Universidad de Sevilla  
Escuela Técnica Superior de Ingeniería Informática

**Diseño y Pruebas II**

**Testing Report**

**C2-Student 2**



Grado en Ingeniería Informática – Ingeniería del Software  
Curso 2024 – 2025

|  |  |  |
| --- | --- | --- |
| **Fecha** | **Versión** | **Repositorio** |
| 01/07/2025 | 1.0 | https://github.com/Javiergutpas/Acme-ANS-C2 |

|  |  |  |
| --- | --- | --- |
| **Grupo de prácticas: C2.032** | |  |
| **Autores por orden alfabético** | **Roles** | **Correo Corporativo** |
| Del Pino Escalante, Nuno José | Analyst Tester, Developer, Operator | nundelesc@alum.us.es |
| Gutiérrez Pastor, Javier | Manager, Tester, Developer | javgutpas@alum.us.es |

**Fecha:** 01/07/2025

**Tabla de contenidos**

[**1.** **Resumen Ejecutivo**: 2](#_Toc202287942)

[**2.** **Tabla de revisión**: 3](#_Toc202287943)

[**3.** **Introducción**: 4](#_Toc202287944)

[**4.** **Contenidos**: 5](#_Toc202287945)

[**4.1** **Testing funcional**: 5](#_Toc202287946)

[**5.** **Rendimiento de testing:** 13](#_Toc202287947)

[**5.1** **Resultados de la primera prueba (sin índices)**: 14](#_Toc202287948)

[**5.2** **Resultados de la primera prueba (con índices):** 15](#_Toc202287949)

[**5.3** **Resultados de la segunda prueba (sin índices en ordenador local):** 17](#_Toc202287950)

[**5.4** **Resultados de la segunda prueba (sin índices en otro ordenador):** 17](#_Toc202287951)

[**5.5** **Primera pruebaZ (mismo ordenador sin índices – con índices):** 19](#_Toc202287952)

[**5.6** **Segunda pruebaZ (distinto ordenador sin índices)** 20](#_Toc202287953)

[6 **Conclusiones**: 21](#_Toc202287954)

[7 **Bibliografía**: 22](#_Toc202287955)

### **Resumen Ejecutivo**:

El presente documento tiene como objetivo analizar la fase de pruebas, correspondiente a la etapa final de la entrega del proyecto. En él se detallan los distintos requisitos establecidos, con el fin de justificar y validar las soluciones propuestas y tomadas para cumplir con los mismos.

### **Tabla de revisión**:

|  |  |  |
| --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** |
| 01/07/2025 | 1.0 | Versión inicial del documento |
|  |  |  |
|  |  |  |
|  |  |  |

### **Introducción**:

Este informe ofrece una explicación de los requisitos establecidos para la fase de testing de esta última entrega, enfocándose en aquellos que requieren un estudio más profundo. Su propósito es documentar y justificar las acciones tomadas para garantizar el cumplimiento de dichos requisitos.

Durante el período transcurrido por el informe, se han realizado diversas tareas que han implicado la toma de decisiones clave durante la fase de prueba (testing), las cuales deben ser validadas para asegurar el cumplimiento adecuado de los requisitos y la satisfacción del cliente.

### **Contenidos**:

### **Testing funcional**:

En esta sección se presentan de forma general los distintos casos de prueba, tanto para los archivos .safe como para los archivos .hack (hacking), correspondiente a los requisitos individuales 8 y 9 del Student 2. Se ofrece una descripción resumida del procedimiento para probar cada uno de ellos y se indica si se ha detectado algún bug durante su ejecución.

