

Informe de Pruebas

Portada

Título del Informe: Informe de Pruebas para Acme-SF-D04

Fecha de Creación: 27/05/2024

Autor del Informe: Francisco Capote Garcia

Índice

1. Functional testing
 - a. Introducción
 - b. Plan de pruebas
 - c. Registros safe y hack
 - d. Features testing
 - e. Análisis de resultados
 - f. Recomendaciones
2. Performance Testing
 - a. Introducción
 - b. Performance setting setup
 - c. Resultados y análisis
 - d. Contrate de los resultados

Introducción

Descripción del Proyecto y Contexto: En este documento se mostrarán los resultados de las pruebas realizadas para la comprobación de los servicios implementados (Create, Update, Publish, Delete y List).

Objetivos del Informe de Pruebas: Comprobar que el código realizado en la anterior entrega realmente es útil, es decir, se ejecuta tarde o temprano en alguna petición y optimizar las mismas llamadas

Alcance de las Pruebas Realizadas: Para las pruebas se han utilizado numerosos casos, estos datos con ayuda del Excel proporcionado para realizarlo, se han realizado varios intentos tanto para casos positivos como negativos.

Metodología de Prueba Utilizada: Hemos utilizado el launcher recorder, para grabar los datos en sus formularios correspondientes y comprobar así, si nos devuelve el valor esperado de error o aceptación.

Plan de Pruebas

Objetivos de las Pruebas: El objetivo es comprobar que si un dato debía de ser rechazado el servidor nos devolviese un mensaje 500, o si algún dato debía de acotar unos valores que la web nos devolviese un mensaje de error describiendo brevemente el motivo

Estrategia de Pruebas:

Para comprobar un servicio debíamos realizar un recorder para grabar todos los datos introducidos.

Si queríamos probar datos positivos, el registro de llamadas se nombraba como *“Entidad-Servicio.safe”*.

Si por lo contrario queremos probar datos no válidos, es decir, casos negativos, el registro es llamado como *“Entidad-Servicio.hack”*.

Criterios de Entrada y Salida para las Pruebas: Se han utilizado diferentes criterios para los datos de entrada según su tipo

Para los String en casos positivos hemos utilizado valores en su rango mínimo y máximo de longitud, valores por el medio, cerca del final e inicial, para los casos negativos valores como *null*, caracteres raros, superar el máximo de longitud.

Para las fechas en casos positivos hemos utilizado valores en pasado, y si dependen de dos fechas y estas definen un intervalo, que estas se mantengan en distancias que superen las validaciones, para los casos negativos hemos utilizado fechas que superan la fecha actual, y fechas que no cumplan con las condiciones de los intervalos.

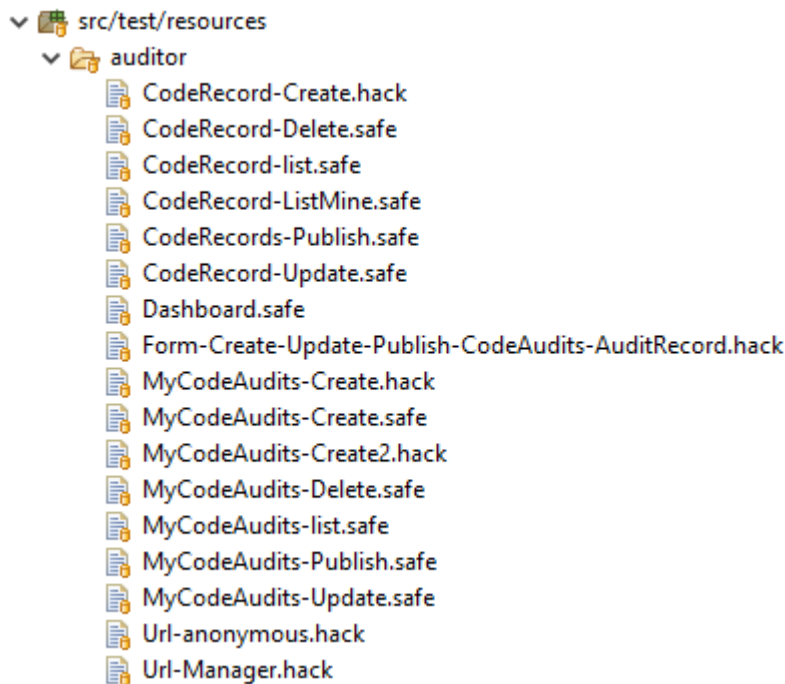
Para las Url en casos positivos hemos utilizado valores en su rango mínimo y máximo de longitud, valores por el medio, cerca del final e inicial, para los casos negativos valores como *null*, caracteres raros, superar el máximo de longitud.

