Testing Report

C2.017

https://github.com/vicgrabru/Acme-SF-D04

Carlos García Ortiz
cargarort3@alum.us.es

Tabla de contenidos

Resumen ejecutivo

Las pruebas de un sistema software son esenciales para garantizar el funcionamiento deseado del producto, permitiendo gran variedad de beneficios como el descubrimiento temprano de bugs, el análisis y comprensión del código, y muchas otras ventajas.

Sin embargo, un testing de calidad implica un procedimiento riguroso comprobando hasta el último recorrido del código a probar mediante un alto coverage, pues sino es así es posible que queden escondidos algunos fallos entre las sombras del sistema.

Tabla de revisiones

Revisión	Fecha	Descripción
1.0	27-06-2024	Primera versión del documento
2.0	02-07-2024	Versión final del documento

Introducción

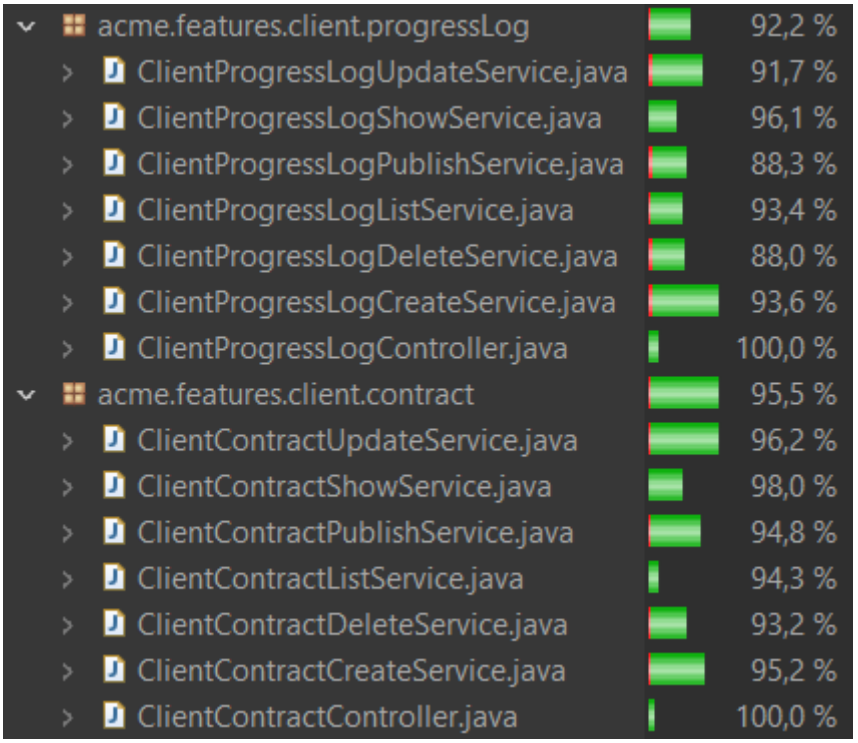
En este documento se ofrece una visión detallada del proceso de elaboración del conjunto de pruebas creadas por el estudiante 2 para las diferentes funcionalidades y características que ofrece el sistema al rol cliente respecto a las entidades Contract y Progress Log.

Además se incluye un análisis de las pruebas de rendimiento en diferentes ordenadores para comprobar el desempeño del sistema en dichas pruebas.

Contenido

Pruebas funcionales

En primer lugar, se muestra la cobertura que se obtiene al reproducir todas las pruebas.



