliminar y publicar correctamente un contrato.

Positive-update.safe: Se ha seguido la metodología explicada para comprobar todas las posibilidades positivas existentes en el formulario de actualización de un contrato teniendo en cuenta las restricciones de cada campo.

	Diseño y Pruebas II Acme-Software-Factory
	Testing Report











Tras ejecutar el replayer obtenemos la siguiente cobertura:

▼  acme.features.clients.contracts	 92,8 %
>  ClientContractPublishService.java	 92,3 %
>  ClientContractUpdateService.java	 92,3 %
>  ClientContractDeleteService.java	 85,1 %
>  ClientContractCreateService.java	 95,4 %
>  ClientContractListService.java	 93,5 %
>  ClientContractShowService.java	 96,4 %
>  ClientContractController.java	 100,0 %

Como vemos, la mayoría de la funcionalidad está cubierta, aunque encontramos dos excepciones:

- En el delete puesto que, al realizar esta acción, no se está ejecutando el código de la función unbind.
- En el update, el código que queda sin probar es el cálculo de una variable que se utiliza para realizar la comparación del coste del proyecto con la suma del precio de los contratos.

Requirement 7

▼  > progressLog
>  negative-create.safe
>  negative-publish.safe
>  negative-update.safe
>  positive-create.safe
>  positive-publish&delete.safe
>  positive-update.safe
>  progressLog-wrongAction.hack
>  progressLog-wrongRole.hack
>  progressLog-wrongUser.hack


Negative-create.safe: Se ha seguido la metodología explicada para comprobar todas las posibilidades negativas existentes en el formulario de creación de un progress log teniendo en cuenta las restricciones de cada campo.

Negative-publish.safe: Se ha seguido la metodología explicada para comprobar todas las posibilidades negativas existentes en el formulario de publicación de un progress log teniendo en cuenta las restricciones de cada campo.

Negative-update.safe: Se ha seguido la metodología explicada para comprobar todas las posibilidades negativas existentes en el formulario de actualización de un progress log teniendo en cuenta las restricciones de cada campo.

Positive-create.safe: Se ha seguido la metodología explicada para comprobar todas las posibilidades positivas existentes en el formulario de creación de un contrato teniendo en cuenta las restricciones de cada campo.

Positive-publish&delete.safe: Se comprueba que se puede eliminar y publicar correctamente un progress log.

	Diseño y Pruebas II Acme-Software-Factory
	Testing Report

















Positive-update:

ProgressLog-wrongAction.hack: Se realiza un registro con el client1 y se comprueba que no podemos borrar, actualizar o publicar un progress log ya publicado.

ProgressLog-wrongRole.hack: Se comprueba que siendo un usuario anónimo no podemos realizar ninguna operación relacionada con los progress log (crear, publicar, actualizar y eliminar).

ProgressLog-wrongUser.hack: En este caso nos registraremos como “client2” y se comprueba que no se pueden realizar operaciones sobre los progress log pertenecientes al “client1”.

Tras ejecutar el replayer obtenemos la siguiente cobertura:

