Test Report- D04



Integrantes del grupo C1.026:

* Ignacio Blanquero Blanco ([ignblabla@alum.us.es](mailto:ignblabla@alum.us.es))
* Adrián Cabello Martín ([adrcabmar@alum.us.es](mailto:adrcabmar@alum.us.es))
* María de la Salud Carrera Talaverón ([marcartal1@alum.us.es](mailto:marcartal1@alum.us.es))
* Joaquín González Ganfornina ([joagongan@alum.us.es](mailto:joagongan@alum.us.es))
* Natalia Olmo Villegas ([natolmvil@alum.us.es](mailto:natolmvil@alum.us.es))

Fecha: Sevilla, 27 de mayo 2024

Tabla de contenido

[Resumen del Informe 5](#_Toc167724671)

[Historial de Versiones 6](#_Toc167724672)

[Introducción 7](#_Toc167724673)

[Testing funcional 8](#_Toc167724674)

[ClientContractListService.java 9](#_Toc167724675)

[ClientContractShowService.java 10](#_Toc167724676)

[ClientContractCreateService.java 11](#_Toc167724677)

[ClientContractUpdateService.java 13](#_Toc167724678)

[ClientContractDeleteService.java 15](#_Toc167724679)

[ClientContractPublishService.java 16](#_Toc167724680)

[ClientContractController.java 18](#_Toc167724681)

[ClientProgressLogListService.java 20](#_Toc167724682)

[ClientProgressLogShowService.java 21](#_Toc167724683)

[ClientProgressLogCreateService.java 22](#_Toc167724684)

[ClientProgressLogUpdateService.java 24](#_Toc167724685)

[ClientProgressLogPublishService.java 26](#_Toc167724686)

[ClientProgressLogDeleteService.java 28](#_Toc167724687)

[ClientProgressLogControllerService.java 30](#_Toc167724688)

[Rendimiento del testing 31](#_Toc167724689)

[Performance-testing sin índices 31](#_Toc167724690)

[Performance-testing con índices 33](#_Toc167724691)

[Performance-testing con ordenador del compañero 35](#_Toc167724692)

[Comparación sin índices - con índices 37](#_Toc167724693)

[Comparación distintos ordenadores 38](#_Toc167724694)

[Ordenador Natalia 41](#_Toc167724695)

[Ordenador Adrián 44](#_Toc167724696)

[Conclusiones 47](#_Toc167724697)

[Bibliografía 48](#_Toc167724698)

# Resumen del Informe

Este informe se divide en dos secciones principales: testing funcional, donde se proporciona un listado detallado de los casos de prueba implementados, organizados por características del sistema. Cada caso de prueba incluye una descripción concisa y una evaluación de su efectividad en la detección de errores; y análisis del rendimiento, donde se presentan gráficos que ilustran el tiempo de respuesta (wall time) del sistema al ejecutar los tests funcionales en dos computadoras diferentes, así como los resultados obtenidos con índices y sin índices. Se calcula un intervalo de confianza del 95% para estos tiempos y se determinará cuál de las dos computadoras es más poderosa en términos de rendimiento al servir las solicitudes del sistema.

# Historial de Versiones

|  |  |  |  |
| --- | --- | --- | --- |
| **Versión** | **Contenidos** | **Fecha** | **Contribuyente** |
| V0.1 | Estructura general del informe | 20/05/2024 | Natalia Olmo Villegas |
| V.1.0 | Análisis cobertura testing funcional | 23/05/2024 | Natalia Olmo Villegas |
| V1.1 | Análisis rendimiento | 26/05/2024 | Natalia Olmo Villegas |
| V2.0 | Repaso final | 27/05/2024 | Natalia Olmo Villegas |

# Introducción

El presente informe detalla el desarrollo y resultados de un conjunto de pruebas de software (test suite) aplicadas al proyecto desarrollado por este grupo Acme-SF. El objetivo principal es evaluar tanto la funcionalidad como el rendimiento del sistema bajo prueba. Para lograr esto, se han implementado diversas pruebas funcionales agrupadas por características, cada una diseñada para detectar errores y asegurar que las funcionalidades del sistema operen conforme a lo esperado. Además, se realiza un análisis del rendimiento, comparando el tiempo de respuesta del sistema en dos computadoras diferentes, con el fin de determinar cuál de ellas ofrece un mejor desempeño.

# Testing funcional

En el proceso de pruebas de la entidad “Contract.java”, se ha logrado alcanzar una cobertura del 95,2%. Esto es un indicador muy positivo del nivel de exhaustividad de nuestras pruebas. Esta cobertura se traduce en que, de un total de 1476 instrucciones que componen esta entidad, 1405 han sido cubiertas por los casos de prueba, dejando solo 71 instrucciones sin cubrir.



