

TESTING REPORT



Santos Martín, Javier
Ruíz Garrido, Javier
Jiménez Morales, Francisco Miguel
García de Tejada Delgado, José
Antonio Daniel Porcar Aragón

General project information






PROJECT NAME			Acme-SF	
PARTICIPANTS				
Name	Email	Role	Username	Photo
Antonio Daniel Porcar Aragón	antporara@alum.us.es	Project Manager, Developer	antporara	
Francisco Miguel Jiménez Morales	frajimmor2@alum.us.es	Tester, Developer	frajimmor2	
Javier Santos Martín	javsanmar5@alum.us.es	Developer, Secretary	javsanmar5	
Javier Ruiz Garrido	javruigar2@alum.us.es	Analyst, Developer	Javiruizg	
José García de Tejada Delgado	josgardel8@alum.us.es	Operator, Developer	JoseGTD	
Stakeholders				
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).				
Start date	Expected date of completion	Deliverables	Date of Document	
12/02/2024	27/05/2024	4	14/02/2024	

Table of contents

REPORT NAME	1
General project information	2
Table of contents	3
Executive summary	4
Revision table	5
Introduction	6
Contents	7
Conclusions:	13
Bibliography	14

Executive summary

En este informe se detalla qué pruebas se han implementado, y cómo han ayudado al desarrollo del proyecto, tanto por mejora en la implementación del código, cómo en encontrar y resolver errores desconocidos.

Revision table

Primera revisión	20/05/2024
Última actualización	05/07/2024

Introduction

A lo largo de este informe veremos las pruebas que se han implementado para las 2 entidades del alumno 5. Se ha cubierto solamente la parte del testeo formal, puesto que por falta disponibilidad de tiempo para esta entrega, se ha probado lo que se ha podido y cómo se ha podido.

Las pruebas implementadas para casos positivos y negativos están guardadas como *acción-a-probar.safe*, mientras que los casos de hackeo/ciberataque están guardados como *acción-a-probar.hack*. Se han implementado un archivo *.hack* y otro *.safe* por las acciones listar, mostrar, crear, borrar, actualizar y publicar de las entidades CodeAudit y AuditRecord.

Tras sus correspondientes arreglos, el 100% de las pruebas se ejecutan con éxito obteniendo los resultados esperados.

Es importante destacar que, aunque el tiempo fue limitado, el proceso de prueba fue exhaustivo y se ha seguido la metodología explicada en clase para garantizar la integridad, la calidad y seguridad de las acciones implementadas.

Estas pruebas servirían en un proyecto real para comprobar que siguen funcionando correctamente si se implementaran nuevas entidades/funcionalidades que afecten a las ya implementadas.

Contents

En general los casos de prueba se han probado de la siguiente forma:

En los test nombrados como `.safe` donde se prueban casos positivos y negativos, hemos buscado abarcar que el sistema funciona correctamente para casos positivos, y que el sistema reacciona correctamente antes casos no válidos para ejecutar la acción a probar.

Los casos de prueba positivos se han centrado en probar ejecutar la acción bajo prueba en diferentes ámbitos en los que se pueda ejecutar correctamente; en los casos que requieran cambio de datos; variando los periodos de fechas, y probando diferentes combinaciones con cadenas más largas o más cortas para verificar que no existe ningún problema con el sistema.

Los casos de prueba negativos se han centrado en comprobar el funcionamiento de todas las validaciones que se estaba probando, comprobando que no solo se activan para valores límite sino que también para valores que sobrepasan por mucho los límites establecidos. Además se han comprobado casos en los que tendrían que saltar las restricciones establecidas por algún fallo del usuario; con casos como crear el objeto con un formulario completa o parcialmente vacío, introducir datos inválidos como caracteres en una fecha o patrones incorrectos.

En los test nombrados como `.hack`; donde se prueba la integridad y confidencialidad del sistema, hemos probado 3 casos para cada acción bajo pruebas. Primero se ha intentado mediante su URL correspondiente comprobar que es imposible ejecutar la acción a probar con un usuario anónimo, no logueado. Luego se ha vuelto a intentar con un usuario logueado pero de distinto rol, y por último con un usuario logueado del mismo rol.

Cabe destacar que esta última prueba no se ejecutó en ciertos casos donde un usuario del mismo rol tiene acceso a ejecutar dicha acción o bien por lógica (como crear una entidad), o bien por estar pensado el sistema para que esto no pueda hacerse; aquí específicamente es la acción de `list-mine`; que utilizando la URL de otro usuario del mismo rol, en vez de dar error 500; muestra las entidades del usuario que usa la URL “robada” y no las del usuario al que quería robar la información.

