

INFORME DE PRUEBAS



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



Diseño y Pruebas 2
Curso 2023-2024

Javier Ruiz Garrido

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	11
6. Conclusiones	14
7. Bibliografía	15

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 componente esencial en el desarrollo y mantenimiento de software, ya que permiten a los miembros del equipo de desarrollo verificar el correcto funcionamiento del sistema y detectar errores.

































En este documento se presenta un informe elaborado por el estudiante 3, que abarca los capítulos sobre pruebas funcionales y pruebas de rendimiento, además de un breve análisis sobre la cobertura de dichas pruebas.

3. Tabla de revisión

No aplica

4. Introducción

El contenido se dividirá en dos secciones principales: las pruebas funcionales y su cobertura, así como las pruebas de rendimiento. La primera sección se centrará en evaluar cómo cada función del software cumple con los requisitos especificados, asegurando que todas las funcionalidades operen correctamente. La segunda sección abordará las pruebas de rendimiento, donde se medirá la eficiencia y rapidez del sistema. Se adjunta foto de la cobertura de los test a los que han sido sometidos las entidades trainingModule y trainingSession.

Element	Covera...	Covered Ins...	Missed Instr...	Total Instruc...
▼  acme.features.developer.training_module	 94,7 %	1.461	81	1.542
>  DeveloperTrainingModulePublishService.java	 94,3 %	367	22	389
>  DeveloperTrainingModuleUpdateService.java	 93,9 %	310	20	330
>  DeveloperTrainingModuleDeleteService.java	 93,2 %	245	18	263
>  DeveloperTrainingModuleCreateService.java	 95,8 %	275	12	287
>  DeveloperTrainingModuleListMineService.java	 94,2 %	81	5	86
>  DeveloperTrainingModuleShowService.java	 97,4 %	147	4	151
>  DeveloperTrainingModuleController.java	 100,0 %	36	0	36
▼  acme.features.developer.training_session	 94,7 %	1.438	81	1.519
>  DeveloperTrainingSessionPublishService.java	 94,6 %	317	18	335
>  DeveloperTrainingSessionUpdateService.java	 94,6 %	313	18	331
>  DeveloperTrainingSessionCreateService.java	 95,2 %	318	16	334
>  DeveloperTrainingSessionDeleteService.java	 90,9 %	159	16	175
>  DeveloperTrainingSessionListMineService.java	 95,5 %	189	9	198
>  DeveloperTrainingSessionShowService.java	 96,4 %	106	4	110
>  DeveloperTrainingSessionController.java	 100,0 %	36	0	36

5. Contenidos

5.1 Pruebas funcionales

Training Module

- **Create.hack:** Sin estar logueado se prueba a cambiar la URL para crear un módulo de entrenamiento. Este intento de hackeo da un error 500 en la aplicación.
- **Create.safe:** Se ha logueado como developer1 y se crea un módulo de entrenamiento. Se prueban también todos los casos posibles y el sistema responde de forma esperada en todos aquellos. Se ha obtenido una cobertura del 95,8%.
- **Delete.hack:** Sin estar logueado se prueba a cambiar la URL para eliminar un módulo de entrenamiento. Registrándose como developer2, se intenta eliminar un módulo de entrenamiento perteneciente al developer1. Por último, logueado como developer1, se intenta eliminar un módulo de entrenamiento que ya está publicado. Todos estos intentos de hackeo dan como resultado un error de código 500 en la aplicación.
- **Delete.safe:** Se ha logueado como developer1 y se eliminan módulos de entrenamiento. Se ha obtenido una cobertura del 93,2%.
- **List-mine.hack:** Sin estar logueado se prueba a cambiar la URL para mostrar módulos de entrenamiento. Este intento de hackeo da como resultado un error de código 500 en la aplicación. Se ha logueado también como developer1 y developer2 para comprobar que se listan los módulos de entrenamiento adecuados.
- **List-mine.safe:** Se ha logueado como developer1 y developer2 para mostrar los listados de módulos de entrenamiento. Se ha obtenido una cobertura del 94,2%.