Se observa que todos los servicios que se han probado superan el 85% sin problemas, consiguiendo mediante unas pruebas rigurosas obtener casi todas las líneas de los servicios con coverage verde. Las instrucciones marcadas en amarillo por el coverage son los siguientes casos:

- **assert object != null**: Estas instrucciones son una medida preventiva para no ejecutar instrucciones que accedan a propiedades de un objeto nulo, lo que lanzaría una excepción y llevaría a mostrar una vista de pánico.

```
assert object != null;
```

- **Rama inaccesible de condicional**: Sucede en el servicio de publicación de Contract, en el apartado de validación. Aquí se comprueba que el presupuesto del contrato a publicar no supere el coste restante de su Proyecto, es necesario comprobar que no se han detectado errores fatales en el campo del presupuesto para poder manejar el atributo en la comprobación y evitar una vista de pánico. Dicha línea amarilla es la comprobación mencionada, debido a que el servicio de publicación toma la entidad de la base de datos sin poder modificarla no es posible probar que haya error en su atributo budget, haciendo que siempre se entre en el condicional. Se podría eliminar dicho condicional pero se ha mantenido por seguridad.

```
if (!super.getBuffer().getErrors().hasErrors("budget"))
    super.state(remainingCost >= this.exchangeRepo.exchangeMoney(object.getBudget()).getAmount(), "budget",
```

Sucede algo similar en una validación del servicio de actualización de update, en un condicional se mira si el presupuesto y el proyecto no tienen errores ya que se compara el presupuesto con el coste del proyecto. Sin embargo, el proyecto no puede modificarse en la actualización así que no es posible recorrer la rama del condicional en la cual haya errores en el campo proyecto. Se podría eliminar la parte del condicional que comprueba que no haya errores en el campo proyecto pero se ha mantenido por seguridad.

```
if (!super.getBuffer().getErrors().hasErrors("budget") && !super.getBuffer().getErrors().hasErrors("project")) {
    super.state(this.exchangeRepo.exchangeMoney(object.getBudget()).getAmount() <= this.exchangeRepo.exchangeMoney(object.getProject()
    super.getBuffer().addGlobal("showExchange", true);
}
```

Casos de pruebas positivos, negativos y hackeos para Contract

List.safe

Con client1 accedemos a su lista de contratos y observamos que todo está bien. Puntualizar que para que se haga todo el coverage se observa una lista que tenga contratos publicados y no publicados ya que muestran diferentes símbolos.

List.hack

Sin iniciar sesión entramos en *localhost:8082/Acme-SF-D04/client/contract/list* para ver que no estamos autorizados.

Show.safe

Con client1 visualizamos el contrato A-000 (id 543) y luego el T-000 (id 580), publicado y no publicado respectivamente. Esto es debido a que si está en modo borrador debe mostrar un campo adicional indicando cuánto dinero del coste del contrato queda disponible.

Show.hack

Sin iniciar sesión y con client2, entramos en *localhost:8082/Acme-SF-D04/client/contract/show?id=543* para ver que no estamos autorizados **(por ser contrato de client1)**.

Sin iniciar sesión y con client2, entramos en la *localhost:8082/Acme-SF-D04/client/contract/show?id=2000* para ver que no estamos autorizados **(por ser contrato inexistente)**.

Delete.safe

Con client1 borramos el contrato T-000 (id 580), el cual por supuesto se encuentra en modo borrador.

Delete.hack

Sin iniciar sesión entramos en *localhost:8082/Acme-SF-D04/client/contract/delete* para ver que no estamos autorizados.

Con client2, visualizamos cualquier contrato en borrador e intentamos eliminarlo modificando la id a 580 con inspeccionar elemento, esperando no estar autorizados **(por ser contrato de client1 sin publicar)**. Se replica con el contrato de id 543 **(de client1 publicado)** y con id 2000 **(por contrato inexistente)**.

Finalmente con client1, visualizamos cualquier contrato en borrador e intentamos eliminar modificando la id a 543 con inspeccionar elemento esperando no estar autorizados **(por contrato de client1 publicado)**.

Publish.safe

Con client1 intentamos publicar el contrato T-000 (id 580), esto falla ya que el presupuesto de este contrato hará que se supere el coste restante del proyecto.

Tras esto intentamos publicar el contrato T-001 (id 581), funcionará ya que no se supera el coste.

