

INFORME DE PRUEBAS



**Grado en Ingeniería Informática – Ingeniería del
Software**





Diseño y Pruebas 2
Curso 2023-2024

José García de Tejada Delgado

Índice

1. Información general del proyecto	3
2. Resumen del ejecutivo	4
3. Tabla de revisión	5
4. Introducción	6
5. Contenidos	7
5.1 Pruebas funcionales	7
5.2 Pruebas de rendimiento	14
6. Conclusiones	17
7. Bibliografía	18

1. Información general del proyecto

NOMBRE DEL PROYECTO			Acme-SF	
PARTICIPANTES				
Nombre	Email	Rol	Nombre de usuario	Foto
Antonio Daniel Porcar Aragón	antporara@alum.us.es	Project Manager, Desarrollador	antporara	
Francisco Miguel Jiménez Morales	frajimmor2@alum.us.es	Tester, Desarrollador	frajimmor2	
Javier Santos Martín	javsanmar5@alum.us.es	Desarrollador, Secretario	javsanmar5	
Javier Ruiz Garrido	javruigar2@alum.us.es	Analista, Desarrollador	Javiruizg	
José García de Tejada Delgado	josgardel8@alum.us.es	Operador, Desarrollador	JoseGTD	
Interesados				
Francisco Miguel Jiménez Morales, Javier Ruiz Garrido, José García de Tejada Delgado, Javier Santos Martín, Antonio Daniel Porcar Aragón and José González Enriquez (the professor).				
Fecha de inicio	Fecha esperada de completado	Entregables	Fecha del documento	
12/02/2024	27/05/2024	4	24/05/2024	

2. Resumen del ejecutivo

Las pruebas son un elemento fundamental en el desarrollo y mantenimiento software, ayudando a los miembros del equipo de desarrollo a comprobar el adecuado funcionamiento del sistema y la detección de errores.

En este documento se refleja un informe sobre el que ha realizado el estudiante 2, mediante los capítulos de pruebas funcionales y pruebas de rendimiento; además se analizará la cobertura de dichas pruebas.

3. Tabla de revisión

Versión	Descripción	Fecha
v1.0	Versión inicial	2/07/2024
v2.0	Versión final	4/07/2024

4. Introducción

En el primer apartado, sobre pruebas funcionales se tratarán los casos de prueba implementados agrupados por implementación, incluyéndose su efectividad a la hora de detectar errores.

En el segundo apartado se trata el rendimiento del sistema, mediante gráficos y análisis estadístico de los tiempos en que se sirven las peticiones, tomando intervalos de confianza del 95%, previos y posteriores a la refactorización con índices; junto con un contraste de hipótesis resultante al análisis.

5. Contenidos

5.1 Pruebas funcionales

A continuación, se indicarán de manera resumida los casos de prueba implementados, indicándose una explicación adicional en las pruebas que permitieron detectar errores en la implementación de las funcionalidades. Considérese que en aquellos casos en que la prueba falló por un error encontrado o se encontró un resultado inesperado durante la realización de la misma, se arregló dicho error y se repitió la prueba, asegurándonos así de su adecuado funcionamiento. También cabe destacar que en todos los casos en los que se menciona la obtención de un error 500, este es un Access is not authorised.

Contract

- **create.safe:**
Se inicia sesión como client1 y se accede al formulario de creación de contratos. Se prueban todos los tipos de entradas posibles para los distintos campos del formulario, tratando de cubrir todos los errores posibles (formulario vacío, formatos de códigos y fechas incorrectos, códigos duplicados, fechas en el futuro, proyecto no seleccionado y también dinero negativo o mayor al del proyecto asociado al objeto). Se detectaron múltiples errores alrededor de las fechas y el proyecto vacío.
- **create.hack:**
Se abre una pestaña privada y se intenta acceder al link del formulario de creación de contratos sin haber iniciado sesión, después se inicia sesión como administrator1 y se vuelve a intentar. Se obtiene error 500 en los dos casos
- **list.safe:**
Se inicia sesión con client1 y se intenta acceder al listado de contratos, no hay errores.
- **list.hack:**
Se abre una pestaña privada y se intenta acceder al link del listado de contratos, luego se inicia sesión con la cuenta administrator1 y se vuelve a intentar, se obtiene error 500 en los dos casos.
- **update.safe:**
Se inicia sesión como client1 y se accede al formulario de un contrato no publicado. Se prueba un abanico de opciones similar al del create.safe, pero también se intenta asignar fechas anteriores al último progress log publicado.
- **update.hack:**
Se abre una pestaña de incógnito y se intenta acceder al link del formulario de actualización del contrato sin iniciar sesión, después se inicia sesión como administrator1 y se vuelve a intentar, se obtiene error 500 en los dos casos.