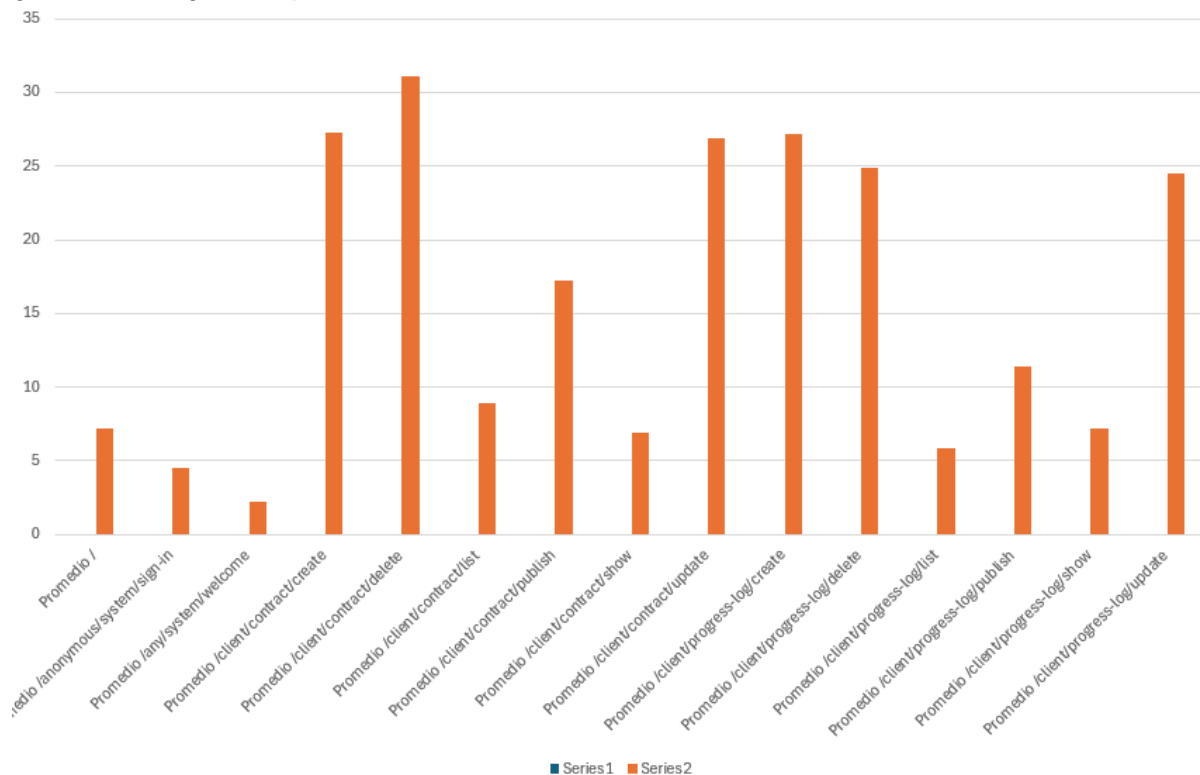
▼  acme.features.clients.progressLog		93,0 %
>  ClientProgressLogDeleteService.java		83,7 %
>  ClientProgressLogCreateService.java		93,8 %
>  ClientProgressLogPublishService.java		93,2 %
>  ClientProgressLogUpdateService.java		93,3 %
>  ClientProgressLogListService.java		95,0 %
>  ClientProgressLogShowService.java		96,4 %
>  ClientProgressLogController.java		100,0 %

Como vemos, la mayoría de la funcionalidad está cubierta, aunque parece que en el delete no del todo. Sin embargo, esto es lógico y se debe a que, al realizar esta acción, no se está ejecutando el código de la función unbind.

Pruebas de rendimiento

Rendimiento con PC personal:

Tras hacer un análisis de los resultados de reproducir las pruebas anteriores, podemos generar este gráfico que muestra cuáles son las solicitudes más ineficientes:



También se ha calculado el intervalo de confianza del 95 % para el tiempo que tarda en atender las solicitudes anteriores mi ordenador personal. Obteniendo como resultado:

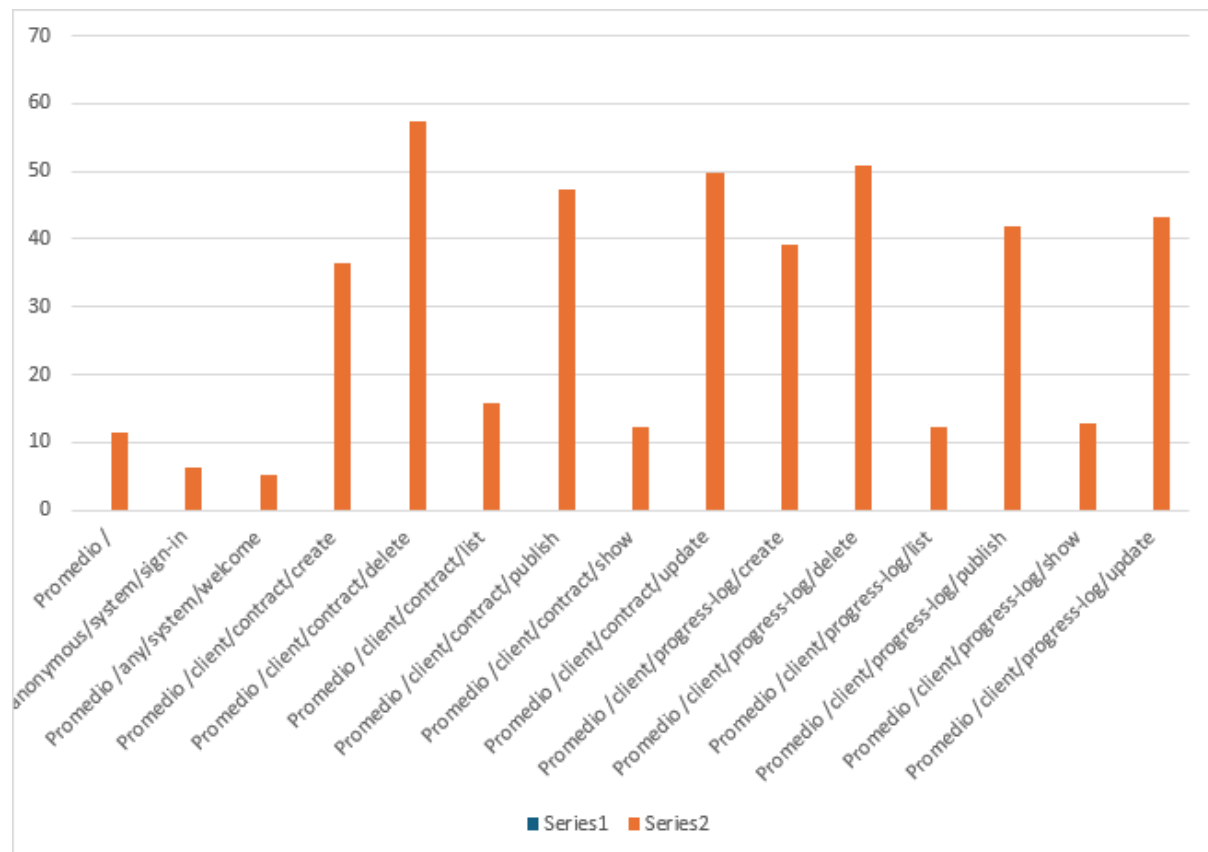
Interval (ms)	13,4138235	16,7116063
Interval (s)	0,01341382	0,01671161



Rendimiento con PC compañero:

Se han realizado los mismos test desde el PC de un compañero, y hemos obtenido que las consultas más ineficientes son:

- Client/contract/delete
- Client/progress-log/delete



También se ha vuelto a calcular el intervalo de confianza del 95 % para el tiempo que tarda en atender las solicitudes anteriores mi ordenador personal. Obteniendo como resultado:

Interval(ms)	24,2885251	28,7129954
Interval(s)	0,02428853	0,028713



Diseño y Pruebas II

Acme-Software-Factory

Testing Report

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.