Finalmente visualizamos e intentamos publicar el contrato T-002 (id 582), esto funciona ya que se llegará justo al límite del coste restante del proyecto sin superarlo.

Obviamente todos estos contratos están en borrador.

Publish.hack

Sin iniciar sesión entramos en *localhost:8082/Acme-SF-D04/client/contract/publish* para ver que no estamos autorizados.

Con client2, visualizamos cualquier contrato en borrador e intentamos publicarlo modificando la id a 580 con inspeccionar elemento, esperando no estar autorizados **(por ser contrato de client1 sin publicar)**. Se replica con el contrato de id 543 **(de client1 publicado)** y con id 2000 **(por contrato inexistente)**.

Finalmente con client1, visualizamos cualquier contrato en borrador e intentamos publicar modificando la id a 543 con inspeccionar elemento esperando no estar autorizados **(por contrato de client1 publicado)**.

Create.safe

Con client1 intentamos crear un contrato sin datos para que nos de fallo en los campos no opcionales. También hay que probar un caso con un budget correcto y un proyecto vacío para obtener cierta rama de coverage.

Luego rellenamos los campos con valores correctos y vamos atributo por atributo comprobando sus casos negativos:

- **code:** Sin respetar patrón (**AAAA-0000**), código repetido (**A-000**).

- **goals/providerName/customerName:** Por encima del tamaño máximo (***texto de tamaño 101/76/76***), con spam (**sex sex sex**).

- **budget:** Sin respetar formato de Money (**EUR 1.234**), por debajo del valor mínimo (**EUR -0.01**), por encima del valor máximo de su proyecto (**EUR 27.00**, ya que el proyecto seleccionado tiene EUR 26.99), por encima del valor máximo para Money (**EUR 10,000,000,000**), divisa no aceptada por el sistema (**JPY 5**).

Una vez comprobados los casos negativos creamos un contrato con datos válidos intermedios: **(FG-456, Goals, EUR 5, Provider, Customer, Test title (proyecto id 279))**

Ahora crearemos múltiples contratos en los que se comprueben todos estos datos de casos positivos:

- **code:** Varios casos: **Z-000, AA-567, AAA-999, ZZZ-998, ZZA-001, ZA-002, BYB-090**.

- **goals/providerName/customerName:** Mínimo tamaño (**G/P/C**), mínimo tamaño + 1 (**Go/Pr/Cu**), máximo tamaño (***texto de tamaño 100/75/75***), mínimo tamaño (***texto de tamaño 99/74/74***), carácteres extranjeros (**العظمى واعتلاء** y 국민경제의 발전을).

- **budget:** Valor mínimo (**EUR 0**), valor mínimo + 1 (**EUR 0.01**), valor máximo respecto al proyecto (**EUR 26.99**, ya que el proyecto seleccionado tiene dicho límite), valor máximo respecto al proyecto - 1 (**EUR 26.98**), valor máximo (**EUR 9999999999.99** usando un proyecto con coste en el límite), valor máximo - 1 (**EUR 9999999999.98** usando un proyecto con coste en el límite), otra divisa aceptada por el sistema (**USD 0**).

Create.hack

Sin iniciar sesión entramos en *localhost:8082/Acme-SF-D04/client/contract/create* para ver que no estamos autorizados.

Con client1 intentamos crear un contrato alterando instantiationMoment mediante inspeccionar elemento. Luego visualizamos el contrato creado y comprobamos que ese campo no se ha modificado por el hack ya que no debe de poder ser modificable.

Con client1 intentamos crear un contrato alterando el id mediante inspeccionar elemento a 543 **(contrato ya existente)** para comprobar que no lo sobreescribe.

Con client1 intentamos crear contractos hackeando con inyección en los atributos goals, provider name y customer name (**<h1>!</h1>** y **' or 'A'='A**) para comprobar que toma dichos valores como simples string.

