DP2 2022-2023

D04 Informe de pruebas

Proyecto Acme L3

Repositorio: <https://github.com/alvgonfri/dp2-acme-l3>

Autores:

* González Frías, Álvaro ([alvgonfri@alum.us.es](mailto:alvgonfri@alum.us.es))
* del Río Pérez, Carlos ([cardelper3@alum.us.es](mailto:cardelper3@alum.us.es))
* Manzano Álvarez, Miguel ([migmanalv@alum.us.es](mailto:migmanalv@alum.us.es))
* Vega Rodríguez, Álvaro ([alvvegrod@alum.us.es](mailto:alvvegrod@alum.us.es))
* López Díaz, Sergio ([serlopdia@alum.us.es](mailto:serlopdia@alum.us.es))

GRUPO C1.03.02

23/05/2023

Índice

[Resumen ejecutivo 3](#_Toc136029100)

[Tabla de revisiones 4](#_Toc136029101)

[Introducción 5](#_Toc136029102)

[Contenido 6](#_Toc136029103)

[Pruebas funcionales 6](#_Toc136029104)

[Pruebas de rendimiento 6](#_Toc136029105)

[Conclusiones 11](#_Toc136029106)

[Bibliografía 12](#_Toc136029107)

# Resumen ejecutivo

El presente informe de pruebas proporciona una evaluación detallada de los resultados obtenidos en el proceso de pruebas del proyecto Acme L3. A lo largo del documento, se describirán cada uno de los test cases que han sido implementados, así como se mostrarán los resultados de las pruebas de rendimiento. La importancia de este informe recae en que es una herramienta que nos permite evaluar de manera objetiva la implementación exitosa del proyecto.

# Tabla de revisiones

|  |  |  |
| --- | --- | --- |
| **Nº Revisión** | **Fecha** | **Descripción de los cambios** |
| 1 | 23/05/2023 | * Creación del documento |
| 2 | 24/05/2023 | * Desarrollo del análisis |

# Introducción

Este documento ha sido creado con el objetivo de tener documentado todo lo referente a las pruebas que están relacionadas con los requisitos grupales.

En cuanto a la organización de la sección "Contenido", se han creado dos subapartados. Uno de ellos se corresponde con las pruebas funcionales, donde se indican todos los test cases implementados, agrupados por feature. En el otro subapartado se encuentran las pruebas de rendimiento, gracias a las cuales podemos analizar el rendimiento de las features y los test cases.

Por último, también es importante indicar la estructura general del documento. Tras una primera página en la que se puede ver la portada del informe junto con algo de información básica, encontramos un índice interactivo que nos sirve para localizar los diferentes apartados que posee el documento. A continuación, se puede leer el resumen ejecutivo en el que se discute sobre la motivación de realizar este informe, y después, se encuentra la tabla de revisiones, que contiene un registro de los cambios realizados. Tras ello, llegamos a este mismo punto, la introducción, en la que se mencionan aspectos tanto de iniciación al informe como de estructura. Si seguimos hacia delante, veremos la sección de contenido, en la que se encuentra la información principal del documento, así como la conclusión, donde se hace una reflexión sobre todo lo descrito anteriormente. Finalmente, encontramos la bibliografía, en la que se facilitará información sobre las fuentes consultadas, en caso de que corresponda.

# Contenido

## Pruebas funcionales

En esta sección se realiza una descripción de cada uno de los test cases que han sido implementados para cumplir con los requisitos grupales, agrupados por feature. Adicionalmente, al final del apartado, se analiza la efectividad de estos test cases a la hora de detectar bugs.

**Feature /any/peep/list:**

* *test100positive*: este test, sin iniciar sesión en la aplicación, lista los peeps y comprueba que el listado tiene los valores esperados. Emplea los datos del archivo */any/peep/list-positive.csv*.
* *test200Negative*: este test no comprueba nada ya que no hay casos de test negativos en un listado. No emplea datos de ningún archivo.
* *test300Hacking*: este test no comprueba nada ya que el listado de peeps es accesible para todo el mundo. No emplea datos de ningún archivo.