**Funcionalidades de Reservas (*Booking*):**

|  |  |  |
| --- | --- | --- |
| **.safe** | **Procedimiento** | **Bugs encontrados** |
| List | Iniciamos sesión en el sistema con Customer1 y listamos todos los booking. | Ninguno |
| Show | Iniciamos sesión en el sistema con Customer 1, listamos los bookings, mostramos los detalles de uno sin pasajeros, return, mostramos los detalles de uno con pasajeros. | Ninguno |
| Create | Iniciamos sesión en el sistema con Customer 1, listamos los bookings, vamos al formulario de creación, enviamos vacío, empezando por el primer campo del formulario, probamos valores erróneos y luego correctos, dejamos el último correcto, y enviamos el formulario de creación correctamente | Ninguno |
| Update | Iniciamos sesión en el sistema con Customer 1, listamos los bookings, mostramos los detalles de uno sin pasajeros, borramos todo, actualizamos, ponemos valores correctos, actualizamos, mostramos los detalles de uno con pasajeros, actualizamos vacío, actualizamos con valores erróneos, actualizamos con valores correctos. | Ninguno |
| Publish | Iniciamos sesión en el sistema con Customer 1, mostramos los detalles de uno sin pasajeros, publicamos, añadimos pasajero, publicamos pasajero, publicamos booking vacio, publicamos booking con datos correctos, entro en un booking con pasajeros, borramos todo y publicamos, publicamos con valores erróneos, publicamos con valores correctos, publicamos, publicamos pasajero, publicamos booking. | Ninguno |
| Delete | Iniciamos sesión en el sistema con Customer 1, mostramos detalles de un booking sin pasajeros, cambiamos en la url "show" por "delete", eliminamos, mostramos detalles de un booking con pasajeros, cambiamos en la url "show" por "delete", eliminamos. | Ninguno |

|  |  |  |
| --- | --- | --- |
| **.hack** | **Descripción** | **Bugs encontrados** |
| List | Se probó si se podía acceder al listado de bookings sin estar logueado en la aplicación. | Ninguno |
| Show | Se probó acceder sin estar logueado al show de un booking sin publicar, otro publicado y otro inexistente, posteriormente si inició sesión con el Customer 2 y se probó a acceder al show de los mismos bookings. | Ninguno |
| Create | Se probó a acceder al create sin estar logueado. Posteriormente se inició sesión con el Customer 1 y se probó hackear el id y el value con valores inexistentes (999). Se probó a hackear el selector del flight poniendo ids de vuelos sin publicar, y vuelos inexistentes. Por último se probó a hackear el selector de travel Class con valores incorrectos como vacío y “HACKED”, además los valores readOnly se modificaron para ver si permitían el hackeo. | Ninguno |
| Update | Se probó a acceder al update de un booking sin publicar, otro publicado y otro inexistente sin estar logueado. Posteriormente se inició sesión con el Customer 2 y se probó acceder al update de los mismo bookings. Luego se intentó hackear el id y el value con valores inexistentes (999). Se probó a hackear el selector del flight poniendo ids de vuelos sin publicar, y vuelos inexistentes. Por último, se probó a hackear el selector de travel Class con valores incorrectos como vacío y “HACKED”, además los valores readOnly se modificaron para ver si permitían el hackeo. | Ninguno |
| Publish | Se probó a acceder al publish de un booking sin publicar, otro publicado y otro inexistente sin estar logueado. Posteriormente se inició sesión con el Customer 2 y se probó acceder al publish de los mismo bookings. Luego se intentó hackear el id y el value con valores inexistentes (999). Se probó a hackear el selector del flight poniendo ids de vuelos sin publicar, y vuelos inexistentes. Por último, se probó a hackear el selector de travel Class con valores incorrectos como vacío y “HACKED”, además los valores readOnly se modificaron para ver si permitían el hackeo. | Ninguno |
| Delete | Se probó a acceder al delete de un booking sin publicar, otro publicado y otro inexistente sin estar logueado. Posteriormente se inició sesión con el Customer 2 y se probó acceder al delete de los mismo bookings. Finalmente se inició sesión con el Customer 1 probamos a acceder al delete de un booking publicado. | Ninguno |

