DP2 2022-2023

Student #2 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) )

GRUPO C1.03.02

22/05/2023

Índice

[Resumen ejecutivo 3](#_Toc136029128)

[Tabla de revisiones 4](#_Toc136029129)

[Introducción 5](#_Toc136029130)

[Contenido 6](#_Toc136029131)

[Pruebas funcionales 6](#_Toc136029132)

[Pruebas de rendimiento 9](#_Toc136029133)

[Conclusiones 14](#_Toc136029134)

[Bibliografía 15](#_Toc136029135)

# 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 | 22/05/2023 | * Creación del documento |
| 2 | 23/05/2023 | * 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 individuales del Student #2.

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 individuales del Student #2, agrupados por feature. Adicionalmente, al final del apartado, se analiza la efectividad de estos test cases a la hora de detectar bugs.

**Feature /student/enrolment/list-mine:**

* *test100positive*: este test ingresa en la aplicación como un estudiante, lista sus matrículas (*enrolments*) y comprueba que el listado tiene los valores esperados. Emplea los datos del archivo */student/enrolment/list-mine-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 intenta listar sus matrículas mediante URL sin haber ingresado en la aplicación, y también habiendo ingresado con usuarios que no deberían tener acceso a dicho listado. Se comprueba que el resultado es que se muestra una pantalla de pánico. No emplea datos de ningún archivo.

**Feature /student/enrolment/show:**

* *test100positive*: este test ingresa en la aplicación como un estudiante, lista sus matrículas, cliquea en una de ellas y comprueba que el formulario tiene los valores esperados. Emplea los datos del archivo */student/enrolment/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 intenta mostrar el formulario de una matrícula mediante URL sin haber ingresado en la aplicación, y también habiendo ingresado con usuarios que no deberían tener acceso a dicho formulario. Se comprueba que el resultado es que se muestra una pantalla de pánico. No emplea datos de ningún archivo.

**Feature /student/enrolment/create:**

* *test100positive*: este test ingresa en la aplicación como un estudiante, lista sus matrículas, crea una nueva y comprueba que la creación se ha completado con éxito. Emplea los datos del archivo */student/enrolment/create-positive.csv*.
* *test200Negative*: este test ingresa en la aplicación como un estudiante, lista sus matrículas, intenta crear una nueva con datos incorrectos y comprueba aparecen errores. Emplea los datos del archivo */student/enrolment/create-negative.csv*.
* *test300Hacking*: este test intenta crear una matrícula mediante URL sin haber ingresado en la aplicación, y también habiendo ingresado con usuarios que no deberían poder crear matrículas. Se comprueba que el resultado es que se muestra una pantalla de pánico. No emplea datos de ningún archivo.

**Feature /student/enrolment/update:**

* *test100positive*: este test ingresa en la aplicación como un estudiante, lista sus matrículas, cliquea en una de ellas, la actualiza y comprueba que la actualización se ha completado con éxito. Emplea los datos del archivo */student/enrolment/update-positive.csv*.
* *test200Negative*: este test ingresa en la aplicación como un estudiante, lista sus matrículas, cliquea en una de ellas, la intenta actualizar con datos incorrectos y comprueba aparecen errores. Emplea los datos del archivo */student/enrolment/update-negative.csv*.
* *test300Hacking*: este test intenta actualizar una matrícula mediante URL sin haber ingresado en la aplicación, y también habiendo ingresado con usuarios que no deberían poder tener acceso a dicha matrícula. Se comprueba que el resultado es que se muestra una pantalla de pánico. No emplea datos de ningún archivo.

**Feature /student/enrolment/finalise:**

* *test100positive*: este test ingresa en la aplicación como un estudiante, lista sus matrículas, cliquea en una de ellas que no está finalizada, la finaliza y comprueba que se ha finalizado con éxito. Emplea los datos del archivo */student/enrolment/finalise-positive.csv*.
* *test200Negative*: este test ingresa en la aplicación como un estudiante, lista sus matrículas, cliquea en una de ellas que no está finalizada, la intenta finalizar sin introducir los datos de una tarjeta y comprueba aparecen errores. Emplea los datos del archivo */student/enrolment/finalise-negative.csv*.
* *test300Hacking*: este test intenta finalizar una matrícula mediante URL sin haber ingresado en la aplicación, y también habiendo ingresado con usuarios con un rol distinto al de estudiante. Se comprueba que el resultado es que se muestra una pantalla de pánico. No emplea datos de ningún archivo.
* *test301Hacking*: este test ingresa en la aplicación como un estudiante e intenta finalizar mediante URL una de sus matrículas que ya se encontraba finalizada. Se comprueba que el resultado es que se muestra una pantalla de pánico. No emplea datos de ningún archivo.
* *test302Hacking*: este test ingresa en la aplicación como un estudiante e intenta finalizar mediante URL una matrícula que no le pertenece. Se comprueba que el resultado es que se muestra una pantalla de pánico. No emplea datos de ningún archivo.