- **delete.safe:**
Se inicia sesión como client1, se accede a un contrato no publicado y se borra mediante el botón que lo facilita. No se encontraron errores.
- **delete.hack:**
Se abre una pestaña de incógnito y se intenta acceder al link de delete sin iniciar sesión, luego se inicia sesión como administrator1 y se vuelve a intentar. Se obtiene error 500 las dos veces.
- **show.safe:**
Se inicia sesión como client1 y se accede a los detalles de un contrato. No se encontraron errores.
- **show.hack:**
Se abre una pestaña de incógnito y se intenta acceder al link de show con la id de un contrato existente perteneciente a client1, después con la de uno que no exista. Se inicia sesión con administrator1 y se repiten las dos acciones anteriores. Se inicia sesión con client2 y se vuelve a repetir las dos acciones anteriores. En todos los casos se obtiene error 500.
- **publish.safe:**
Se inicia sesión como client1 y se accede a un contrato no publicado. Se repiten las pruebas del update.safe, añadiendo la prueba de que el presupuesto del contrato, junto con el del resto de contratos, sume más que el presupuesto del proyecto asociado (sin que el contrato en específico llegue a sobrepasarlo). No se encontraron errores.
- **publish.hack:**
Se abre una pestaña de incógnito y se intenta acceder al link de publish con la id de un contrato existente perteneciente a client1, después con la de uno que no exista. Se inicia sesión con administrator1 y se repiten las dos acciones anteriores. Se inicia sesión con client2 y se vuelve a repetir las dos acciones anteriores. En todos los casos se obtiene error 500.

Progress Log

- **list.safe:**
Se inicia sesión como client1 y se accede su listado de registros de progreso. No se encontraron errores.
- **list.hack:**
Se abre una pestaña de incógnito y se intenta acceder al link del listado registros de progreso de un contrato existente de client1, y después al de un contrato inexistente, después se inicia sesión con administrator1 y se vuelve a intentar las dos acciones. Por último se inicia sesión con client2 y se intenta de nuevo acceder al listado de registros de progreso del contrato de client1. Se obtienen error 500 en todos los casos.
- **create.safe:**
Se inicia sesión como client1 y se accede al formulario de creación de registros de progreso desde un contrato no publicado. Se prueban todos los tipos de entradas posibles para los distintos campos del formulario, tratando de cubrir todos los errores posibles (formulario vacío, formatos de códigos y fechas incorrectos, códigos duplicados, fechas en el futuro, fechas anteriores a la del contrato asociado, fechas anteriores a las de registros de progreso ya publicados, completitud del contrato negativo o menor al último registro de progreso publicado, etc). Se detectaron múltiples errores alrededor de las fechas y la completitud de proyecto.
- **create.hack:**
Se abre una pestaña de incógnito y se intenta acceder al link del formulario de creación de registro de progreso con la id de un contrato existente no publicado perteneciente a client1, y luego con la de uno que no exista. Se inicia sesión con administrator1 y se repiten los dos últimos pasos. Se inicia sesión con client2 y se repiten los dos últimos pasos. Se inicia sesión con client1 y se intenta con la id de un contrato existente ya publicado. En todos los casos se obtiene un error 500.
- **update.safe:**
Se inicia sesión como client1 y se accede al formulario de actualización de un registro de progreso no publicado. Se intentan las mismas acciones que en el create.safe. No se encontraron errores.
- **update.hack:**
Se abre una pestaña de incógnito y se intenta acceder al link del formulario de actualización de registro de progreso. Se inicia sesión con administrator1 y se vuelve a intentar. Se obtiene error 500 en los dos casos.
- **show.safe:**
Se inicia sesión como client1 y se accede a uno de los registros de progreso que le pertenecen. No se encontraron errores.
- **show.hack:**
Se abre una pestaña de incógnito y se intenta acceder al link de un registro de progreso existente perteneciente a client1, y después a uno que no exista. Se inicia sesión con administrator1 y se vuelven a intentar las dos acciones. Se inicia sesión con client2 y se vuelve a intentar acceder al registro de progreso de client1. En todos los casos se obtiene error 500.