**Feature /any/peep/show:**

* *test100positive*: este test, sin iniciar sesión en la aplicación, lista los peeps, cliquea en uno de ellos y comprueba que el formulario tiene los valores esperados. Emplea los datos del archivo */any/peep/show-positive.csv*.
* *test200Negative*: este test no comprueba nada ya que no hay casos de test negativos en un show. No emplea datos de ningún archivo.
* *test300Hacking*: este test no comprueba nada ya que los detalles de los peeps son accesibles para todo el mundo. No emplea datos de ningún archivo.

**Feature /any/peep/create:**

* *test100positive*: este test, sin iniciar sesión en la aplicación, lista los peeps, crea uno nuevo y comprueba que la creación se ha completado con éxito. Emplea los datos del archivo */any/peep/create-positive.csv*.
* *test200Negative*: este test, sin iniciar sesión en la aplicación, lista los peeps, intenta crear uno nuevo con datos incorrectos y comprueba aparecen errores. Emplea los datos del archivo */any/peep/create-negative.csv*.
* *test300Hacking*: este test no comprueba nada ya que todo el mundo puede crear un peep. No emplea datos de ningún archivo.

Por último, cabe mencionar que los test no han servido para detectar ningún bug en el código. Esto se debe a que se realizaron pruebas informales exhaustivas de todas las funcionalidades, y también a que dichas funcionalidades fueron probadas por el profesor durante las distintas sesiones de seguimiento. Sin embargo, si consideramos que los test son útiles porque nos sirven para corroborar que realmente el código implementado cumple con los requisitos.

## Pruebas de rendimiento

Tras hacer este tipo de pruebas hemos obtenido una serie de datos y gráficos que nos sirven para conocer más acerca del rendimiento que tiene nuestro proyecto en cuanto a tiempos de espera cada vez que se realiza una petición. A continuación, se explican todos los pasos que se han dado para analizar el rendimiento del sistema correctamente.

Es importante mencionar que en este apartado vamos a considerar un supuesto requisito de rendimiento para el proyecto Acme-L3, que, aunque realmente no exige ninguno, sí que puede ser de ayuda para exponer de forma más clara el análisis de rendimiento. El requisito en cuestión es el siguiente:

“El sistema deberá atender las peticiones en menos de medio segundo de media.”

El primer paso fue el de ejecutar los test de la aplicación tras haber conseguido eliminar todos los errores y fallos que nos aparecían. Tras esto, dentro de la carpeta de logs del proyecto se generaron los archivos *tester-request-performance.csv* y *tester-test-performance.csv*, que fueron el punto de partida de las pruebas de rendimiento. Debido a que este documento es un informe grupal, se filtraron los dos archivos anteriores de forma que solo aparecieran líneas relacionadas con los requisitos grupales.

El archivo *tester-request-performance.csv* nos proporciona datos sobre las distintas peticiones que han sido realizadas mientras los test se estaban ejecutando. Gracias a Excel fue posible agrupar los datos de manera que se mostrara el tiempo de respuesta medio por cada feature (Figura 1), además de visualizar los datos gráficamente (Figura 2).

Tabla

Descripción generada automáticamente

*(Figura 1)*

*(Figura 2)*

Como se puede ver, todas las features tienen un tiempo de respuesta inferior a los 20 ms, lo que es bastante aceptable. La que más tarda es la feature del show de peeps, pero sin tener un valor muy exagerado con respecto al resto.

El fichero *tester-test-performance.csv* incluye información detallada por cada uno de los test cases ejecutados. Empleando un proceso similar al anterior, se pudieron obtener los tiempos que tardaron en ejecutarse cada uno de los test y presentarlos gráficamente (Figura 3).