Update.safe

Con el cliente1 se comprueba exactamente lo mismo que en create.safe editando el contrato T-000 (id 580), aunque en este caso no hay que comprobar el atributo code porque no puede modificarse una vez asignado.

Para comprobar la edición de budget máximo absoluto actualizamos el contrato T-003 (id 583) pertenece a un proyecto con esa cantidad de coste.

Update.hack

Sin iniciar sesión entrar en *localhost:8082/Acme-SF-D04/client/contract/update* para ver que no estamos autorizados.

Con client2 intentamos actualizar un contrato alterando la id a 580 con inspeccionar elemento esperando no ser autorizados **(contrato sin publicar de cliente1)**. Repetimos para id 543 **(por contrato publicado de client1)** y para id 2000 **(por contrato inexistente)**.

Con client1 intentamos actualizar un contrato alterando la id a 543 con inspeccionar elemento esperando no ser autorizados **(por contrato publicado de client1)**.

Con client1 intentamos actualizar cualquier contrato en borrador alterando el code e instantiation moment mediante inspeccionar elemento. Luego visualizamos el contrato actualizado y comprobamos que esos campos no se han modificado por el hack.

Con client1 intentamos actualizar cualquier contracto hackeando con inyección en los atributos goals, provider name y customer name (**<h1>!</h1>** y **' or 'A'='A**) para comprobar que toma dichos valores como simples string.

Casos de pruebas positivos, negativos y hackeos para Progress Log

List.safe

Con client1 entramos en el listado de progress logs del contrato A-000 (id 543), el cual no está en modo borrador, y observamos que todo está bien. Puntualizar que para que se haga todo el coverage se observa una lista que tenga progress logs publicados y no publicados ya que muestran diferentes símbolos.

List.hack

Sin iniciar sesión y con client2, entramos en *localhost:8082/Acme-SF-D04/client/progress-log/list?masterId=543* para ver que no estamos autorizados **(por contrato de client1)**.

Sin iniciar sesión y con client2 entramos en *localhost:8082/Acme-SF-D04/client/progress-log/list?masterId=2000* para ver que no estamos autorizados **(por contrato inexistente)**.

Con client1 entramos en *localhost:8082/Acme-SF-D04/client/progress-log/list?masterId=580 para ver que no estamos autorizados* **(por contrato no publicado de client1).**

show.safe

Con client1 visualizamos el progress log PG-A-0000 (id 778) y luego el PG-T-0000 (id 802), no estando el primero en modo borrador y el segundo sí. Ambos pertenecientes al contrato A-000 (id 543).

show.hack

Sin iniciar sesión y con client2 entramos en *localhost:8082/Acme-SF-D04/client/progress-log/show?id=778* para ver que no estamos autorizados **(por progress log de contrato de client1)**.

Sin iniciar sesión y con client2 entramos en *localhost:8082/Acme-SF-D04/client/contract/show?id=2000* para ver que no estamos autorizados **(por progress log inexistente)**.

delete.safe

Con client1 eliminamos el progress log PG-T-0000 (id 802) perteneciente al contrato A-000 (id 543). Obviamente dicho progress log está en borrador y el contrato no.

delete.hack

Sin iniciar sesión entramos en *localhost:8082/Acme-SF-D04/client/progress-log/delete* para ver que no estamos autorizados.

Con client2 creamos un progress log en cualquier contrato y lo intentamos eliminar alterando la id a 802 con inspeccionar elemento para ver que no estamos autorizados **(por progress log sin publicar de contrato de client1)**. Repetir con id 778 **(por progress log publicado de contrato de client1)** e id 2000 **(contrato inexistente).**

Con client1 repetimos intentando eliminar un progress log alterando la id a 778 **(por progress log publicado de contrato de client1)**.

publish.safe

Con client1 publicamos el progress log PG-T-0000 (id 802) perteneciente al contrato A-000 (id 543). Obviamente dicho progress log está en borrador y el contrato no.

publish.hack

Sin iniciar sesión entramos en *localhost:8082/Acme-SF-D04/client/progress-log/publish* para ver que no estamos autorizados.