Aunque el 95,2% de cobertura es un excelente indicador, es importante recordar que el objetivo no es alcanzar el 100% de cobertura, sino asegurar que las áreas más críticas y susceptibles a errores del código están bien cubiertas, lo cual ha sido cumplido con éxito, pues ninguna de esas áreas corresponde a esas 71 líneas sin cubrir.

El alto rango de cobertura de la entidad "Contract.java" contribuye enormemente a la calidad y estabilidad del software desarrollado.

A continuación, se muestra con más detalle la cobertura de test de cada feature implementada para esta entidad. Cabe señalar que todos los test tienen líneas amarillas correspondientes a ***“assert object !=null”,*** lo cual es completamente normal ya que si esta línea se hubiera cubierto significaría que el sistema ha llegado a un fallo irreversible durante los casos de prueba. Son líneas que están para prevenir, pero afortunadamente no han sido cubiertas.

Asimismo, es importante destacar que sólo se han tenido en cuenta los posibles hackeos que consistían en insertar urls en la barra de búsqueda, debido a que se desconocía como implementar otros tipos de hacking en este proyecto concretamente.

## ClientContractListService.java

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteTexto

Descripción generada automáticamente



En esta primera característica observamos que todas las líneas de la clase han sido cubiertas por completo y se ha obtenido un alto porcentaje de cobertura. Ha conseguido pasar todas las pruebas con éxito y no se encontró ningún fallo respecto a esta característica durante el desarrollo de los casos de prueba. La única modificación que se realizó fue la actualización del Payload debido a la actualización del framework.

## ClientContractShowService.java

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente



En esta característica observamos que todas las líneas de la clase han sido cubiertas por completo y se ha obtenido un alto porcentaje de cobertura. Ha conseguido pasar todas las pruebas con éxito y no se encontró ningún fallo respecto a esta característica durante el desarrollo de los casos de prueba. No obstante, se consiguió simplificar el método authorise(), pues estaba cubierto en amarillo y se hicieron varias modificaciones, siempre teniendo en cuenta que otro usuarios maliciosos no puedan hackear nuestra sistema.

## ClientContractCreateService.java Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico Descripción generada automáticamente Interfaz de usuario gráfica, Texto Descripción generada automáticamente Texto Descripción generada automáticamente Interfaz de usuario gráfica, Texto, Aplicación Descripción generada automáticamente

En esta característica observamos que todas las líneas de la clase han sido cubiertas por completo y se ha obtenido un alto porcentaje de cobertura. Ha conseguido pasar todas las pruebas con éxito. Sin embargo, mediante la realización de esta prueba se pudo detectar un fallo. Este consistía en que no se estaba contemplando la posibilidad de que se quisiera crear un contrato a un proyecto no existente, y por tanto, esta excepción no era tratada. Afortunadamente, se logró manejar la excepción introduciendo el código necesario para ello en el método validate. Esta excepción sería también modificada en las clases ClientContractUpdateService.java y ClientContractPublishService.java correspondientes. Además, se modificó el método authorise para evitar un posible hackeo de un usuario externo que no estuviera logueado como cliente, pues anteriormente en authorise la única restricción que existía para crear un contrato era esta logueado. se encontró ningún fallo respecto a esta característica durante el desarrollo de los casos de prueba.

## ClientContractUpdateService.java

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Texto

Descripción generada automáticamente con confianza mediaInterfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente



En esta característica observamos que todas las líneas de la clase han sido cubiertas por completo menos la línea que se encarga de la declaración de la variable status en el método authorise(). Esto es debido a que se ha considerado que se podría hacer algún tipo de hacking para ello, pero no se ha encontrado la url necesaria para realizar este hacking en el conjunto de pruebas o no se podía acceder a ella ya que no aparecían ids en las URL para ello. El tipo de test que cubriría esta línea amarilla sería *“update-right-role-wrong-user” y “update-right-role-right-user-wrong-action”*, por eso indica que debe tener el rol cliente y que esté en modo borrador, pues un intento de hacking malicioso podría ser intentar editar un contrato que está publicado. Por otro lado, se ha obtenido un alto porcentaje de cobertura por lo que no se ha considerado un gran problema que esa línea no estuviera cubierta al completo, pues no es una línea roja. Ha conseguido pasar todas las pruebas con éxito y el fallo encontrado, comentado antes en la característica de crear contrato, fue solventado.