- **delete.safe:**
Se inicia sesión como client1 y se intenta borrar un registro de progreso no publicado. No se encontraron errores.
- **delete.hack:**
Se abre una pestaña de incógnito y se intenta acceder al link de delete de registro de progreso. Se inicia sesión con administrator1 y se vuelve a intentar. Se encuentra error 500 en los dos casos
- **publish.safe:**
Se inicia sesión con client1 y se accede a un registro de progreso no publicado. Se repiten las acciones del update.safe. No se encontraron errores.
- **publish.hack:**
Se abre una pestaña de incógnito y se intenta acceder al link de publicación de un registro de progreso perteneciente a client1, y luego a uno que no existe. Se inicia sesión con administrator1 y se repiten las dos acciones. Se inicia sesión con client2 y se repiten las dos acciones. Se inicia sesión con client1 y se intenta acceder al link de publicación de un registro de progreso ya publicado. Se obtiene error 500 en todos los casos.

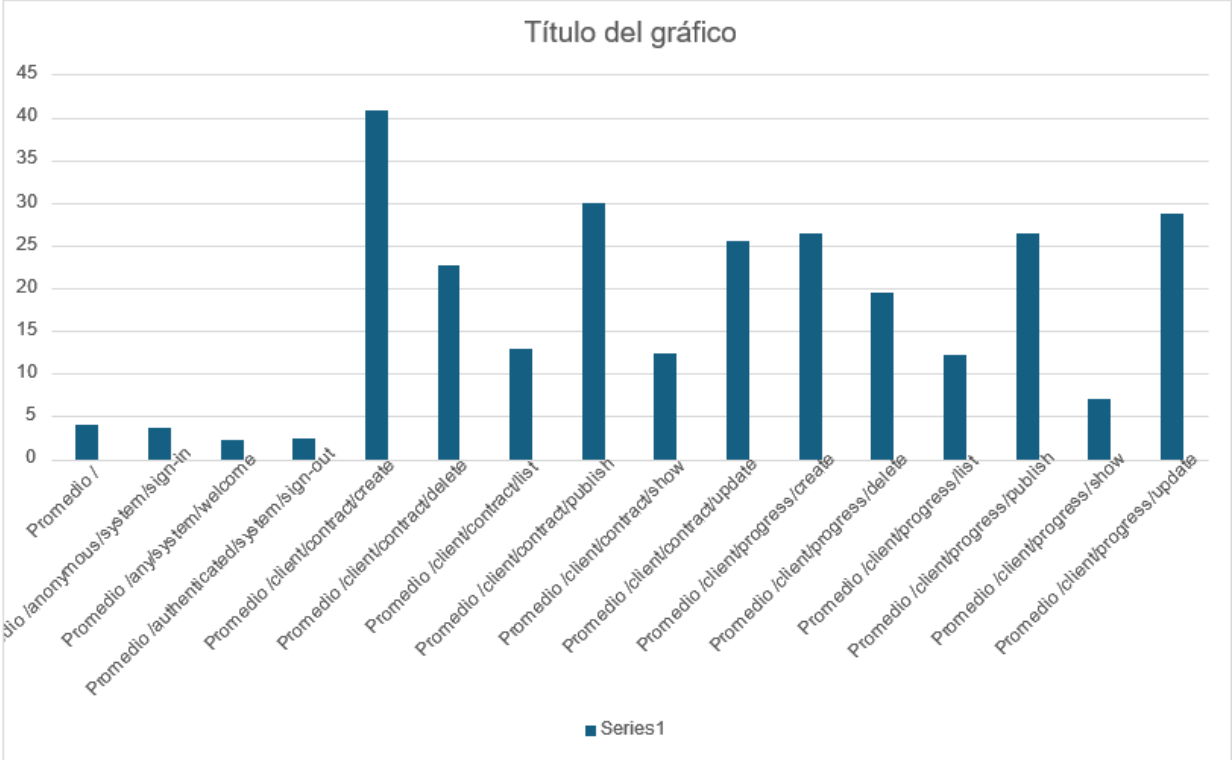
El coverage final de los tests es el siguiente:

▼	acme.features.client.contract	90,5 %	1.421	149	1.570
>	ClientContractPublishService.j	95,5 %	386	18	404
>	ClientContractUpdateService,	94,4 %	357	21	378
>	ClientContractCreateService.j	93,9 %	276	18	294
>	ClientContractDeleteService.j	63,6 %	145	83	228
>	ClientContractShowService.ja	97,0 %	131	4	135
>	ClientContractListService.java	94,8 %	91	5	96
>	ClientContractController.java	100,0 %	35	0	35
▼	acme.features.client.progressLog	92,1 %	1.358	117	1.475
>	ClientProgressLogCreateServi	95,4 %	331	16	347
>	ClientProgressLogPublishServ	94,9 %	317	17	334
>	ClientProgressLogUpdateServ	94,6 %	313	18	331
>	ClientProgressLogListService.j	95,6 %	153	7	160
>	ClientProgressLogDeleteServi	66,0 %	107	55	162
>	ClientProgressLogShowServic	96,2 %	102	4	106
>	ClientProgressLogController.j	100,0 %	35	0	35

Como se puede ver, se ha alcanzado un coverage casi completo, excepto por algunas restricciones excesivas/redundantes en los authorise, y el unbind en el delete (no se usa realmente pues el delete no llega a tener casos negativos donde tenga que recargar la página)