acme.features.clients.contracts.ClientContractListService.load ()	0,0 ms	(-%)	267 ms	(15,1%)
acme.features.clients.contracts.ClientContractUpdateService.validate ()	0,0 ms	(-%)	147 ms	(8,4%)
acme.features.clients.contracts.ClientContractShowService.authorise ()	0,0 ms	(-%)	125 ms	(7,1%)
acme.features.clients.progressLog.ClientProgressLogUpdateService.authorise ()	0,0 ms	(-%)	116 ms	(6,6%)
acme.features.clients.progressLog.ClientProgressLogListService.authorise ()	0,0 ms	(-%)	112 ms	(6,4%)
acme.features.clients.progressLog.ClientProgressLogCreateService.authorise ()	0,0 ms	(-%)	107 ms	(6,1%)
acme.features.clients.progressLog.ClientProgressLogShowService.authorise ()	0,0 ms	(-%)	88,2 ms	(5%)
acme.features.clients.progressLog.ClientProgressLogPublishService.authorise ()	0,0 ms	(-%)	79,8 ms	(4,5%)
acme.features.clients.contracts.ClientContractPublishService.validate ()	0,0 ms	(-%)	71,5 ms	(4%)
acme.features.clients.contracts.ClientContractCreateService.unbind ()	0,0 ms	(-%)	69,8 ms	(4%)
acme.features.clients.contracts.ClientContractCreateService.load ()	0,0 ms	(-%)	67,8 ms	(3,8%)
acme.features.clients.contracts.ClientContractCreateService.validate ()	0,0 ms	(-%)	62,2 ms	(3,5%)
acme.features.clients.contracts.ClientContractCreateService.bind ()	0,0 ms	(-%)	54,8 ms	(3,1%)
acme.features.clients.contracts.ClientContractUpdateService.authorise ()	0,0 ms	(-%)	51,7 ms	(2,9%)
acme.features.clients.contracts.ClientContractUpdateService.validatorProjectCost ()	0,0 ms	(-%)	42,3 ms	(2,4%)
acme.features.clients.contracts.ClientContractShowService.load ()	0,0 ms	(-%)	36,8 ms	(2,1%)
acme.features.clients.contracts.ClientContractListService.authorise ()	0,0 ms	(-%)	36,5 ms	(2,1%)
acme.features.clients.progressLog.ClientProgressLogCreateService.load ()	0,0 ms	(-%)	24,7 ms	(1,4%)
acme.features.clients.contracts.ClientContractUpdateService.load ()	0,0 ms	(-%)	23,5 ms	(1,3%)
acme.features.clients.progressLog.ClientProgressLogListService.load ()	0,0 ms	(-%)	22,4 ms	(1,3%)
acme.features.clients.progressLog.ClientProgressLogDeleteService.bind ()	0,0 ms	(-%)	22,4 ms	(1,3%)
acme.features.clients.contracts.ClientContractUpdateService.bind ()	0,0 ms	(-%)	21,5 ms	(1,2%)
acme.features.clients.contracts.ClientContractCreateService.perform ()	0,0 ms	(-%)	20,2 ms	(1,1%)
acme.features.clients.progressLog.ClientProgressLogPublishService.load ()	0,0 ms	(-%)	15,4 ms	(0,9%)
acme.features.clients.contracts.ClientContractDeleteService.bind ()	0,0 ms	(-%)	15,2 ms	(0,9%)
acme.features.clients.progressLog.ClientProgressLogCreateService.bind ()	0,0 ms	(-%)	15,1 ms	(0,9%)
acme.features.clients.contracts.ClientContractListService.unbind ()	0,0 ms	(-%)	12,9 ms	(0,7%)
acme.features.clients.progressLog.ClientProgressLogPublishService.validate ()	0,0 ms	(-%)	11,2 ms	(0,6%)
acme.features.clients.progressLog.ClientProgressLogUpdateService.bind ()	0,0 ms	(-%)	10,4 ms	(0,6%)
acme.features.clients.contracts.ClientContractCreateService.authorise ()	0,0 ms	(-%)	4,66 ms	(0,3%)
acme.features.clients.progressLog.ClientProgressLogDeleteService.authorise ()	0,0 ms	(-%)	4,26 ms	(0,2%)
acme.features.clients.progressLog.ClientProgressLogCreateService.perform ()	0,0 ms	(-%)	4,1 ms	(0,2%)
acme.features.clients.contracts.ClientContractPublishService.authorise ()	0,0 ms	(-%)	0,0 ms	(0%)

	Diseño y Pruebas II Acme-Software-Factory
	Testing Report

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 contract como progress_log funcionan correctamente bajo las condiciones evaluadas.

En cuanto a la comparativa de rendimiento se puede observar que no se obtienen resultados parecidos ya que nuestros equipos no cuentan con características similares, siendo el de mi compañero peor en cuanto a componentes se refiere.

No obstante, aprovechando que contamos con dos muestras de datos, se ha realizado una **prueba Z para medias de dos muestras**, para poder obtener más información respecto a la comparación. De esta podemos destacar el *valor crítico de z (dos colas)*, el cual resulta ser: 0,46253216

	Diseño y Pruebas II Acme-Software-Factory
	Testing Report

Recalcar que toda la información referente a las pruebas mostradas se encuentra en la rama "edubusluc-FixJuly" por si se desea reproducir o verificar los resultados obtenidos.

6. Bibliografía

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