Para otros tipos de dato como el code que tienen estos formatos "[A-Z]{1,3}-[0-9]{3}" hemos utilizado diversos valores para comprobar también en los que se cumple el formato y en otros que no.

Registros safe y hack

Descripción de los registros:

Acontinuacion se puede observar las trazas creadas apartir de los recorder.



Como se explicó anteriormente se crearon los archivos .safe y .hack para comprobar las pruebas de sus correspondientes entidades y features.

Nota: Debido a que mi ordenador tardaba mucho tiempo en ejecutar cada recorder y replay tome la decisión de realizar todas las pruebas de .hack de Create Update Publish y delete en un mismo record 😊

Features testing



1. Feature 1: Code Audit

a. Test Case 1.1: Create

- i. Descripción: Al rellenar el formulario con datos validos funcionaba correctamente y las validaciones para denegar la entrada de datos no permitidos funciono correctamente.

- ii. Coverage: High – La única parte ejecutada parcialmente es comprobar es un condicional

```
super.state(existing == null || existing.equals(object),
```

 AuditorCodeAuditsCreateService.java  92,8 %



b. Test Case 1.2: Update

- i. Descripción: Al rellenar el formulario con datos validos funiconaba correctamente y las validaciones

para denegar la entrada de tados no permitidos funciono correctamente.

- ii. Coverage: Alta – solo dos lineas ejecutas parcialmente.



```
status = auditor == codeAudit.getAuditor().getId() && codeAudit.isPublished() == false;
if (!super.getBuffer().getErrors().hasErrors("published"))
    super.state(object.isPublished() == false, "published", "validation.codeaudit.published");
```

 AuditorCodeAuditsUpdateService.java  92,6 %

c. Test Case 1.3: Delete

- i. Descripcion: Se puede eliminar satisfactoriamente un codeAudit no publicado.
- ii. Coverage: Medio - Debido a que el apartado de la funcion unbind que no es ejecutado y las mismas dos lineas que en el update esto debido a que un codeAudit publicado no existe el boton de publicar,deletear o updatear.



```
marks = this.repository.findMarksByCodeAuditId(object.getId()).stream().toList();
Collection<Project> allProjects = this.repository.findAllProjects();
choices = SelectChoices.from(Type.class, object.getType());
choices2 = SelectChoices.from(allProjects, "code", (Project) allProjects.toArray()[0]);
dataset = super.unbind(object, "code", "executionDate", "type", "proposedCorrectiveActions", "optionalLink", "project", "published");
dataset.put("mark", this.repository.averageMark(marks));
dataset.put("type", choices);
dataset.put("projects", choices2);
super.getResponse().addData(dataset);
```

 AuditorCodeAuditsDeleteService.java  55,2 %

d. Test Case 1.4: Publish



- i. Descripcion: Se puede publicar codeAudits que cumplen con los campos válidos, es rechazado en caso contrario, y una vez publicado ya no puede ser editado ni borrado.
- ii. Coverage: Medio - Debido a la función unbind y a que no se ha probado todas las posibles medias del atributo Mark (6 de 8).

```
if (!super.getBuffer().getErrors().hasErrors("mark")) {
    List<Mark> marks;
    marks = this.repository.findMarksByCodeAuditId(object.getId()).stream().toList();
    Mark nota = this.repository.averageMark(marks);
    super.state(nota == Mark.C || nota == Mark.B || nota == Mark.A || nota == Mark.A_PLUS, "mark", "auditor.codeaudit.form.error.mark");
```

 AuditorCodeAuditsPublishService.java  56,6 %

e. Test Case 1.5: List



- i. Descripción: Se listan sin problemas todos los codeAudits.
- ii. Coverage: Alta - Prácticamente todo el código es ejecutado excepto los assert.

 AuditorCodeAuditsListPublishedService  95,5 %

f. Test Case 1.6: Show

- i. Descripción: Ver los detalles de un codeAudit funciona correctamente.

- ii. Coverage: Alta – No se detectaron bugs.



 AuditorCodeAuditsShowService.java  96,4 %

2. Feature 2: Code Record

a. Test Case 2.1: Create

- i. Descripción: Al rellenar el formulario con datos validos funcionaba correctamente y las validaciones para denegar la entrada de datos no permitidos funciono correctamente.

- ii. Coverage: High – No se detectaron bugs.



 AuditorAuditRecordCreateService.java  92,6 %

b. Test Case 2.2: Update

- i. Descripción: Funciona correctamente, aunque probando el intervalo en la validación de fechas no se cumplía la condición de que una tenía que ser antes de otra por lo que fue añadido lo siguiente.

```
if (!super.getBuffer().getErrors().hasErrors("auditPeriodStart"))  
    super.state(MomentHelper.isAfter(object.getAuditPeriodEnd(), object.getAuditPeriodStart()))
```



- ii. Coverage: Alta – Se detecto un bug en las validaciones y fue corregido

 AuditorAuditRecordUpdateService.java  93,6 %

c. Test Case 2.3: Delete

- i. Descripcion: Se puede eliminar satisfactoriamente un codeRecord no publicado.