*(Figura 3)*

Como podemos ver en la gráfica, algunos test negativos no tienen duración ya que no tiene sentido hacerlos en listados y shows. Además, tampoco se han implementado test de hacking por que los peeps son accesibles por todos. Por otro lado, parece ser que el test que más tiempo ha llevado ha sido el test positivo de crear peeps, algo normal ya que es el que más acciones debe realizar en el sistema.

Una vez obtenidos estos datos iniciales, el siguiente paso fue el de calcular el intervalo de confianza, con un nivel de confianza del 95 %. Para ello, se usaron los datos del archivo *tester-request-performance.csv* y se emplearon las herramientas para el análisis disponibles en Excel.

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamente

*(Figura 4)*

Tras completar este proceso, se obtuvieron datos generales que sirvieron para calcular el intervalo de confianza. En concreto, se calculó restando y sumando a la media el “Nivel de confianza (95,0%)”, y se pasó a segundos. El supuesto requisito que hemos impuesto antes exigía que el tiempo de respuesta fuera inferior a 0,5 segundos. En este caso, el requisito queda cumplido con creces ya que el límite superior del intervalo [0’01047, 0’01747] es inferior a 0,5.

Al haber superado este requisito, habríamos acabado con el análisis de rendimiento de forma exitosa. Sin embargo, tal y como se solicita en los anexos del proyecto, también se realizó el contraste de hipótesis entre los test ejecutados por dos ordenadores diferentes. En este caso, uno de los ordenadores fue el del Student #2 (Álvaro González), del que se han mostrado los datos anteriormente, y además se contó con la colaboración del Student #5 (Miguel Manzano) que también ejecutó los test en su ordenador. A continuación, se muestran los resultados obtenidos del análisis en el ordenador del Student #5 (Figuras 5, 6 y 7). Adicionalmente, se realizó el cálculo del intervalo de confianza con los datos del Student #5, cuyo resultado se podrá ver en la Figura 8.

Texto, Tabla

Descripción generada automáticamente

*(Figura 5)*

*(Figura 6)*

*(Figura 7)*

Para realizar el contraste de hipótesis, se creó un nuevo libro de Excel en el que se incluyeron los datos del tiempo de ejecución de cada petición realizada durante los test, en una columna para cada ordenador. También se muestran los intervalos de confianza calculados.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

*(Figura 8)*

Con el objetivo de comparar ambos resultados, se realizó un test Z, donde se supuso que el ordenador del Student #2 tenía mayor potencia por haber obtenido un tiempo de ejecución medio menor. El valor del parámetro alpha se sitúo en 0,05.

Tabla, Excel

Descripción generada automáticamente

*(Figura 9)*

Dentro de los resultados proporcionados por el test Z, lo importante es fijarse en el valor de la fila “P(Z<=z) dos colas”. Como podemos ver, dicho valor es 3.75678E-06 y consecuentemente se sitúa en el intervalo [0, alpha). En el supuesto de que hubiéramos obtenido este valor comparando los datos antes y después de haber introducido cambios en los test, podríamos concluir que se ha conseguido mejorar el rendimiento. Sin embargo, en nuestro caso estamos comparando los datos de los mismos test ejecutados por dos ordenadores distintos, por lo que la conclusión real es que el rendimiento del ordenador del Student #2 al ejecutar los test ha sido superior que el rendimiento del ordenador del Student #5.

# Conclusiones

En resumen, pensamos que ha sido positivo realizar este informe, debido a que el test de rendimiento ha sido de utilidad para saber si nuestro proyecto se encontraba bien optimizado o no. Como único punto negativo, destacaríamos que al no exigirse en los requisitos el realizar cambios en los test para mejorar el rendimiento (y así poder realizar un contraste de hipótesis con más sentido), las conclusiones que se pueden obtener del contraste de hipótesis entre los dos ordenadores no aportan nada especial.

# Bibliografía

Intencionadamente vacío.