## ClientContractDeleteService.java

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente Texto

Descripción generada automáticamente



En esta característica observamos que todas las líneas de la clase han sido cubiertas por completo menos la línea que se encarga de la declaración de la variable status en el método authorise(). Esto es debido a que se ha considerado que se podría hacer algún tipo de hacking para ello, pero no se ha encontrado la url necesaria para realizar este hacking en el conjunto de pruebas. El tipo de test que cubriría esta línea amarilla sería *“delete-right-role-wrong-user” y “delete-right-role-right-user-wrong-action”*, por eso indica que debe tener el rol cliente y que esté en modo borrador, pues un intento de hacking malicioso podría ser intentar borrar un contrato que está publicado. Por otro lado, se ha obtenido un alto porcentaje de cobertura por lo que no se ha considerado un gran problema que esa línea no estuviera cubierta al completo, pues no es una línea roja. Ha conseguido pasar todas las pruebas con éxito y sin fallos. Asimismo, es importante destacar que el método unbind() se ha eliminado de esta clase porque no se consideraba necesario. Se pueden eliminar todos los datos de ese contrato y eliminarlo, pues se ha considerado que no tiene sentido implementar ninguna validación para ello, ya que si el contrato se va a eliminar no importa si sus datos son válidos o no, pues no van a quedar registrados en ninguna parte y se van a eliminar de la base de datos.

## ClientContractPublishService.java

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente Imagen que contiene Texto

Descripción generada automáticamente Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente



En esta característica observamos que todas las líneas de la clase han sido cubiertas por completo menos la línea que se encarga de la declaración de la variable status en el método authorise(). Esto es debido a que se ha considerado que se podría hacer algún tipo de hacking para ello, pero no se ha encontrado la url necesaria para realizar este hacking en el conjunto de pruebas. El tipo de test que cubriría esta línea amarilla sería *“publish-right-role-wrong-user” y “publish-right-role-right-user-wrong-action”*, por eso indica que debe tener el rol cliente y que esté en modo borrador, pues un intento de hacking malicioso podría ser intentar publicar un contrato que está publicado. Por otro lado, se ha obtenido un alto porcentaje de cobertura por lo que no se ha considerado un gran problema que esa línea no estuviera cubierta al completo, pues no es una línea roja. Ha conseguido pasar todas las pruebas con éxito y el fallo encontrado, comentado antes en la característica de crear contrato, fue solventado.

## ClientContractController.java

Tabla

Descripción generada automáticamente Texto

Descripción generada automáticamente



Esta clase ha conseguido ser cubierta al 100%, sin ningún problema.

A continuación, analizaremos la segunda entidad a realizar por el estudiante 2, ProgressLog.java Se ha conseguido un 93,4% de cobertura en esta.Esto es un indicador muy positivo del nivel de exhaustividad de nuestras pruebas. Esta cobertura se traduce en que, de un total de 1112 instrucciones que componen esta entidad, 1191 han sido cubiertas por los casos de prueba, dejando solo 79 instrucciones sin cubrir.



Aunque el 93,4% de cobertura es un excelente indicador, es importante recordar que el objetivo no es alcanzar el 100% de cobertura, sino asegurar que las áreas más críticas y susceptibles a errores del código están bien cubiertas, lo cual ha sido cumplido con éxito, pues ninguna de esas áreas corresponde a esas 79 líneas sin cubrir.

El alto rango de cobertura de la entidad "Contract.java" contribuye enormemente a la calidad y estabilidad del software desarrollado.

## ClientProgressLogListService.java

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente



En esta primera característica observamos que todas las líneas de la clase han sido cubiertas por completo, menos la línea que se encarga de la declaración de la variable status en el método authorise(). Esto es debido a que se ha considerado que se podría hacer algún tipo de hacking para ello, pero no se ha encontrado la url necesaria para realizar este hacking en el conjunto de pruebas. Por otro lado, se ha obtenido un alto porcentaje de cobertura por lo que no se ha considerado un gran problema que esa línea no estuviera cubierta al completo, pues no es una línea roja. Ha conseguido pasar todas las pruebas con éxito y no se encontró ningún fallo respecto a esta característica durante el desarrollo de los casos de prueba. La única modificación que se realizó fue la actualización del Payload debido a la actualización del framework.

## ClientProgressLogShowService.java

Interfaz de usuario gráfica, Texto, Aplicación, Carta

Descripción generada automáticamente

Texto

Descripción generada automáticamente con confianza baja



En esta característica observamos que todas las líneas de la clase han sido cubiertas por completo y se ha obtenido un alto porcentaje de cobertura. Ha conseguido pasar todas las pruebas con éxito y no se encontró ningún fallo respecto a esta característica durante el desarrollo de los casos de prueba. No obstante, se consiguió simplificar el método authorise(), pues estaba cubierto en amarillo y se hicieron varias modificaciones, siempre teniendo en cuenta que otro usuarios maliciosos no puedan hackear nuestra sistema.