- ii. Coverage: Medio - Debido a que el apartado de la funcion unbind que no es ejecutado y las mismas dos lineas que en el update esto debido a que un codeAudit publicado no existe el boton de publicar,deletear o updatear.

 AuditorAuditRecordDeleteService.java  57,0 %

d. Test Case 2.4: Publish

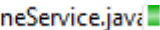
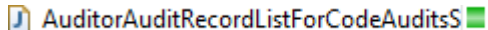
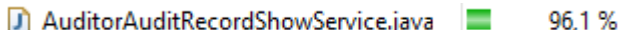
- i. Descripción: Se puede publicar codeRecord que cumplen con los campos válidos, es rechazado en caso contrario, y una vez publicado ya no puede ser editado ni borrado.

- ii. Coverage: Medio - Debido a la función unbind.

 AuditorAuditRecordPublishService.java  50,0 %

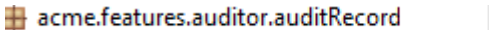
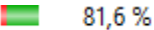
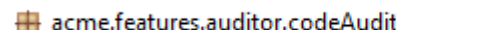
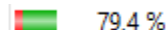
e. Test Case 2.5: List

- i. Descripcion: Se listan perfectamente todos los auditRecords.

- ii. Coverage: Alta- la clase se ejecuta entera menos los asserts.  93,9 %
- f. Test Case 2.6: ListCodeAudit
 - i. Descripción: La clase lista perfectamente los auditRecord de su codeAudit.
 - ii. Coverage: Alta - No se encontraron bugs.
 -  93,1 %
- g. Test Case 2.7: Show
 - i. Descripción: Se muestras satisfactoriamente los detalles de un auditRecord.
 - ii. Coverage: Alta- No se encontraron bugs.
 -  96,1 %

Análisis de Resultados

Análisis de los Resultados Obtenidos: Los resultados son bastante buenos ya que el 80% está probado para todas las features de ambas entidades.

| | |
|---|--|
|  acme.features.auditor.auditRecord |  81,6 % |
|  acme.features.auditor.codeAudit |  79,4 % |

No se encontraron problemas de seguridad con técnicas de hacking, post hacking, valores no permitidos...

Recomendaciones

Recomendaciones para resolver posibles defectos:

Es cierto que ampliando el número de peticiones con más combinaciones de valores excepcionales podrían encontrarse algún tipo de fallo o bug, pero con las pruebas aportadas anteriormente, no se han encontrado más errores en el código y las fallas que se encontraron fueron solucionadas.

Performance Testing

Introducción

Los resultados de las pruebas de rendimiento, incluidos gráficos y análisis estadísticos. Se proporciona un intervalo de confianza del 95% para el tiempo que tarda el proyecto en atender las solicitudes para dos ordenadores diferentes. Además, un contraste de hipótesis con un 95% de confianza determina cuál es el ordenador más poderoso.

Performance Testing Setup

Ordenador 1:

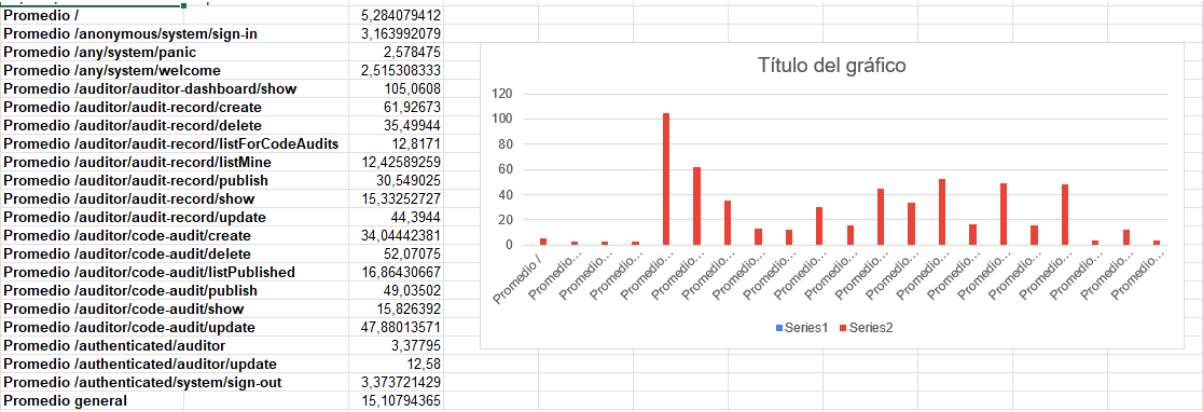
CPU: Intel Core i5-8400

RAM: 8GB

OS: Windows 10

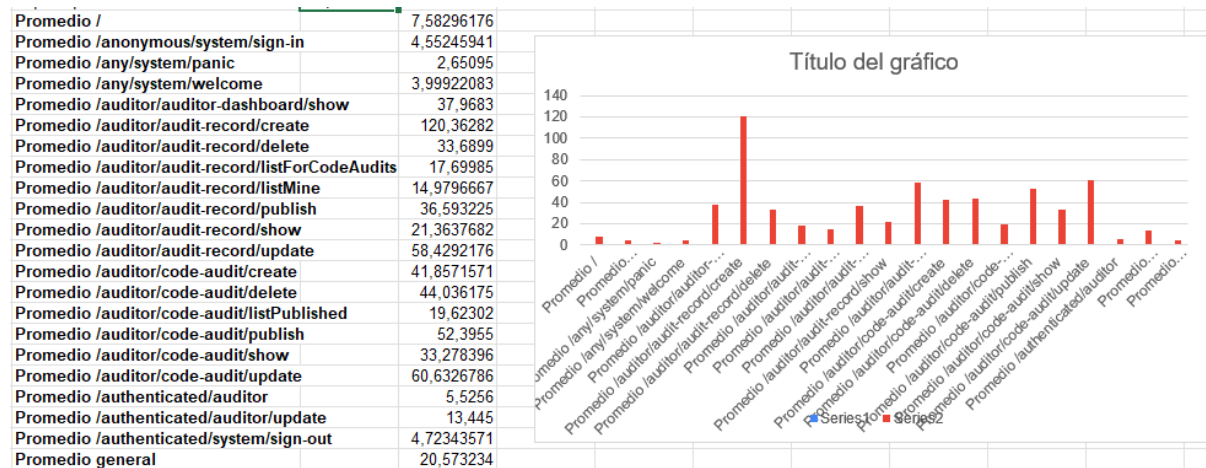
Resultados y analisis

Tenemos en primer lugar el resultado del ordenador 1 tras realizar las llamadas sin utilizar los índices en las entidades correspondientes, tal y como se explica en las diapositivas:



Este seria a continuación el resultado del ordenador 1 tras utilizar los índices explicados en las diapositivas de clase para mejorar el

rendimiento de las llamadas:



Aquí se adjunta unas imágenes para comparar las medias, error típico, mediana...

| Before | | | | | | After | | | |
|---------------------------|------------|------------|--|--|--|---------------------------|------------|------------|--|
| Media | 15,1079437 | | | | | Media | 20,573234 | | |
| Error típico | 1,10521028 | | | | | Error típico | 1,99402999 | | |
| Mediana | 6,6449 | | | | | Mediana | 8,9225 | | |
| Moda | #N/D | | | | | Moda | #N/D | | |
| Desviación estándar | 21,9377977 | | | | | Desviación estándar | 39,5803651 | | |
| Varianza de la muestra | 481,266969 | | | | | Varianza de la muestra | 1566,6053 | | |
| Curtosis | 8,27991126 | | | | | Curtosis | 90,5583613 | | |
| Coefficiente de asimetría | 2,7158757 | | | | | Coefficiente de asimetría | 7,77723825 | | |
| Rango | 150,1808 | | | | | Rango | 552,2826 | | |
| Mínimo | 1,4591 | | | | | Mínimo | 1,4418 | | |
| Máximo | 151,6399 | | | | | Máximo | 553,7244 | | |
| Suma | 5952,5298 | | | | | Suma | 8105,8542 | | |
| Cuenta | 394 | | | | | Cuenta | 394 | | |
| Nivel de confianza(95,0%) | 2,17286399 | | | | | Nivel de confianza(95,0%) | 3,92030007 | | |
| Interval(ms) | 17,2808076 | 12,9350797 | | | | Interval(ms) | 24,4935341 | 16,6529339 | |
| Interval(s) | 0,01728081 | 0,01293508 | | | | Interval(s) | 0,02449353 | 0,01665293 | |

| | before | after |
|---------------------------------|-------------|-----------|
| Media | 15,1079437 | 20,573234 |
| Varianza (conocida) | 481,266969 | 1566,6053 |
| Observaciones | 394 | 394 |
| Diferencia hipotética de las me | 0 | |
| z | -2,39723118 | |
| P(Z<=z) una cola | 0,00825975 | |
| Valor crítico de z (una cola) | 1,64485363 | |
| Valor crítico de z (dos colas) | 0,0165195 | |
| Valor crítico de z (dos colas) | 1,95996398 | |

Contraste de los resultados

Dado el valor critico de z (dos colas) = 0,0165 < 1, según la teoría vista en clase las pruebas han mejorado respecto a los realizados anteriormente sin los índices.