**Funcionalidades de Reservas-pasajeros (*BookingRecord*):**

|  |  |  |
| --- | --- | --- |
| **.safe** | **Descripción** | **Bugs encontrados** |
| List | Iniciamos sesión en el sistema con Customer 1, mostramos los detalles de un booking con pasajeros, listamos sus pasajeros, mostramos los detalles de un booking sin pasajeros, listamos sus pasajeros, | Ninguno |
| Create | Iniciamos sesión en el sistema con Customer 1, mostramos los detalles de un booking sin pasajeros, añadimos pasajero vacío, añadimos pasajero, vamos a añadir pasajero otra vez, listamos de nuevo las posibilidades, añadimos un pasajero distinto. | Ninguno |
| Delete | Iniciamos sesión en el sistema con Customer 1, mostramos los detalles de un booking con más de un pasajero, cambiamos show por delete en la url, eliminamos pasajero vacío, eliminamos pasajero, volvemos a elegir para eliminar pasajero, eliminamos pasajero. | Ninguno |

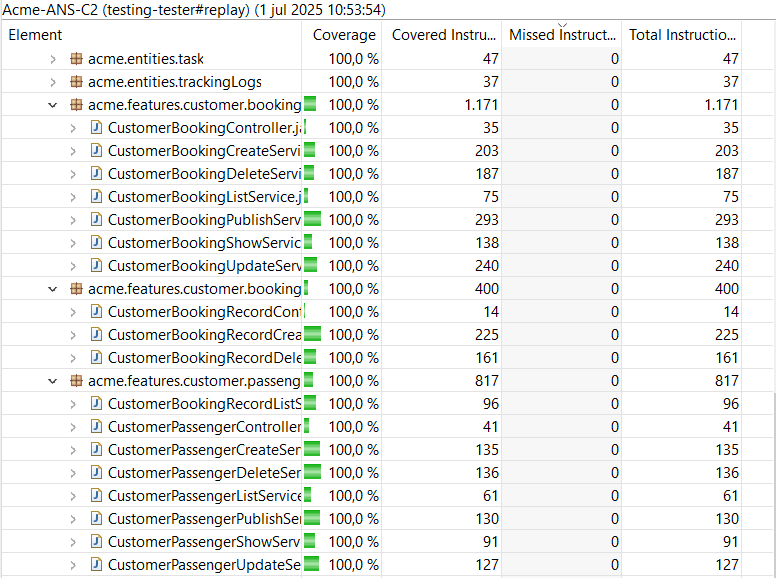
|  |  |  |
| --- | --- | --- |
| **.hack** | **Descripción** | **Bugs encontrados** |
| List | Se probó acceder a ver los pasajeros de un booking existente y otro inexistente sin estar logueado y luego se probó lo mismo con el Customer 2 que no era el creador de dichas bookings. | Daba un error 500 distinto al “Access is not authorised” porque el booking era igual a null. (Se solucionó correctamente) |
| Create | Se probó a añadir pasajeros a un booking sin publicar con pasajeros asociados, a un booking publicado, a un booking que no existe y a un booking de otro Customer, todo ello sin tener sesión iniciada. Posteriormente se hizo lo mismo para otro customer (Customer 2). Por último, se probó con el Customer 1 acceder a la pantalla de añadir pasajeros de un de un booking publicado, se probó a hackear el id y el value con valores inexistentes (999) y se probó a hackear los ids de los pasajeros que aparecían en el selector, con ids incorrectos (999), ids de pasajeros que había creado otro customer y ids de pasajeros que ya estaban asociado al booking. | Ninguno |
| Delete | Se probó a eliminar pasajeros a un booking sin publicar con pasajeros asociados, a un booking publicado, a un booking que no existe y a un booking de otro customer, todo ello sin tener sesión iniciada. Posteriormente se hizo lo mismo para otro customer (Customer 2). Por último, se probó con el Customer 1 acceder a la pantalla de eliminar pasajeros de un de un booking publicado | Ninguno |