CODE AUDIT

Show `.safe` y `.hack`

Los casos de prueba `.safe` se han reproducido correctamente demostrando la correcta funcionalidad del sistema. En los casos `.hack`, gracias a estos se ha corregido una validación que permitía mostrar cualquier `codeAudit` a cualquier auditor.

List `.safe` y `.hack`

Los casos de prueba `.safe` y `.hack` se han reproducido correctamente demostrando la correcta funcionalidad del sistema.

Create .safe y .hack

Gracias a los casos de prueba .safe se ha encontrado y corregido un error en las validaciones de la fecha. Los casos de prueba .hack se han reproducido correctamente demostrando la correcta funcionalidad del sistema.

Update .safe y .hack

Gracias a los casos de prueba .safe se ha encontrado y corregido un error en las validaciones de la fecha. Los casos de prueba .hack se han reproducido correctamente demostrando la correcta funcionalidad del sistema.

Delete .safe y .hack

Los casos de prueba .safe y .hack se han reproducido correctamente demostrando la correcta funcionalidad del sistema.

Publish .safe y .hack

Los casos de prueba .safe y .hack se han reproducido correctamente demostrando la correcta funcionalidad del sistema.

Cabe destacar que el acto de publicar se considera únicamente como publicar el Code Audit en el estado en el que se encuentra al acceder al formulario de dicho Code Audit. Si se quisieran hacer cambios previos a la publicación, sería necesario actualizarlo y luego publicarlo.

AUDIT RECORD

Show .safe y .hack

Los casos de prueba .safe y .hack se han reproducido correctamente demostrando la correcta funcionalidad del sistema.

List .safe y .hack

Los casos de prueba .safe y .hack se han reproducido correctamente demostrando la correcta funcionalidad del sistema.

Create .safe y .hack

Gracias a los casos de prueba .safe se ha encontrado y corregido un error en las validaciones de la fecha. Los casos de prueba .hack se han reproducido correctamente demostrando la correcta funcionalidad del sistema.

Update .safe y .hack

Gracias a los casos de prueba .safe se ha encontrado y corregido un error en las validaciones de la fecha. Los casos de prueba .hack se han reproducido correctamente demostrando la correcta funcionalidad del sistema.

Delete .safe y .hack

Los casos de prueba .safe y .hack se han reproducido correctamente demostrando la correcta funcionalidad del sistema.

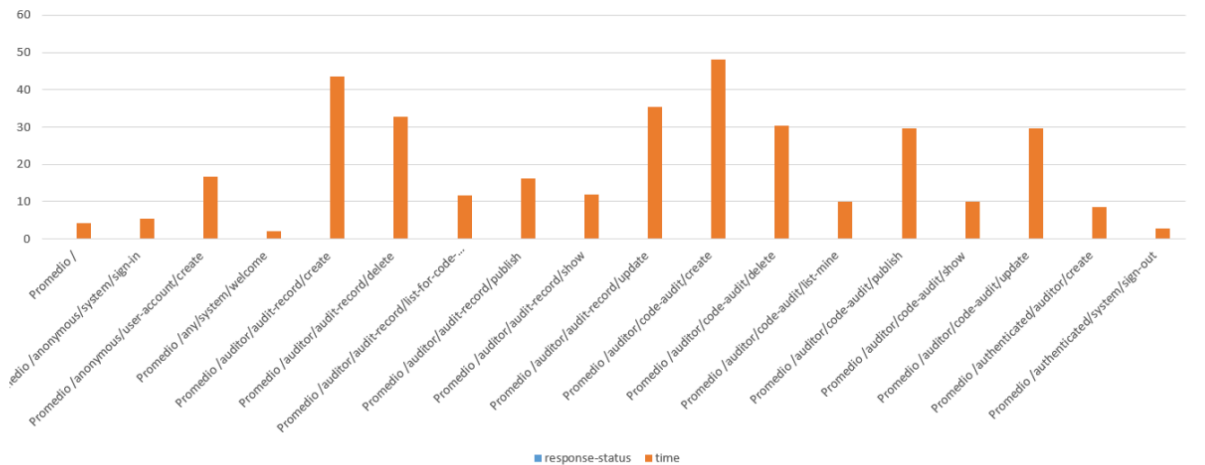
Publish .safe y .hack

Los casos de prueba .safe y .hack se han reproducido correctamente demostrando la correcta funcionalidad del sistema.

Cabe destacar que el acto de publicar se considera únicamente como publicar el Audit Record en el estado en el que se encuentra al acceder al formulario de dicho Audit Record. Si se quisieran hacer cambios previos a la publicación, sería necesario actualizarlo y luego publicarlo.