## ClientProgressLogCreateService.java

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente Texto

Descripción generada automáticamente Texto

Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente



En esta característica observamos que todas las líneas de la clase han sido cubiertas por completo y se ha obtenido un alto porcentaje de cobertura. Ha conseguido pasar todas las pruebas con éxito, menos la línea que se encarga de la declaración de la variable status en el método authorise(). Esto es debido a que se ha considerado que se podría hacer algún tipo de hacking para ello, pero no se ha encontrado la url necesaria para realizar este hacking en el conjunto de pruebas. El tipo de test que cubriría esta línea amarilla sería *“create-right-role-wrong-user””*, por eso indica que debe tener el rol cliente y que esté en modo borrador, pues un intento de hacking malicioso podría ser intentar publicar un contrato que está publicado. Por otro lado, se ha obtenido un alto porcentaje de cobertura por lo que no se ha considerado un gran problema que esa línea no estuviera cubierta al completo, pues no es una línea roja. Sin embargo, mediante la realización de esta prueba se pudo detectar un fallo. Este consistía en que no se estaba contemplando la posibilidad de que se quisiera crear un informe de progreso con un grado de completitud determinado si ese contrato no tenía todavía ningún informe de progreso publicado. El grado de completitud es necesario que se compare con los anteriores que hayan sido publicados para restringir que este sea mayor, pero si no había ningún informe de progreso publicado no se podía hacer esta comparación. Afortunadamente, se logró manejar la excepción introduciendo el código necesario para ello en el método validate. Esta excepción sería también modificada en las clases ClientPregressLogUpdateService.java y ClientProgressLogPublishService.java correspondientes.

## ClientProgressLogUpdateService.java

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente Tabla

Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente



En esta característica observamos que todas las líneas de la clase han sido cubiertas por completo menos la línea que se encarga de la declaración de la variable status en el método authorise(). Esto es debido a que se ha considerado que se podría hacer algún tipo de hacking para ello, pero no se ha encontrado la url necesaria para realizar este hacking en el conjunto de pruebas. Por otro lado, se ha obtenido un alto porcentaje de cobertura por lo que no se ha considerado un gran problema que esa línea no estuviera cubierta al completo, pues no es una línea roja. Ha conseguido pasar todas las pruebas con éxito y el fallo encontrado, comentado antes en la característica de crear un informe de progreso, fue solventado.

## ClientProgressLogPublishService.java

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente Texto

Descripción generada automáticamente Imagen que contiene Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente



En esta característica observamos que todas las líneas de la clase han sido cubiertas por completo menos la línea que se encarga de la declaración de la variable status en el método authorise(). Esto es debido a que se ha considerado que se podría hacer algún tipo de hacking para ello, pero no se ha encontrado la url necesaria para realizar este hacking en el conjunto de pruebas. El tipo de test que cubriría esta línea amarilla sería *“publish-right-role-wrong-user” y “publish-right-role-right-user-wrong-action”*, por eso indica que debe tener el rol cliente y que esté en modo borrador, pues un intento de hacking malicioso podría ser intentar publicar un informe de progreso que está publicado. Por otro lado, se ha obtenido un alto porcentaje de cobertura por lo que no se ha considerado un gran problema que esa línea no estuviera cubierta al completo, pues no es una línea roja. Ha conseguido pasar todas las pruebas con éxito y el fallo encontrado, comentado antes en la característica de crear un informe de progreso, fue solventado.

## ClientProgressLogDeleteService.java

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente Texto

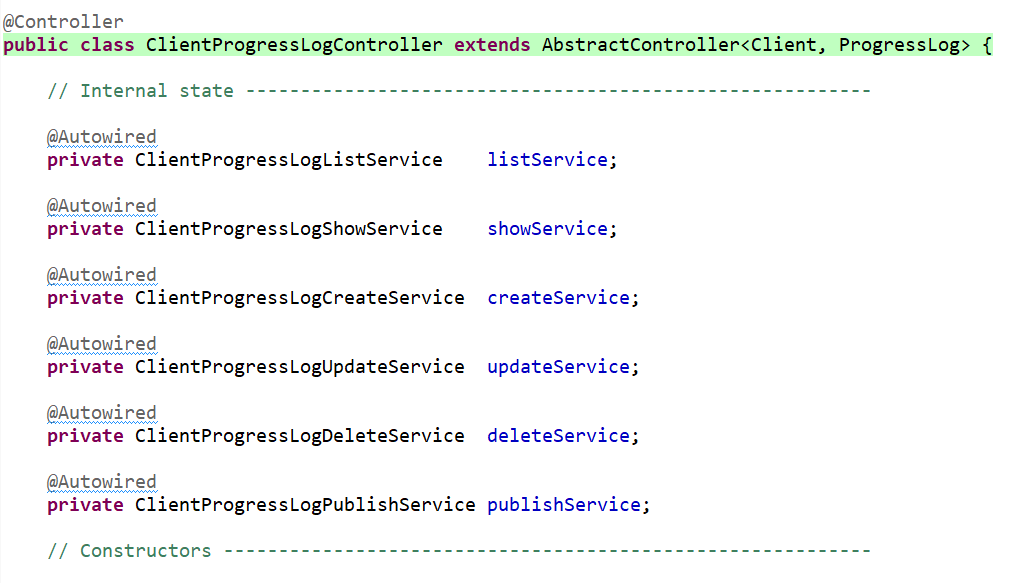
Descripción generada automáticamente con confianza media Texto

