

UNIVERSIDAD D SEVILLA

E2 Testing report D04

Group: C1.031

Student #2

Name: Nuñez Sanchez Juan
Email: juanunsan2@alum.us.es

Repository: https://github.com/alvarous/Acme-One-C1.031

Contenido

Contenido
Resumen ejecutivo
Registro de cambios
Introducción
Testing Funcional
Test de Rendimiento
Conclusión
Bibliografía

Resumen ejecutivo

En este documento veremos todos los casos de prueba y nos daremos cuenta de que hemos encontrado algunos bugs en view de Contract, un bug en la implementación de la API y hemos detectado problemas en la base de datos debido a la insuficiencia de pruebas en la población de los datos que dificultaban la creación de estas pruebas.

Registro de cambios

Versión	Causa del Cambio	Responsable del Cambio	Fecha del Cambio
1.0	Testing Report	Juan Nuñez Sanchez	08/07/2024

Introducción

En este documento, se incluirán tanto los casos de prueba que hemos llevado a cabo como un análisis detallado de estos y una evaluación del rendimiento de nuestro proyecto en un ordenador de gama media.

Testing Funcional

Primero comentaremos las diferentes pruebas realizadas para realizar testing funcional para las entidades contract y progressLogs creadas por el estudiante:

CONTRACT.

Listar: Para verificar que la funcionalidad de listado funciona correctamente, iniciamos sesión como cliente y accedimos a la lista de contratos. Tras confirmar que los campos se muestran correctamente, dimos por concluida la prueba. Esta prueba no detectó ningún bug.

Mostrar: Para asegurarnos de que la función de visualización funciona adecuadamente, iniciamos sesión como cliente y accedimos a la lista de contratos para luego ingresar en uno de ellos. Tras verificar que los campos se muestran de manera correcta, dimos por finalizada la prueba. Esta prueba no detectó ningún bug.

Crear: Para comprobar que la creación de un contrato funciona correctamente, probamos diferentes posibilidades: intentamos crearlo vacío, crearlo con todos los campos válidos excepto uno vacío, y crear un contrato ya existente. También probamos cada campo de entrada uno por uno con diferentes valores no válidos. Detectamos un bug que impedía la creación de un contrato vacío y provocaba un error 500, el cual ha sido solucionado.

Eliminar: Para verificar que la eliminación funciona correctamente, accedimos a un contrato sin publicar y lo eliminamos. No se pudo probar la función de "unbind" ya que no es posible eliminar un contrato de un contrato ya publicado.

Actualizar: Para asegurar el correcto funcionamiento de la actualización, repetimos exactamente las mismas pruebas realizadas para la creación de un contrato. Estas pruebas no detectaron ningún bug.

Publicar: Para verificar la funcionalidad de publicación de contratos, repetimos las mismas pruebas que en la creación y actualización. Además, probamos cada campo de entrada uno por uno con diferentes valores no válidos.

~	# acme.features.client.contract	90,0 %	1.316	146	1.462
	> ClientContractUpdateService.	88,0 %	324	44	368
	> ClientContractCreateService.ja	87,0 %	287	43	330
	> 🗾 ClientContractPublishService.j	89,9 %	320	36	356
	> Z ClientContractDeleteService.j.	91,4 %	128	12	140
	> 🗾 ClientContractShowService.ja	95,7 %	156	7	163
	> 🗾 ClientContractListAllService.ja	94,3 %	66	4	70
	> 🗾 ClientContractController.java	100,0 %	35	0	35

Progress Logs.

Listar: Para verificar que la funcionalidad de listado funciona correctamente, iniciamos sesión como cliente y accedimos a la lista de contratos, entramos en uno de ellos y accedemos a sus progress logs. Tras confirmar que los campos se muestran correctamente, dimos por concluida la prueba. Esta prueba no detectó ningún bug.

Mostrar: Para asegurarnos de que la función de visualización funciona adecuadamente, iniciamos sesión como cliente y accedimos a la lista de contratos, entramos en uno de ellos y accedemos a sus progress logs y accedemos a uno de ellos. Tras verificar que los campos se muestran de manera correcta, dimos por finalizada la prueba. Esta prueba no detectó ningún bug.

Crear: Para comprobar que la creación de un progress log funciona correctamente, probamos diferentes posibilidades: intentamos crearlo vacío, crearlo con todos los campos válidos excepto uno vacío, y crear un contrato ya existente. También probamos cada campo de entrada uno por uno con diferentes valores no válidos. No se detectó ningún bug.

Eliminar: Para verificar que la eliminación de un progress log funciona correctamente, accedimos a un progress log no publicado y lo eliminamos. No se pudo probar la función de "unbind" ya que no es posible eliminar un progress log ya publicado.

Actualizar: Para asegurar el correcto funcionamiento de la actualización, repetimos exactamente las mismas pruebas realizadas para la creación de un progress log. Estas pruebas no detectaron ningún bug.

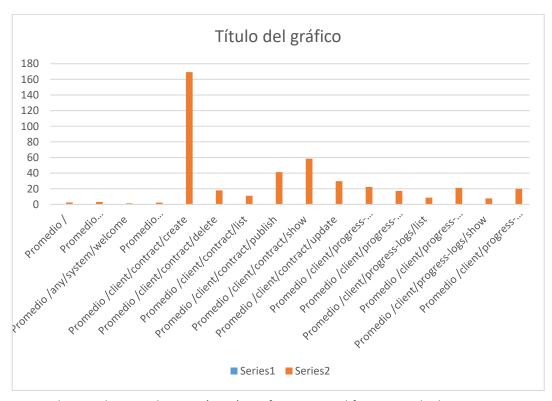
Publicar: Para verificar la funcionalidad de publicación de progress logs, repetimos las mismas pruebas que en la creación y actualización.