- **Publish.hack:** Sin estar logueado se prueba a cambiar la URL para publicar un módulo de entrenamiento. Registrándose como developer2, se intenta publicar un módulo de entrenamiento perteneciente al developer1. Por último, logueado como developer1, se intenta publicar un módulo de entrenamiento que ya está publicado. Todos estos intentos de hackeo dan como resultado un error de código 500 en la aplicación.
- **Publish.safe:** Se ha logueado como developer1 y se publica un módulo de entrenamiento. Se prueban también todos los casos posibles y el sistema responde de forma esperada en todos aquellos, incluidos los casos de intentar publicar un módulo de entrenamiento sin sesiones de entrenamiento asociadas o con sesiones de entrenamiento en modo borrador. Se ha obtenido una cobertura del 94,3%.
- **Show.hack:** Sin estar logueado se prueba a cambiar la URL para mostrar un módulo de entrenamiento. Registrándose como developer2, se intenta mostrar un módulo de entrenamiento perteneciente al developer1. Todos estos intentos de hackeo dan como resultado un error de código 500 en la aplicación.
- **Show.safe:** Se ha logueado como developer1 y se muestran varios módulos de entrenamiento, y repetimos el proceso logueados como developer2. Se ha obtenido una cobertura del 97,4%.
- **Update.hack:** Sin estar logueado se prueba a cambiar la URL para modificar un módulo de entrenamiento. Registrándose como developer2, se intenta modificar un módulo de entrenamiento perteneciente al developer1. Por último, logueado como developer1, se intenta modificar un módulo de entrenamiento que ya está publicado. Todos estos intentos de hackeo dan como resultado un error de código 500 en la aplicación.
- **Update.safe:** Se ha logueado como developer1 y se modifica un módulo de entrenamiento. Se prueban también todos los casos posibles y el sistema responde de forma esperada en todos aquellos. Se ha obtenido una cobertura del 93,9%.