Descripción generada automáticamente



En esta característica observamos que todas las líneas de la clase han sido cubiertas por completo menos la línea que se encarga de la declaración de la variable status en el método authorise(). Esto es debido a que se ha considerado que se podría hacer algún tipo de hacking para ello, pero no se ha encontrado la url necesaria para realizar este hacking en el conjunto de pruebas. El tipo de test que cubriría esta línea amarilla sería *“delete-right-role-wrong-user” y “delte-right-role-right-user-wrong-action”*, por eso indica que debe tener el rol cliente y que esté en modo borrador, pues un intento de hacking malicioso podría ser intentar borrar un informe de progreso que está publicado. Por otro lado, se ha obtenido un alto porcentaje de cobertura por lo que no se ha considerado un gran problema que esa línea no estuviera cubierta al completo, pues no es una línea roja. Ha conseguido pasar todas las pruebas con éxito y se encontró un fallo. Como se ha comentado anteriormente, para borrar un contrato se eliminó el método unbind(), pues no se consideraba necesario y no ocasionó ningún problema. Sin embargo, a la hora de hacer lo mismo para esta entidad, sí que surgió un problema, pues se mostraba un error 500 si se vaciaba el formulario y se intentaba borrar un informe de progreso, por lo que en este caso si que se tuvo que añadir el método unbind() para manejar esta excepción, lo cual hizo bajar un poco el porcentaje de cobertura. No obstante, hay que saber priorizar que la aplicación funciones correctamente a tener un rango de cobertura un poco más elevado.

## ClientProgressLogControllerService.java

 Texto

Descripción generada automáticamente



Esta clase ha conseguido ser cubierta al 100%, sin ningún problema.

# Rendimiento del testing

Se han realizado varios “*performance-testing*”, uno de ellos sin los índices correspondientes para agilizar las querys SQL, otro de ellos con los índices que acabamos de mencionar y otro de ellos realizado en el ordenador de uno de los integrantes del grupo, ya que dependiendo del ordenador dónde se realice se obtendrán unos resultados u otros.

## Performance-testing sin índices

Hablemos, en primero lugar del *performance-testing* cuando todavía no se habían introducido los índices. Tras la creación del Excel se obtuvieron los siguientes resultados:

**1)Tester.trace en excel**

Imagen que contiene Tabla

Descripción generada automáticamente

**2) Promedio de cada una de las peticiones y el tiempo que se ha tardado en realizar cada una para analizar el rendimiento**

Gráfico

Descripción generada automáticamente

Gráfico

Descripción generada automáticamente

Como podemos observar, la petición más ineficiente con diferencia es la de listar contratos. Se ha deducido que la causa de esto puede ser porque para probas todas las demás características de la aplicación se debe acceder primero a la lista de contratos, lo cual va acumulando en ella una gran cantidad de peticiones afectando a su rendimiento negativamente.

Una posible solución podría ser implementar otras pestañas para que el usuario pueda acceder directamente las características determinadas sin tener que pasar por la lista de contratos. No obstante, esto contribuiría a una interfaz de usuario muchísimo menos intuitiva y uno de los requisitos fundamentales del cliente es que la interfaz sea lo más simple posible, por lo que se ha decidido dejar este problema así. Además, esto no supone un gran problema que se debe solucionar, pues fijándonos en el eje y se ve que la barra no llega ni siquiera al 60, lo cual es un número considerablemente bajo.

[Aquí](file:///C:\Users\natal\Workspace-24\Projects\Acme-SF-D04\logs\tester-performance-clean-uno.xlsx) se adjunta un enlace directo a este libro de Excel por si se requiere consultar en más detalle.

## Performance-testing con índices

A continuación, analizamos el segundo caso, en el que ya se habían introducido los índices.

Estos índices fueron los siguientes:  
  
Interfaz de usuario gráfica

Descripción generada automáticamente con confianza bajaImagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Tras la creación del Excel se obtuvieron los siguientes resultados:

**1)Tester.trace en excel**

Imagen que contiene Tabla

Descripción generada automáticamente

**2) Promedio de cada una de las peticiones y el tiempo que se ha tardado en realizar cada una para analizar el rendimiento**

Gráfico

Descripción generada automáticamente

Gráfico

Descripción generada automáticamente

Como podemos observar, la petición más ineficiente es la de borrar contratos. Se ha deducido que la causa de esto puede ser porque para probas todas las demás características de la aplicación se debe acceder primero a la lista de contratos, lo cual va acumulando en ella una gran cantidad de peticiones afectando a su rendimiento negativamente. No obstante, en este análisis ha subido también el tiempo promedio de las demás peticiones sobre todo las relacionas con crear, modificar y publicar contratos., obteniendo por tanto resultados peores.