Con client2 creamos un progress log en cualquier contrato y lo intentamos publicar alterando la id a 802 con inspeccionar elemento para ver que no estamos autorizados **(por progress log sin publicar de contrato de client1)**. Repetir con id 778 **(por progress log publicado de contrato de client1)** e id 2000 **(contrato inexistente).**

Con client1 repetimos intentando publicar un progress log alterando la id a 778 **(por progress log publicado de contrato de client1)**.

create.safe

Con client1 vamos al listado de progress logs del contrato A-000 (id 543) e intentamos crear un progress log sin datos para que nos de fallo en los campos no opcionales.

Luego rellenamos los campos con valores correctos y vamos atributo por atributo comprobando sus casos negativos:

- **recordId**: Sin respetar patrón (**AA-AAA-00000**), código repetido (**PG-A-0000**).

- **completeness:** Por debajo del valor mínimo (**0.00**), por encima del valor máximo (**100.01**), con exceso de decimales (**50.555**).

- **comment/responsiblePerson:** Por encima del tamaño máximo (***texto de tamaño 101/76***), con spam (**sex sex sex**).

Una vez comprobados los casos negativos creamos un progess log con datos válidos intermedios: **(PG-FG-4567, 50.00, Comment, Responsible)**

Ahora crearemos múltiples progress logs comprobando todos estos casos positivos:

- **recordId:** Diferentes posibilidades: **PG-B-0000, PG-Z-0001, PG-AA-4444, PG-ZZ-7777, PG-ZY-9998, PG-AB-9999**.

- **completeness:** Valor mínimo (**0.01**), valor minimo + 1 (**0.02**), valor máximo (**100.00**), valor máximo - 1 (**99.99**).

- **comment/responsiblePerson:** Mínimo tamaño (**C/R**), mínimo tamaño + 1 (**Co/Re**), máximo tamaño (***texto de tamaño 100/75***), máximo tamaño - 1 (***texto de tamaño 99/74***), carácteres extranjero (**العظمى واعتلاء** y 국민경제의 발전을).

create.hack

Sin iniciar sesión y con client2, entramos en *localhost:8082/Acme-SF-D04/client/progress-log/create?masterId=543* para ver que no estamos autorizados **(por contrato publicado de client1)**. Repetimos para *localhost:8082/Acme-SF-D04/client/progress-log/create?masterId=580* **(por contrato no publicado de client1)** y *localhost:8082/Acme-SF-D04/client/progress-log/create?masterId=2000* **(por contrato inexistente)**.

Con client1, entramos en *localhost:8082/Acme-SF-D04/client/progress-log/create?masterId=580 para ver que no estamos autorizados* **(por contrato no publicado de client1)**.

Con client1 intentamos crear un progress log alterando el registration moment mediante inspeccionar elemento. Luego visualizamos el progress log creado y comprobamos que esos campos no se han modificado por el hack ya que no deben de poder ser modificables.

Con client1 intentamos crear un progress log alterando el id mediante inspeccionar elemento a 778 **(progress log ya existente)** para comprobar que no lo sobreescribe.

Con client1 intentamos crear progress logs hackeando con inyección en los atributos comment y responsible person (**<h1>!</h1>** y **' or 'A'='A**) para comprobar que toma dichos valores como simples string.

update.safe

Con el cliente1 se comprueba exactamente lo mismo que en create.safe editando el progress log PG-T-0000 (id 802), aunque en este caso no hay que comprobar valores de recordId porque no puede modificarse una vez asignado.

update.hack

Sin iniciar sesión entrar en *localhost:8082/Acme-SF-D04/client/progress-log/update* para ver que no estamos autorizados.