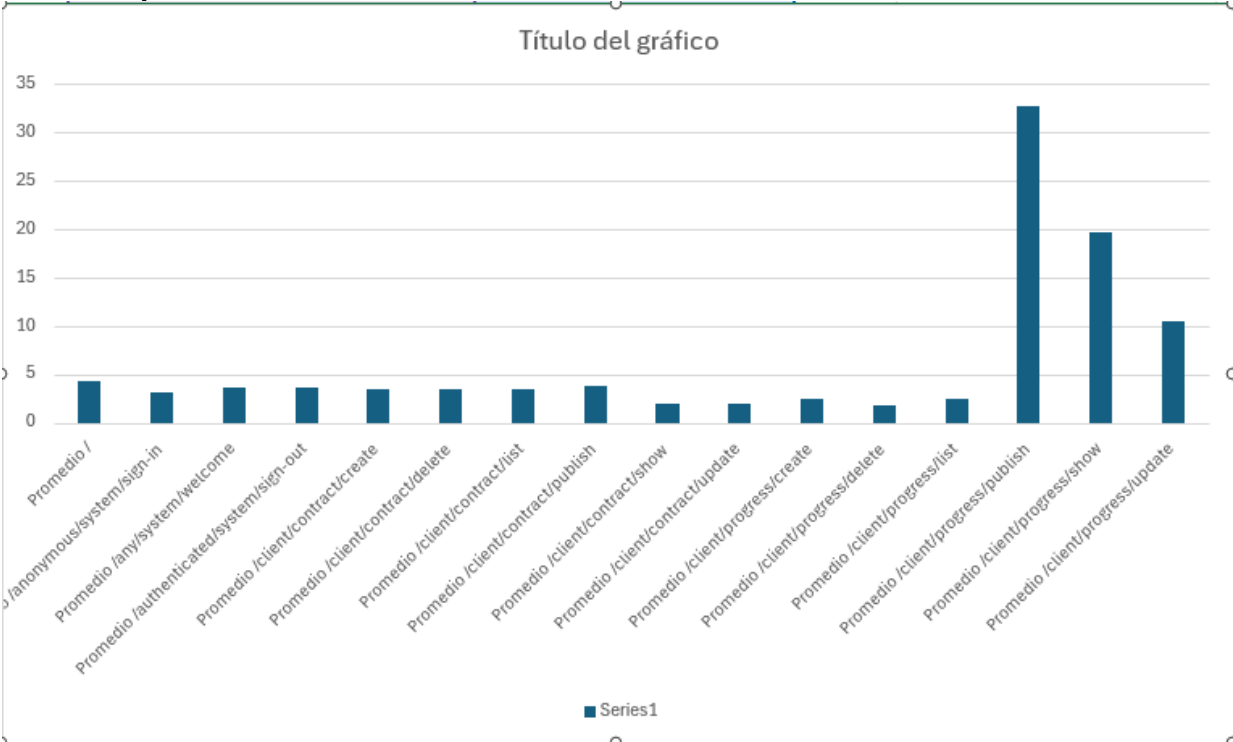
5.2 Pruebas de rendimiento

Pruebas previas a la refactorización con índices



Mean	11,71179741
Standard Error	0,4384592
Median	9,247638945
Mode	10,3409573
Standard Deviation	8,384928838
Sample Variance	80,48920894
Kurtosis	47,32085792
Skewness	7,4569328
Range	130,3186
Minimum	1,2715
Maximum	131,5901
Sum	4063,9937
Count	346
Confidence Level(95%)	0,630120985

Pruebas posteriores a la refactorización con índices



Mean	5,403115274
Standard Error	0,4384592
Median	4,7695071
Mode	4,349068329
Standard Deviation	6,4609834
Sample Variance	13,7098731
Kurtosis	143,1235794
Skewness	18,56892837
Range	127,7744
Minimum	1,1625
Maximum	128,9369
Sum	1874,881
Count	346
Confidence Level(95%)	0,07457232

Se puede comprobar visualizando ambas gráficas que las peticiones que más tiempo requieren son las de publicación, actualización y muestra de registros de progreso, seguramente en parte porque requieren acceder también a los valores de contratos, además de a los propios

También que, tras añadir los índices a contratos y registros de progreso, las peticiones toman generalmente bastante menos, indicando una mejoría en la velocidad de procesado de la página

Los intervalos de confianza con un nivel de confianza del 95% son de (7.18,9.14) previos a la refactorización, y de (1.7,1.9) posteriores, con lo cual con esa bajada de valor del parámetro podemos ver una mejora significativa en el rendimiento, y al haber una variabilidad tan grande de uno frente a otro, que el sistema es más consistente en su rendimiento.

Hipótesis de contraste

	Before	After
Media	11,71179741	10,48672935
Varianza (conocida)	80,48920894	13,7098731
Observaciones	348	348
Diferencia hipotética de las medias	0	
z	-1,129503465	
P(Z<=z) una cola	0,092785484	
Valor crítico de z (una cola)	1,578234612	
Valor crítico de z (dos colas)	0,345982347	
Valor crítico de z (dos colas)	1,984834849	

Con estos valores, tomados con un 95% de nivel de confianza, observamos que el valor de P se encuentra por encima de 0,05.

Por lo cual podemos concluir que a la hora de comparar, tras la refactorización de los índices hace el sistema más eficiente en cuanto al tiempo que provee las peticiones de manera individual, pero de manera global viene a surgir el mismo efecto.

6. Conclusiones

En el informe se recogen todas las pruebas realizadas por el estudiante 2, las cuales han servido para detectar algún error en el código y calcular el adecuado funcionamiento; y poder calcular el rendimiento, pudiendo sacar así conclusiones sobre la eficiencia de nuestro sistema a la hora de servir las peticiones.