En esta ocasión la introducción de índices no esté siendo favorable al rendimiento es que, por tanto, deducimos que a veces los algoritmos gastan más recursos buscando un índice determinado que en devolver un dato directamente. Los índices se han colocado correctamente siguiendo las indicaciones de las transparencias de clase así que se descarta la opción de que ese sea el origen del problema.

No obstante, esto no supone un gran problema, pues de todas las peticiones ninguna supera el 100, un número aceptable.

[Aquí](file:///C:\Users\natal\Workspace-24\Projects\Acme-SF-D04\logs\tester-performance-prueba.xlsx) se adjunta un enlace directo a este libro de Excel por si se requiere consultar en más detalle.

## Performance-testing con ordenador del compañero

A continuación, analizamos el último caso, en el que se ejecutaron los tests desde el ordenador de otro miembro del grupo. Tras la creación del Excel se obtuvieron los siguientes resultados:

**1)Tester.trace en excel**

Imagen que contiene Tabla

Descripción generada automáticamente

**2) Promedio de cada una de las peticiones y el tiempo que se ha tardado en realizar cada una para analizar el rendimiento**

Interfaz de usuario gráfica, Gráfico, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Gráfico, Aplicación

Descripción generada automáticamente

Como podemos observar, la petición más ineficiente con diferencia es la de listar contratos. Se ha deducido que la causa de esto puede ser porque para probas todas las demás características de la aplicación se debe acceder primero a la lista de contratos, lo cual va acumulando en ella una gran cantidad de peticiones afectando a su rendimiento negativamente.

Una posible solución podría ser implementar otras pestañas para que el usuario pueda acceder directamente las características determinadas sin tener que pasar por la lista de contratos. No obstante, esto contribuiría a una interfaz de usuario muchísimo menos intuitiva y uno de los requisitos fundamentales del cliente es que la interfaz sea lo más simple posible, por lo que se ha decidido dejar este problema así. Las demás peticiones, excepto el listado de informes de progreso, que en este caso también es algo más elevado, están cerca del 50, un número considerablemente bajo.

[Aquí](file:///C:\Users\natal\Workspace-24\Projects\Acme-SF-D04\logs\libro-adri.xlsx) y [aquí](file:///C:\Users\natal\Workspace-24\Projects\Acme-SF-D04\logs\tester-performance-clean-nat.xlsx) se adjunta un enlace directo a estos libros de Excel por si se requiere consultar en más detalle.

## Comparación sin índices - con índices

Primero, se hizo una comparación entre estos dos casos:

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamente

Analizando los intervalos, determinamos que el intervalo de confianza del 95% sin índices sería [15.73, 18.82] milisegundos, mientras que con índices será [32.56, 39.54] 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.

El objetivo de todo esto es determinar si los cambios han mejorado el MIR o no. Para ello se debe revisar si la diferencia entre las medias es cero o no está en un intervalo de confianza del 95%.

Para simplificar este proceso de comparación se realizará una prueba z, en este caso se han obtenido los siguientes resultados:

Tabla

Descripción generada automáticamente

Para analizar esta prueba es necesario fijarse en el valor de la celda “Valor crítico de z(dos colas)”. Como podemos comprobar, este valor (0) está entre [0.00,Alpha], lo que indica que se puede hacer una comparación de los tiempos medios. No obstante, 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.

## Comparación distintos ordenadores

Por último, comentaremos la comparación entre las pruebas con el índice y las pruebas realizadas en otro dispositivo:

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente

Aquí se han calculado varias medidas estadísticas conforme los datos analizados en el ordenador del estudiante 2, Natalia Olmo Villegas, y en el ordenador del estudiante 4, Adrián Cabello Martín.

Analizando los intervalos, determinamos que el intervalo de confianza del 95% para el ordenador de Natalia sería [32.56, 39.54] milisegundos, mientras que en el intervalo de Adrián será [33.35, 44.87] 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.

El objetivo de todo esto es determinar si los cambios han mejorado el MIR o no. Para ello se debe revisar si la diferencia entre las medias es cero o no está en un intervalo de confianza del 95%.

Para simplificar este proceso de comparación se realizará una prueba z, en este caso se han obtenido los siguientes resultados:

Tabla

Descripción generada automáticamente

Para analizar esta prueba es necesario fijarse en el valor de la celda “Valor crítico de z(dos colas)”. Como podemos comprobar, este valor (0,373076558) está entre [Alpha,1], lo que indica que no hay cambios significantes entre los resultados obtenidos por el ordenador de Natalia y el ordenador de Adrián. Los tiempos de ejemplo son diferentes, siendo el del ordenador de Adrián un poco más elevado, pero son prácticamente iguales. Por tanto, el rendimiento en ambos ordenadores es similar con respecto a software.