Con client2 intentamos actualizar un progress log alterando la id a 802 con inspeccionar elemento esperando no ser autorizados **(por progress log sin publicar de contrato de cliente1)**. Repetimos para id 778 **(por progress log publicado de contrato de client1)** y para id 2000 **(por progress log inexistente)**.

Con client1 intentamos actualizar un progress log alterando la id a 778 con inspeccionar elemento esperando no ser autorizados **(por progress log publicado de contrato de client1)**.

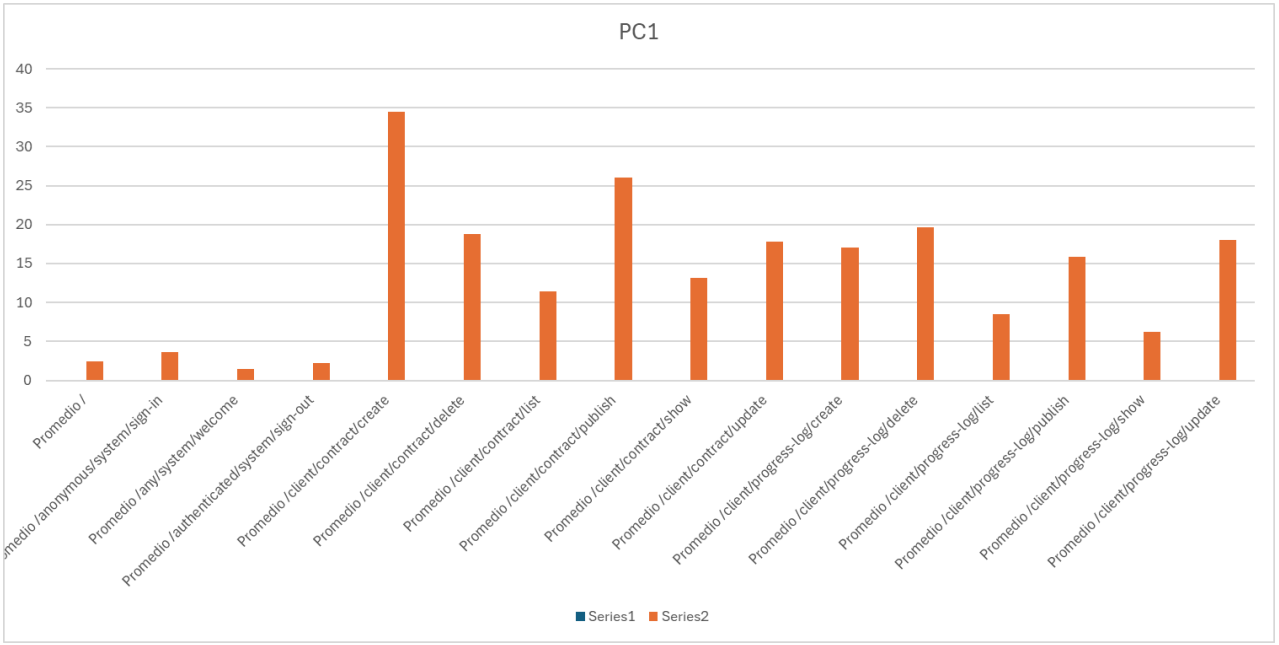
Con client1 intentamos actualizar cualquier progress log en borrador alterando el record id y el registration moment mediante inspeccionar elemento. Luego visualizamos el progress log actualizado y comprobamos que esos campos no se han modificado por el hack.