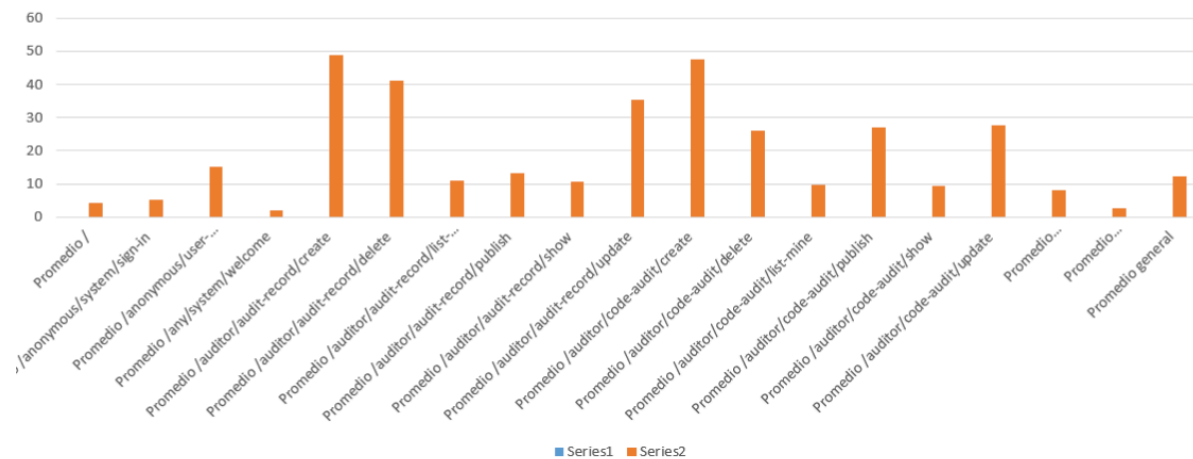
PERFORMANCE TESTING

Pruebas previas a la refactorización con índices



		<i>Columna1</i>
	Media	12,79864168
	Error típico	0,780803416
	Mediana	6,6804
	Moda	2,0399
	Desviación estándar	16,37827062
	Varianza de la muestra	268,2477485
	Curtosis	5,918392773
	Coefficiente de asimetría	2,317237844
	Rango	100,609
	Mínimo	1,1408
	Máximo	101,7498
	Suma	5631,402337
	Cuenta	440
	Nivel de confianza(95,0%)	1,534577343
Interval(ms)	11,26406433	14,33321902
Interval(s)	0,011264064	0,014333219

Pruebas posteriores a la refactorización con índices



Columna2				
			Interval(ms)	10,95119
			Interval(s)	0,01095119
Media	12,5063636			14,0615372
Error típico	0,7912879			
Mediana	6,3322			
Moda	#N/D			
Desviación es	16,617046			
Varianza de la	276,126217			
Curtosis	11,4266252			
Coefficiente de	2,81029554			
Rango	140,693			
Mínimo	1,1688			
Máximo	141,8618			
Suma	5515,30636			
Cuenta	441			
Nivel de confi	1,5551736			

En ambas gráficas podemos observar que las peticiones que demandan más tiempo de ejecución son las de creación, actualización y borrado. Podemos observar que la refactorización ha contribuido a la reducción de estos tiempos en pequeña medida.

Los intervalos de confianza con un nivel de confianza del 95% son de (11.2, 14.3) previos a la refactorización, y de (10, 14) posteriores, con lo cual con esa bajada de valor del parámetro podemos ver una mejora significativa en el rendimiento, igualmente la bajada no es demasiado grande por lo que no podemos asegurar que es una gran mejora para el rendimiento del sistema.

Hipótesis de contraste

Prueba z para medias de dos muestras		
	<i>Before</i>	<i>After</i>
Media	12,79800316	12,50636363
Varianza (conocida)	268,2477485	276,1262168
Observaciones	441	441
Diferencia hipotética de las medias	0	
z	0,262492563	
P(Z<=z) una cola	0,396470859	
Valor crítico de z (una cola)	1,644853627	
Valor crítico de z (dos colas)	0,792941718	
Valor crítico de z (dos colas)	1,959963985	

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

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. Igualmente este efecto no será demasiado grande debido a la poca reducción en ms del tiempo de ejecución.

Conclusions:

Podemos observar que la creación de estas pruebas no solo ha ayudado a la detección de errores, sino que también ayudará a comprobar en el futuro el correcto funcionamiento de las funcionalidades implementadas para un buen mantenimiento del proyecto, garantizar la calidad del software, seguridad, y reducción de costes a largo plazo; esto último ayudando a encontrar problemas antes de que sea muy costoso resolverlos.

Bibliography

Intentionally blank.