[Aquí](file:///C:\Users\natal\Workspace-24\Projects\Acme-SF-D04\logs\libro-comparaciones.xlsx) se adjunta un enlace directo a este libro de Excel por si se requiere consultar en más detalle.

Además, se ha utilizado la herramienta *visualvm* para hacer un mayor análisis de la situación y se han obtenido los siguientes resultados ordenando los métodos por el tiempo total:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Como podemos deducir, ye indicamos anteriormente con las gráficas, el método que más tiempo consume es el de listar contratos por las razones explicadas anteriormente. Sin embargo, podemos comprobar que el self time de este método es de 0 milisegundos, lo que indica que no es problema de nuestro método el hecho de que consuma más tiempo, si no que la causa de su mayor consumo de tiempo se debe a los métodos que invoca.

A continuación, vamos a inspeccionar las características de hardware de ambos ordenadores para determinar cuál es el ordenador más potente.

En este análisis, hemos examinado varios aspectos clave del rendimiento del ordenador ejecutando el launcher replayer para los casos de prueba del estudiante dos. Se ha obtenido entonces una gráfica que refleja métricas como el uso de la interfaz de red, el porcentaje de tiempo del procesador, la longitud de la cola del disco físico y el porcentaje de bytes almacenados en memoria. A continuación, se presenta una explicación detallada de estos resultados y una evaluación del rendimiento general del sistema de ambos ordenadores respectivamente.

### Ordenador Natalia

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Según la leyenda de colores, podemos ver que se ha representado de la siguiente forma:

* rojo: información del procesador
* negro (pone rojo 10 pero eso es negro): longitud actual de cola del disco físico
* verde: total de bytes de la interfaz de red
* azul: %de bytes confirmados en memoria

Se puede comprobar que con respecto al número total de bytes transferidos a través de la interfaz de red tenemos unos valores que se mantienen más o menos constantes y no es usado el 100% del tiempo. El uso constante y no saturado de la interfaz de red indica que las tareas de red no están sobrecargando el sistema. Esto sugiere que las transferencias de datos no son el principal cuello de botella en el rendimiento del sistema. Esto refleja que en este entorno la red no es intensamente utilizada o el ancho de banda disponible es suficiente para manejar la carga de trabajo sin problemas.

Por otro lado, acerca del porcentaje de tiempo del procesador cabe destacar que en la primera foto se produjo un cuello de botella entre el 2:7:00 y el 2:7:05, puede ser cuando se realizara la petición a la lista de contratos, que como hemos visto antes, es la característica que más recursos consume y sería una operación intensiva en términos de CPU. No obstante, el resto del tiempo su comportamiento es similar a la interfaz de red, sin nada remarcable. Dejando a un lado este momento, este nivel se mantiene en niveles normales, indicando que el sistema no está constantemente bajo alta carga de procesamiento, lo cual es un signo positivo de eficiencia.

Respecto a la longitud actual de la cola del disco físico, comprobamos que sus valores son más o menos contantes durante todo el recorrido. Estos son mayores en las dos primeras fotos, pero a medida que transcurre el tiempo va disminuyendo. Esta cola de disco constante y no excesivamente larga sugiere que el sistema de entrada/salida está manejando adecuadamente las solicitudes de lectura/escritura. Los valores mayores al inicio pueden deberse a operaciones de arranque o carga inicial de datos. La disminución de la longitud de la cola con el tiempo indica que el sistema está procesando eficientemente las operaciones de disco, sin acumular un número excesivo de solicitudes pendientes.

Por último, el porcentaje de bytes almacenados en memoria se mantiene constante en el 80% durante todo el proceso. Esto indica una buena gestión de la memoria, asegurando que hay suficiente espacio libre para operaciones adicionales sin alcanzar niveles críticos de utilización, es un comportamiento común en sistemas bien configurados, usando la memoria eficientemente sin que el sistema se sature.

En conclusión, el ordenador de Natalia es de buena calidad en términos de rendimiento. Los datos indican que, aunque se presenten momentos específicos de alta carga (como el cuello de botella en el procesador), el sistema en general maneja bien su carga de trabajo. La capacidad para mantener el uso de red, procesador y disco dentro de límites manejables, junto con una utilización constante y eficiente de la memoria, sugiere que el ordenador es adecuado para las tareas a las que está siendo sometido. El hecho de que no haya una saturación continua de recursos clave indica un buen equilibrio en la configuración y capacidad del hardware del ordenador.