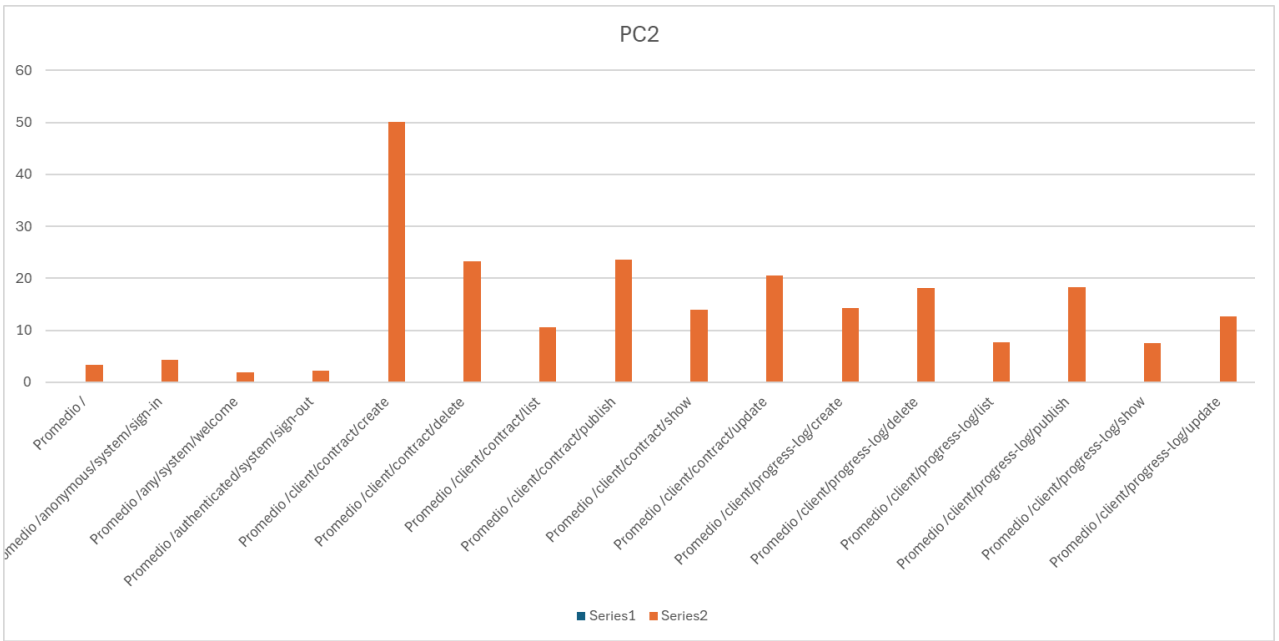
Con client1 intentamos actualizar cualquier progress log hackeando con inyección en los atributos comment y responsible person (**<h1>!</h1>** y **' or 'A'='A**) para comprobar que toma dichos valores como simples string.

Resultados de pruebas de rendimiento

Para probar el rendimiento del sistema respecto a las pruebas se ha lanzado el replayer en 2 ordenadores personales distintos. Para ambos casos se ha usado la versión del sistema con las optimizaciones de índices.

Los gráficos de promedios de tiempo obtenidos en el primer y segundo ordenador son los siguientes:





Para ser más precisos estos son los datos que aparecen en las gráficas:

request-path	time
Promedio /	2,4717414
Promedio /anonymous/system/sign-in	3,6546484
Promedio /any/system/welcome	1,4850409
Promedio /authenticated/system/sign-out	2,23765
Promedio /client/contract/create	34,442267
Promedio /client/contract/delete	18,788929
Promedio /client/contract/list	11,406436
Promedio /client/contract/publish	26,019925
Promedio /client/contract/show	13,201079
Promedio /client/contract/update	17,859218
Promedio /client/progress-log/create	17,012035
Promedio /client/progress-log/delete	19,616829
Promedio /client/progress-log/list	8,4708605
Promedio /client/progress-log/publish	15,856114
Promedio /client/progress-log/show	6,1776897
Promedio /client/progress-log/update	18,06

request-path	time
Promedio /	3,3704655
Promedio /anonymous/system/sign-in	4,2703806
Promedio /any/system/welcome	1,8569136
Promedio /authenticated/system/sign-out	2,2693125
Promedio /client/contract/create	50,051214
Promedio /client/contract/delete	23,246157
Promedio /client/contract/list	10,614618
Promedio /client/contract/publish	23,649563
Promedio /client/contract/show	13,967153
Promedio /client/contract/update	20,538818
Promedio /client/progress-log/create	14,208878
Promedio /client/progress-log/delete	18,120614
Promedio /client/progress-log/list	7,7082326
Promedio /client/progress-log/publish	18,310243
Promedio /client/progress-log/show	7,5982448
Promedio /client/progress-log/update	12,691022