**Feature /student/enrolment/delete:**

* *test100positive*: este test ingresa en la aplicación como un estudiante que solo tiene una matrícula no finalizada, lista su matrícula, cliquea en ella, la borra y comprueba que su listado de matrículas se encuentra ahora vacío. Emplea los datos del archivo */student/enrolment/delete-positive.csv*.
* *test200Negative*: este test no comprueba nada ya que no hay casos de test negativos en un borrado. No emplea datos de ningún archivo.
* *test300Hacking*: este test intenta borrar una matrícula mediante URL sin haber ingresado en la aplicación, y también habiendo ingresado con usuarios que no deberían poder tener acceso a dicha matrícula. Se comprueba que el resultado es que se muestra una pantalla de pánico. No emplea datos de ningún archivo.

**Feature /student/activity/list:**

* *test100positive*: este test ingresa en la aplicación como un estudiante, lista sus matrículas, cliquea en una de ellas, lista sus actividades y comprueba que el listado tiene los valores esperados. Emplea los datos del archivo */student/activity/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 intenta listar las actividades de sus matrículas mediante URL sin haber ingresado en la aplicación, y también habiendo ingresado con usuarios que no deberían tener acceso a dicho listado. Se comprueba que el resultado es que se muestra una pantalla de pánico. No emplea datos de ningún archivo.

**Feature /student/activity/show:**

* *test100positive*: este test ingresa en la aplicación como un estudiante, lista sus matrículas, cliquea en una de ellas, lista sus actividades, cliquea en una de ellas y comprueba que el formulario tiene los valores esperados. Emplea los datos del archivo */student/activity/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 intenta mostrar el formulario de una actividad mediante URL sin haber ingresado en la aplicación, y también habiendo ingresado con usuarios que no deberían tener acceso a dicha actividad. Se comprueba que el resultado es que se muestra una pantalla de pánico. No emplea datos de ningún archivo.

**Feature /student/activity/create:**

* *test100positive*: este test ingresa en la aplicación como un estudiante, lista sus matrículas, cliquea en una de ellas, lista sus actividades, crea una nueva y comprueba que la creación se ha completado con éxito. Emplea los datos del archivo */student/activity/create-positive.csv*.
* *test200Negative*: este test ingresa en la aplicación como un estudiante, lista sus matrículas, cliquea en una de ellas, lista sus actividades, intenta crear una nueva con datos incorrectos y comprueba aparecen errores. Emplea los datos del archivo */student/activity/create-negative.csv*.
* *test300Hacking*: este test intenta crear una actividad mediante URL sin haber ingresado en la aplicación, y también habiendo ingresado con usuarios con un rol distinto al de estudiante. Se comprueba que el resultado es que se muestra una pantalla de pánico. No emplea datos de ningún archivo.
* *test301Hacking*: este test ingresa en la aplicación como un estudiante e intenta crear mediante URL una actividad en una matrícula que aún no ha sido finalizada. Se comprueba que el resultado es que se muestra una pantalla de pánico. No emplea datos de ningún archivo.
* *test302Hacking*: este test ingresa en la aplicación como un estudiante e intenta crear mediante URL una actividad en una matrícula que no le pertenece. Se comprueba que el resultado es que se muestra una pantalla de pánico. No emplea datos de ningún archivo.

**Feature /student/activity/update:**

* *test100positive*: este test ingresa en la aplicación como un estudiante, lista sus matrículas, cliquea en una de ellas, lista sus actividades, cliquea en una de ellas, la actualiza y comprueba que la actualización se ha completado con éxito. Emplea los datos del archivo */student/activity/update-positive.csv*.
* *test200Negative*: este test ingresa en la aplicación como un estudiante, lista sus matrículas, cliquea en una de ellas, lista sus actividades, cliquea en una de ellas, la intenta actualizar con datos incorrectos y comprueba aparecen errores. Emplea los datos del archivo */student/activity/update-negative.csv*.
* *test300Hacking*: este test intenta actualizar una actividad mediante URL sin haber ingresado en la aplicación, y también habiendo ingresado con usuarios que no deberían poder tener acceso a dicha actividad. Se comprueba que el resultado es que se muestra una pantalla de pánico. No emplea datos de ningún archivo.

**Feature /student/activity/delete:**

* *test100positive*: este test ingresa en la aplicación como un estudiante, lista sus matrículas, cliquea en una de ellas que tiene una sola actividad, lista la actividad, cliquea en ella, la borra y comprueba que el listado de actividades de dicha matrícula se encuentra ahora vacío. Emplea los datos del archivo */student/activity/delete-positive.csv*.
* *test200Negative*: este test no comprueba nada ya que no hay casos de test negativos en un borrado. No emplea datos de ningún archivo.
* *test300Hacking*: este test intenta borrar una actividad mediante URL sin haber ingresado en la aplicación, y también habiendo ingresado con usuarios que no deberían poder tener acceso a dicha actividad. Se comprueba que el resultado es que se muestra una pantalla de pánico. 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 considero 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 individual, se filtraron los dos archivos anteriores de forma que solo aparecieran líneas relacionadas con los requisitos del Student #2.