~	acme.features.client.progresslog ■	92,0 %	1.056	92	1.148
	> 🗾 ClientProgressLogsCreateServ	90,6 %	202	21	223
	> 🗾 ClientProgressLogsUpdateSer	91,3 %	211	20	231
	> 🗾 ClientProgressLogsDeleteSen 📙	86,8 %	112	17	129
	> 🗾 ClientProgressLogsPublishSer	93,5 %	246	17	263
	> 🗾 ClientProgressLogsListService	92,6 %	151	12	163
	> 🗾 ClientProgressLogsShowServi	95,2 %	99	5	104
	> 🗾 ClientProgressLogsController.	100,0 %	35	0	35

Test de Rendimiento.

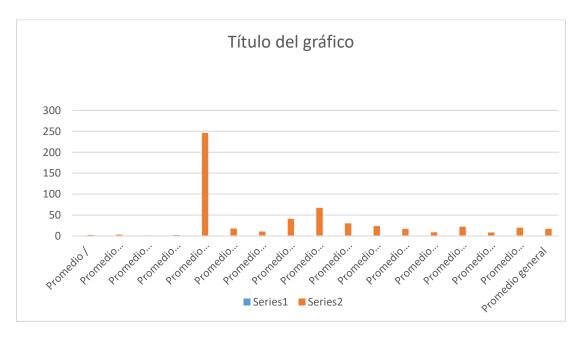
Para el test de rendimiento se han registrado los tiempos promedio que ha tomado la aplicación en responder a cada solicitud. Este gráfico presenta esos resultados.

Esta primera sin Índices:



Como podemos observar, la petición más ineficiente con diferencia es la de crear contratos. Se ha deducido que la causa de esto puede ser porque tiene que crear el objeto desde 0 y tiene que acceder a la lista de proyectos para asignárselo al contrato.

Estos son los resultados para el promedio con Índices:



before	after						
58,4247	62,0227						
5,4766	2,627999						
5,2778	2,986						
11,1362	2,1523						
2,8281	2,3347		befo	ore		aft	er
3,1856	1,6911						
32,4499	2,5687		Media	14,3236279		Media	16,6546589
18,8056	1,6203		Error típico	1,69162045		Error típico	2,7499296
121,8104	2,5652		Mediana	4,1326		Mediana	4,1366
237,1415	1,8451		Moda	1,3747		Moda	1,2187
240,7869	1,8482		Desviación e	34,7914967		Desviación e	56,5577025
229,4853	1,7383		Varianza de	1210,44824		Varianza de	3198,77371
236,4088	2,7484		Curtosis	24,4654151		Curtosis	143,606255
202,4136	2,0916		Coeficiente	4,84830107		Coeficiente (10,3558287
196,9459	1,3389		Rango	239,9334		Rango	896,7823
195,2708	1,6121		Mínimo	0,8535		Mínimo	0,7388
16,8141	1,2516		Máximo	240,7869		Máximo	897,5211
2,4228	1,5638		Suma	6058,8946		Suma	7044,9207
2,4293	1,4674		Cuenta	423		Cuenta	423
1,8578	2,2468		Nivel de con	3,32505147		Nivel de con	5,40526537
6,9199	1,7762						
2,5357	1,6279	Interval(ms)	10,9985764	17,6486794	Interval(ms)	11,2493935	22,0599242
1,8238	2,1413	Interval(s)	0,01099858	0,01764868	Interval(ms)	0,01124939	0,02205992
11,9904	1,1913						
171,862	1,5714						
44 4945	1 5007						

Analizando los intervalos, determinamos que el intervalo de confianza del 95% sin índices sería [10.99, 17.64] milisegundos, mientras que con índices será [11.24, 22.05] milisegundos. Al comprobar la correspondencia de milisegundos a segundos, se asegura que es un intervalo comprendido en menos de un segundo, lo cual era requerido para esta asignatura.

Prueba z para medias de dos m		
	before	after
Media	14,3236279	16,6546589
Varianza (conocida)	1210,44824	3198,77371
Observaciones	423	423
Diferencia hipotética de las me	0	
z	-0,72200008	
P(Z<=z) una cola	0,23514721	
Valor crítico de z (una cola)	1,64485363	
Valor crítico de z (dos colas)	0,47029443	
Valor crítico de z (dos colas)	1,95996398	

Podemos observar que el tiempo medio ha crecido considerablemente, lo que nos lleva a señalar que la introducción de los índices en este caso no ha favorecido el rendimiento, probablemente porque el algoritmo gasta más recursos buscando un índice determinado que en devolviendo los datos directamente.

Conclusión

En conclusión, aunque realizar pruebas puede ser una tarea tediosa y repetitiva, hacerlo de manera sistemática es fundamental para detectar errores y fallos en el proyecto. Además, el análisis de rendimiento nos permite ver claramente cómo la introducción de índices en la base de datos mejora el funcionamiento, aunque para haberlo notado más en este caso hubiera sido necesario que los índices tuvieran más peso en las llamadas. En general, estas actividades son muy beneficiosas y esenciales durante el desarrollo de un producto de software.

Intencionalmente en blanco.