A simple vista las gráficas son bastante similares con promedios por lo general mejores en PC1. Observemos ahora los intervalos de confianza del 95% para el tiempo de respuesta de los ordenadores:

PC1			*PC2*		
Media	10,7701821		Media	11,8635394	
Error típico	0,67451065		Error típico	0,73120785	
Mediana	5,1654		Mediana	6,8348	
Moda	0,9845		Moda	9,9764	
Desviación estándar	15,0372009		Desviación estándar	16,3011798	
Varianza de la muestra	226,117412		Varianza de la muestra	265,728463	
Curtosis	19,7122333		Curtosis	26,1276845	
Coeficiente de asimetría	3,63367212		Coeficiente de asimetría	4,12143732	
Rango	142,1625		Rango	172,2513	
Mínimo	0,6924		Mínimo	1,0622	
Máximo	142,8549		Máximo	173,3135	
Suma	5352,7805		Suma	5896,1791	
Cuenta	497		Cuenta	497	
Nivel de confianza(95,0%)	1,32525038		Nivel de confianza(95,0%)	1,43664667	
Interval (ms)	9,44493171	12,0954325	Interval (ms)	10,4268928	13,3001861
Interval (s)	0,00944493	0,01209543	Interval (s)	0,01042689	0,01330019

PC1 tiene un intervalo de confianza 95% (9.44, 12.09) y PC2 tiene intervalo (10.42, 13.3), son intervalos aceptables.

A continuación, realizamos la prueba z para medias de dos muestras:

Prueba z para medias de dos muestras		
	PC1	*PC2*
Media	10,7701821	11,8635394
Varianza (conocida)	226,117412	265,728463
Observaciones	497	497
Diferencia hipotética de las medias	0	
z	-1,09907114	
P(Z<=z) una cola	0,13586852	
Valor crítico de z (una cola)	1,64485363	
Valor crítico de z (dos colas)	0,27173704	
Valor crítico de z (dos colas)	1,95996398	

Con todos estos datos podremos realizar un contraste de hipótesis: tenemos la hipótesis nula (H_o), y la hipótesis alternativa (H_1) siguientes:

- H_o : Uno de los ordenadores es más eficiente que el otro, pudiéndose comparar sus medias (en este caso PC1 sería más eficiente que PC2).
- H_1 : Ambos ordenadores poseen una eficiencia similar, no pudiéndose comparar sus medias.

Para un nivel de confianza del 95%, tenemos un nivel de significación (α) de 0.05. El p-value obtenido es de 0.27, como este valor no se encuentra en el intervalo [0.00, α), no tenemos información suficiente para aceptar la hipótesis nula, por lo que podemos aceptar la hipótesis alternativa: Ambos ordenadores poseen una eficiencia similar, no pudiéndose comparar sus medias.

Conclusión

Los tests se han realizado de la manera más rigurosa posible, alcanzando un coverage muy satisfactorio que sólo incluye unos pocos casos en los que no es posible alcanzar todas las ramas de ejecución e igualmente no muestran ningún tipo de riesgo al cliente.

Por otro lado, las pruebas de rendimiento han permitido observar la eficiencia de dos ordenadores distintos, para así poder compararlos. Como resultado ambos ordenadores parecen tener un rendimiento similar, sin embargo cabe aclarar que sería recomendable realizar más pruebas de rendimiento utilizando unos tests con muchas más observaciones (en este caso tenemos sólo 497). Con dichas pruebas se podría medir con más certeza qué hipótesis es la correcta.

Bibliografía

No aplica.