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)*

Gráfico

Descripción generada automáticamente

*(Figura 2)*

Como se puede ver, todas las features tienen un tiempo de respuesta bastante aceptable. La que más tarda es la feature del listado de matrículas, 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).

Gráfico

Descripción generada automáticamente

*(Figura 3)*

Lo que llama más la atención de esta gráfica es la diferencia entre los test de hacking y el resto. De estos resultados se puede deducir que los test de hacking son más lentos, probablemente porque en la mayoría de ellos hay que cerrar e iniciar sesión con varios usuarios, lo cual aumenta considerablemente el tiempo de ejecución. Sin embargo, sin tener en cuenta los test de hacking, podemos observar que ninguno tarda más de 30 segundos (30.000 ms) en ejecutarse.

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 [0’00966, 0’01129] 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 mío, del que se han mostrado los datos anteriormente, y además se contó con la colaboración del Student #3 (Álvaro Vega) que ejecutó los test en su ordenador y me proporcionó los datos necesarios para el proceso. A continuación, se muestran los resultados obtenidos del análisis en el ordenador del Student #3 (Figuras 5, 6 y 7). Adicionalmente, se realizó el cálculo del intervalo de confianza con los datos del Student #3, cuyo resultado se podrá ver en la Figura 8.

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamente

*(Figura 5)*

Gráfico

Descripción generada automáticamente

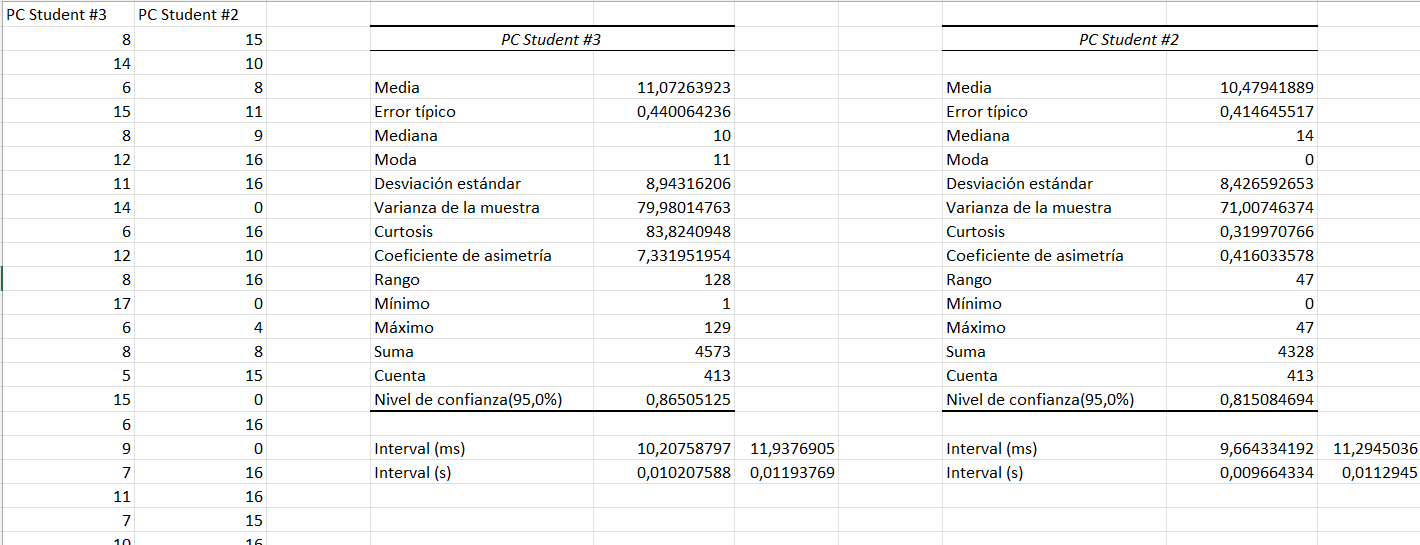
*(Figura 6)*

Gráfico

Descripción generada automáticamente

*(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.



*(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

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 0,3265… y consecuentemente se sitúa en el intervalo (alpha, 1]. 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 no se ha conseguido ningún cambio realmente significativo. 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 de los dos ordenadores al ejecutar los test es bastante similar.

# Conclusiones

En resumen, pienso que ha sido positivo realizar este informe, primeramente porque al realizar la descripción de las pruebas funcionales me he dado cuenta de algunas comprobaciones que deberían realizar mis test pero que no las estaban haciendo, como por ejemplo comprobar que saltaba la pantalla de pánico en el test de hacking de la finalización de matrículas. Por otro lado, el test de rendimiento ha sido de utilidad para saber si nuestro proyecto se encontraba bien optimizado o no. Como único punto negativo, destacaría 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.