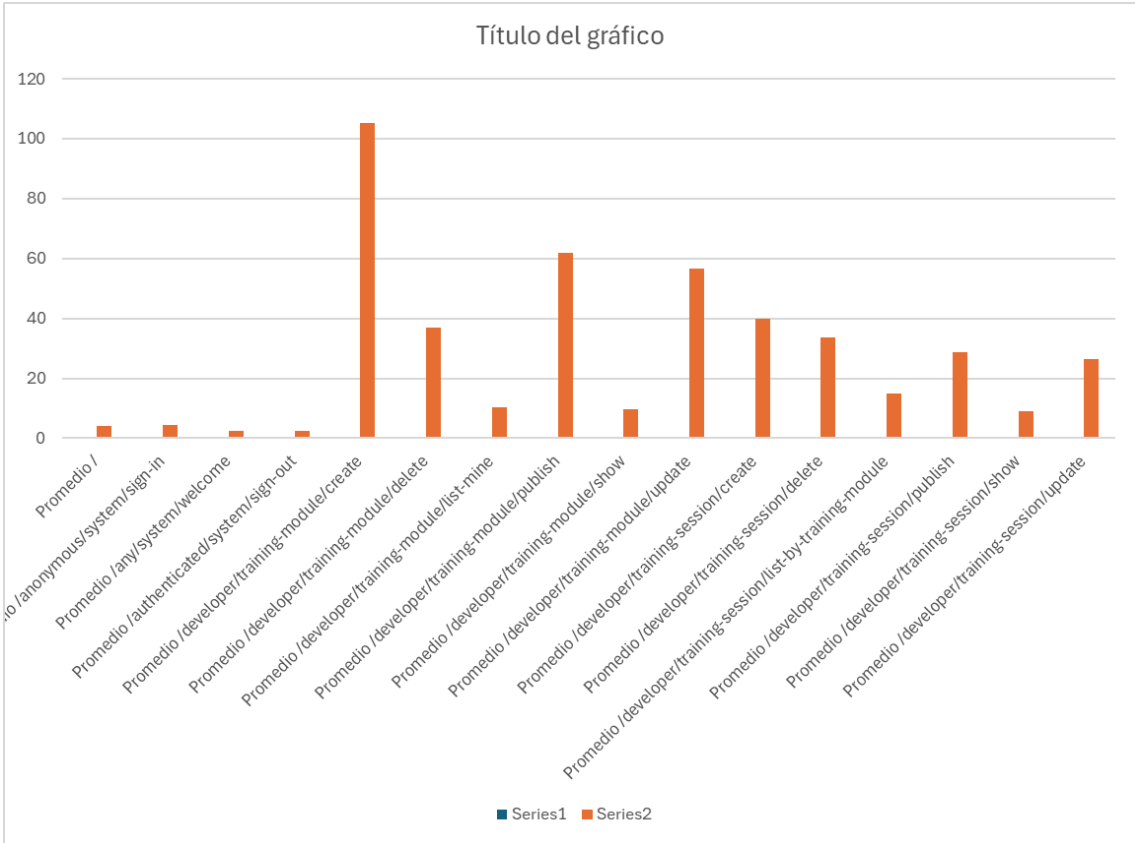
Training Session

- **Create.hack:** Sin estar logueado se prueba a cambiar la URL para crear una sesión de entrenamiento. Se loguea como developer2 y se intenta crear una sesión de entrenamiento para un módulo perteneciente al developer1. Por último se loguea como developer1 y se intenta crear una sesión de entrenamiento de un módulo ya publicado. Estos intentos de hackeo dan un error 500 en la aplicación.
- **Create.safe:** Se ha logueado como developer1 y se crea una sesión de entrenamiento. Se prueban también todos los casos posibles y el sistema responde de forma esperada en todos aquellos. Se ha obtenido una cobertura del 95,2%.
- **Delete.hack:** Sin estar logueado se prueba a cambiar la URL para eliminar una sesión de entrenamiento. Registrándose como developer2, se intenta eliminar una sesión de entrenamiento perteneciente al developer1. Por último, logueado como developer1, se intenta eliminar una sesión de entrenamiento que ya está publicada. Todos estos intentos de hackeo dan como resultado un error de código 500 en la aplicación.
- **Delete.safe:** Se ha logueado como developer1 y se eliminan sesiones de entrenamiento probando todos los casos posibles. Se obtienen los resultados esperados y un 90,9% de cobertura..
- **List-mine.hack:** Sin estar logueado se prueba a cambiar la URL para mostrar sesiones de entrenamiento. Este intento de hackeo da como resultado un error de código 500 en la aplicación. Se ha logueado también como developer2 y se ha intentado mostrar las sesiones de entrenamiento de un módulo perteneciente a developer1, resultando también con un error 500 en la aplicación.
- **List-mine.safe:** Se ha logueado como developer1 y developer2 para mostrar los listados de sesiones de entrenamiento. Se ha obtenido una cobertura del 95,5%.

- **Publish.hack:** Sin estar logueado se prueba a cambiar la URL para publicar una sesión de entrenamiento. Registrándose como developer2, se intenta publicar una sesión de entrenamiento perteneciente al developer1. Por último, logueado como developer1, se intenta publicar una sesión de entrenamiento que ya está publicada. Todos estos intentos de hackeo dan como resultado un error de código 500 en la aplicación.
- **Publish.safe:** Se ha logueado como developer1 y se publica una sesión de entrenamiento. Se prueban también todos los casos posibles y el sistema responde de forma esperada en todos aquellos. Se ha obtenido una cobertura del 94,6%.
- **Show.hack:** Sin estar logueado se prueba a cambiar la URL para mostrar una sesión de entrenamiento. Registrándose como developer2, se intenta mostrar una sesión de entrenamiento perteneciente a un módulo del developer1. Todos estos intentos de hackeo dan como resultado un error de código 500 en la aplicación.
- **Show.safe:** Se ha logueado como developer1 y se muestran varias sesiones de entrenamiento, y repetimos el proceso logueados como developer2. Se ha obtenido una cobertura del 96,4%.
- **Update.hack:** Sin estar logueado se prueba a cambiar la URL para actualizar una sesión de entrenamiento. Registrándose como developer2, se intenta actualizar una sesión de entrenamiento perteneciente a un módulo del developer1. Por último, logueado como developer1, se intenta actualizar una sesión de entrenamiento que ya está publicada. Todos estos intentos de hackeo dan como resultado un error de código 500 en la aplicación.
- **Update.safe:** Se ha logueado como developer1 y se modifica una sesión de entrenamiento. Se prueban también todos los casos posibles y el sistema responde de forma esperada en todos aquellos. Se ha obtenido una cobertura del 94,6%.