### Ordenador Adrián

Interfaz de usuario gráfica

Descripción generada automáticamente Interfaz de usuario gráfica

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente

En esta ocasión, la leyenda de colores ha cambiado:

* rojo: información del procesador
* negro (pone rojo 10 pero eso es negro): longitud actual de cola del disco físico
* rosa: total de bytes de la interfaz de red
* morado: %de bytes confirmados en memoria

Se puede comprobar que con respecto al número total de bytes transferidos a través de la interfaz de red tenemos unos valores que se mantienen más o menos constantes y no es usado el 100% del tiempo, igual que en el ordenador de Natalia, de hecho, los del ordenador de Adrián tienden más a 0. Al igual que en el ordenador de Natalia, el uso constante y bajo de la interfaz de red en el ordenador de Adrián indica que las tareas de red no están sobrecargando el sistema.

Por otro lado, acerca del porcentaje de tiempo del procesador se obtienen resultados muy diversos, oscilando entre 40% y 60% su valor todo el tiempo, incluso en el momento próximo a 12:36:50 llega a 80%, y finalmente vuelve a 0 al final de la prueba. No obstante, no se ha producido ningún cuello de botella, como ha ocurrido en el ordenador de Natalia. Esta oscilación sugiere una carga de trabajo más dinámica y posiblemente más exigente que en el caso de Natalia. Sin embargo, la ausencia de cuellos de botella indica que el procesador está manejando eficientemente estas variaciones sin llegar a un punto de saturación.

Respecto a la longitud actual de la cola del disco físico, comprobamos que se producen algunos pequeños picos que no llegan a superar el valor de 20. Estos picos son menos frecuentes y no disminuyen con el tiempo, apareciendo esporádicamente. La presencia de pocos picos indica que las operaciones de entrada/salida del disco están bien manejadas, sin acumulación significativa de solicitudes. El hecho de que no haya una tendencia decreciente, como pasaba ne el ordenador de Natalia, sugiere una estabilidad en la carga de trabajo del disco.

Por último, el porcentaje de bytes almacenados en memoria se mantiene constante en el 60% aproximadamente durante todo el proceso, menos que el ordenador de Natalia. Esto indica una buena gestión de la memoria, asegurando que hay suficiente espacio libre para operaciones adicionales sin alcanzar niveles críticos de utilización, es un comportamiento común en sistemas

En conclusión, el ordenador de Adrián es un buen ordenador en términos de rendimiento. La capacidad de manejar eficientemente las demandas de red, procesamiento, disco y memoria, junto con la ausencia de problemas significativos de rendimiento, sugiere que el sistema está bien configurado y es capaz de manejar las tareas necesarias de manera efectiva.

Ahora vamos a determinar qué ordenador es mejor, analizando cada aspecto:

**Interfaz de red**

**-** Natalia: Uso constante, no llega al 100%

**-** Adrián: Uso constante, tiende más a 0

Los dos manejan bien la red, pero Adrián tiene un uso menor

**Tiempo del procesador**

- Natalia: Uso generalmente bajo, un cuello de botella notable

- Adrián: Uso variable entre 40% y 60%, con un pico al 80%, sin cuellos de botella

El ordenador de Adrián maneja mejor las variaciones en la carga del procesador sin llegar a un cuello de botella.

**Longitud actual de la cola del disco físico:**

- Natalia: Picos frecuentes, pero menores de 20, disminuyendo con el tiempo

- Adrián: Menos picos, no disminuyen, uno de vez en cuando

El ordenador de Adrián muestra menos picos y mayor estabilidad.

**Número de bytes almacenados en memoria:**

- Natalia: 80% constante.

- Adrián: 60% constante.

El ordenador de Adrián usa menos memoria.

Por lo que todo esto nos lleva a determinar que el ordenador de Natalia será mejor en términos de rendimiento que el de Adrián. Sin embargo, el motivo por el que en este caso el ordenador de Natalia mostrara los resultados 3 milisegundos antes que el ordenador de Adrián puede ser al uso de la memoria, ya que, aunque un menor uso de memoria (60%) es generalmente positivo, puede haber situaciones donde una mayor utilización de la memoria (80%) permita un acceso más rápido a datos frecuentemente utilizados.

# Conclusiones

Tras este análisis de los tests que han sido probados, se destaca que el grado de cobertura de las entidades a realizar por el estudiante dos, Natalia Olmo Villegas, ha sido bastante alto y aceptable. A su vez, se ha decretado el índice del 95% para el conjunto de casos de prueba sin los índices recomendados, para el otro conjunto que sí que ha utilizado los índices recomendados y por último para los casos de prueba probados desde otro dispositivo diferente. Por último, se han comparado varios ordenadores, de Adrián y Natalia para medir su rendimiento y se ha concluido que el ordenador de Adrián es mejor en aspectos de rendimiento.

# Bibliografía

En blanco a propósito.