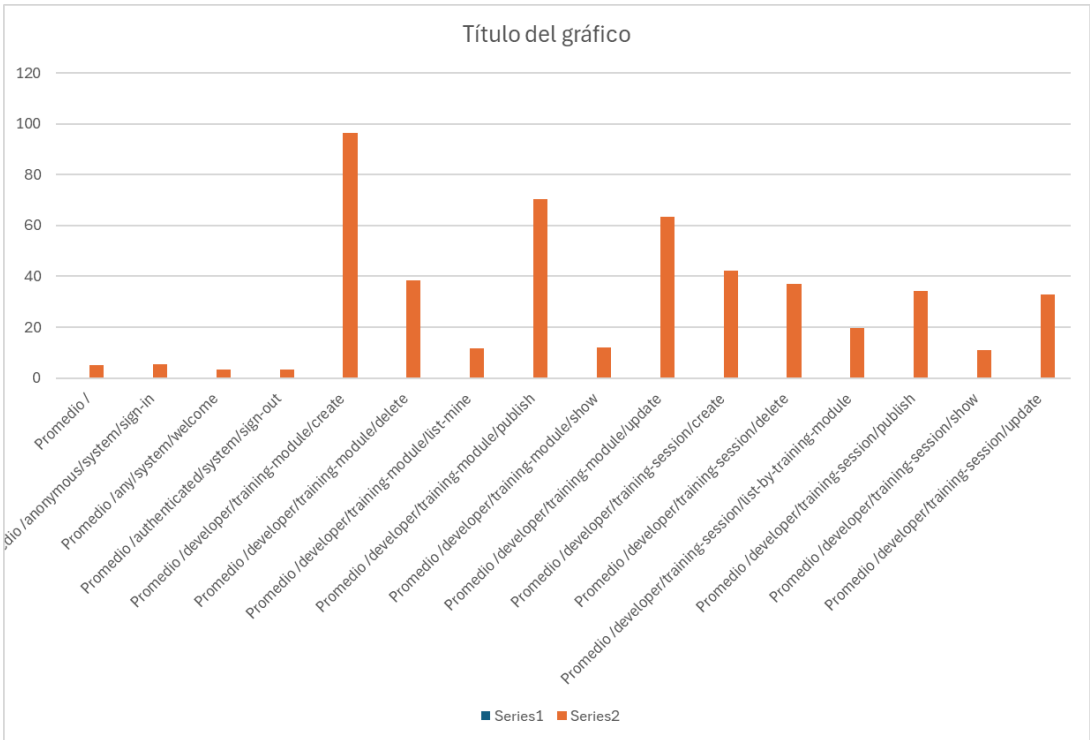
5.2 Pruebas de rendimiento

Previas a la refactorización con índices



Columna1					
Media	20,90819		Interval (ms)	18,57256	23,243821
Error típico	1,1891529		Interval(s)	0,0185726	0,0232438
Mediana	7,6601				
Moda	#N/D				
Desviación estándar	28,49008				
Varianza de la muestra	811,68464				
Curtosis	7,5471867				
Coeficiente de asimetría	2,4530771				
Rango	198,1868				
Mínimo	1,3593				
Máximo	199,5461				
Suma	12001,301				
Cuenta	574				
Nivel de confianza(95,0%)	2,3356304				

Posteriores a la refactorización con índices



Columna1					
Media	23,1724939	Interval(ms)	20,8471767	25,4978112	
Error típico	1,18390218	Interval(s)	0,02084718	0,02549781	
Mediana	9,80345				
Moda	#N/D				
Desviación estándar	28,3642801				
Varianza de la muestra	804,532388				
Curtosis	4,25037494				
Coeficiente de asimetría	1,95026781				
Rango	177,6098				
Mínimo	1,8701				
Máximo	179,4799				
Suma	13301,0115				
Cuenta	574				
Nivel de confianza(95,0%)	2,32531729				

Como se puede apreciar, después de la refactorización del código, las tareas que previamente demandaban más tiempo del sistema han reducido su tiempo promedio. Por otro lado, aquellas tareas que inicialmente requerían menos tiempo ahora presentan un ligero incremento en su duración.

Los intervalos de confianza con un nivel del 95% son (18.55, 23.25) antes de la refactorización y (20.85, 25.5) después de la misma. Se puede observar que la amplitud del intervalo es similar en ambos casos, aunque el tiempo promedio es ligeramente mayor tras la refactorización.

Hipótesis de contraste

Prueba z para medias de dos muestras		
	<i>Before</i>	<i>After</i>
Media	21,0977711	23,1724939
Varianza (conocida)	809,556291	804,532388
Observaciones	591	574
Diferencia hipotética de las medias	0	
z	-1,246258	
P(Z<=z) una cola	0,10633484	
Valor crítico de z (una cola)	1,64485363	
Valor crítico de z (dos colas)	0,21266968	
Valor crítico de z (dos colas)	1,95996398	

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 esta razón, podemos concluir que la refactorización no ha sido exitosa, ya que pese a que los tiempos sean diferentes, en términos generales, pueden considerarse iguales.

6. Conclusiones

En el informe se han recogido todas las pruebas realizadas por el estudiante 3, con las cuales se asegura el correcto funcionamiento del sistema además de un análisis sobre el rendimiento del mismo. La refactorización con índices no ha dado resultados positivos además de que estos resultados carecían de valor ya que globalmente no suponían ninguna diferencia.

7. Bibliografía

Intencionalmente en blanco.