**Funcionalidades de Pasajeros (*Passenger*)**

|  |  |  |
| --- | --- | --- |
| **.safe** | **Descripción** | **Bugs encontrados** |
| List | Iniciamos sesión en el sistema con Customer 1 y listamos todos los Passenger. | Ninguno |
| Show | Iniciamos sesión en el sistema con Customer 1, listamos todos los pasajeros y mostramos los detalles de uno de ellos. | Ninguno |
| Create | Iniciamos sesión en el sistema con Customer 1, listamos todos los pasajeros, vamos al formulario de creación, enviamos vacío, empezamos a probar valores erróneos  y luego valores correctos, enviamos el formulario correctamente. | Ninguno |
| Update | Iniciamos sesión en el sistema con Customer 1, listamos todos los pasajeros, mostramos los detalles de uno de ellos, cambiamos show por update en la url, actualizamos con todos los valores vacíos, empezamos a probar valores erróneos  y luego valores correctos, actualizamos el pasajero correctamente. | Ninguno |
| Publish | Iniciamos sesión en el sistema con Customer 1, listamos todos los pasajeros, mostramos los detalles de uno de ellos, cambiamos show por publish en la url, publicamos con todos los valores vacíos, empezamos a probar valores erróneos  y luego valores correctos, publicamos el pasajero correctamente. | Ninguno |
| Delete | Iniciamos sesión en el sistema con Customer 1, listamos todos los pasajeros, mostramos los detalles de uno de ellos, cambiamos show por delete en la url, eliminamos el pasajero. | Ninguno |

|  |  |  |
| --- | --- | --- |
| **.hack** | **Descripción** | **Bugs encontrados** |
| List | Se probó si se podía acceder al listado de passengers sin estar logueado en la aplicación. | Ninguno |
| Show | Se probó acceder al show de un pasajero sin estar logueado. Luego se probó acceder al show de un passenger sin publicar, otro publicado y otro inexistente. Por último, iniciamos sesión con Customer 2 (customer que no era el creador de dicho passengers) y probamos a acceder al show de los mismos passengers mencionados anteriormente. | Ninguno |
| Create | Se probó a acceder al créate de un passenger sin estar logueado. Posteriormente se inició sesión con el Customer 1 y se probó a hackear el id y el version con valores incorrectos (999). | Ninguno |
| Update | Se probó a actualizar un passenger sin publicar, otro publicado y otro inexistente, todo ello sin estar logueado. Posteriormente iniciamos sesión con Customer 2 (customer que no era el creador de dicho passengers) y probamos acceder al update de dichos passengers. Por último, iniciamos sesión con el customer 1 y probamos a acceder al update de un passenger ya publicado. | Ninguno |
| Publish | Se probó a publicar un passenger sin publicar, otro publicado y otro inexistente, todo ello sin estar logueado. Posteriormente iniciamos sesión con Customer 2 (customer que no era el creador de dicho passengers) y probamos acceder al publish de dichos passengers. Por último, iniciamos sesión con el Customer 1 y probamos a acceder al publish de un passenger ya publicado. | Ninguno |
| Delete | Se probó a eliminar un passenger sin publicar, otro publicado y otro inexistente, todo ello sin estar logueado. Posteriormente iniciamos sesión con Customer 2 (customer que no era el creador de dicho passengers) y probamos acceder al delete de dichos passengers. Por último, iniciamos sesión con el Customer 1 y probamos a acceder al delete de un passenger ya publicado. | Ninguno |

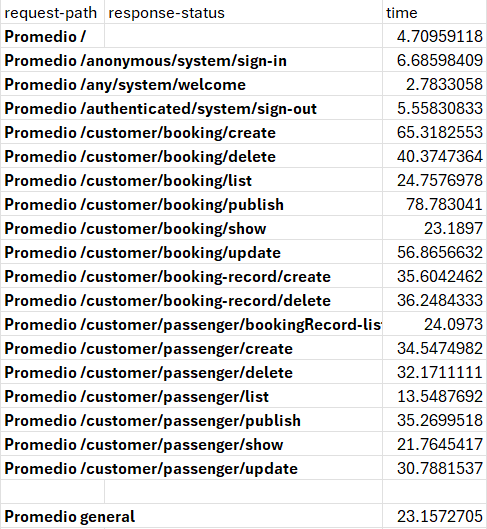
Tras realizar todas las pruebas mencionadas anteriormente, conseguimos llegar al 100% de coverage para todas las features:



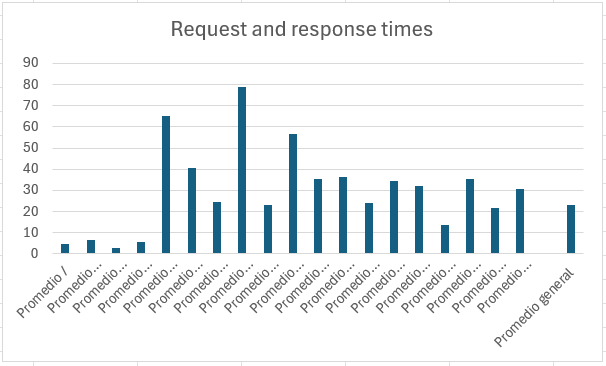
### **Rendimiento de testing:**

Se probaron de dos maneras distintas todas las pruebas mencionadas anteriormente. Primero se hicieron las mismas en un mismo ordenador de tal manera que primero se probó sin índices implementados y luego con índices implementados. Por último, se probaron todas las pruebas, sin índices implementados, en ordenadores distintos.

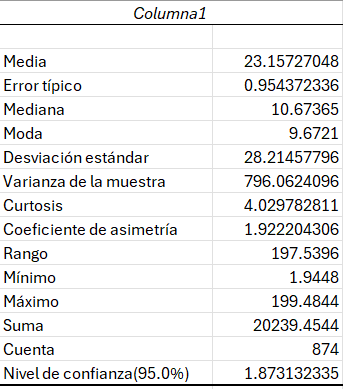
### **Resultados de la primera prueba (sin índices)**:

Se muestran para este caso las siguientes medidas de tiempo (en milisegundos) para cada URI correspondiente a la petición que representa:

A continuación, representamos estos datos de manera gráfica:



Por tanto, de estos datos obtenemos la siguiente estadística:

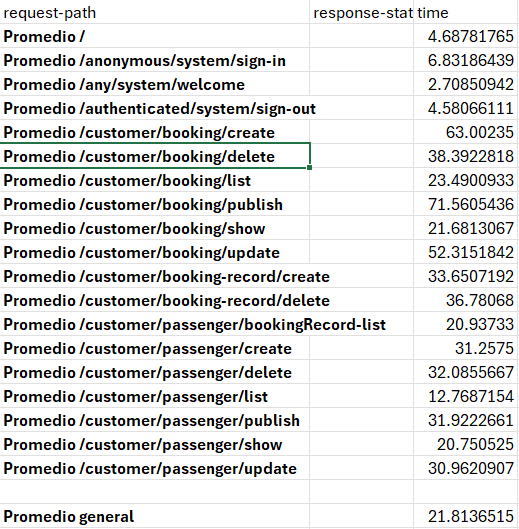


Como se puede observar, obtenemos un nivel de confianza de **1.873132335** y por tanto podemos establecer el siguiente intervalo de confianza:

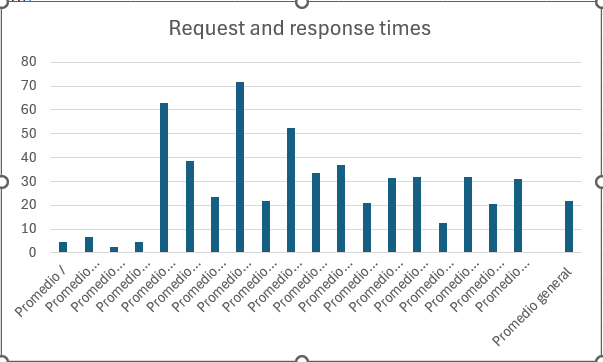


### **Resultados de la primera prueba (con índices):**

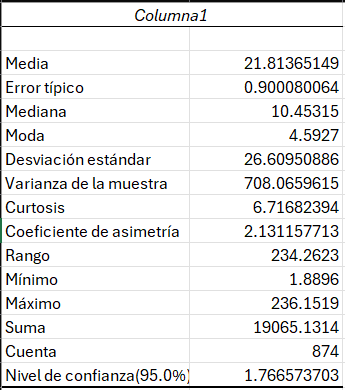
Se muestran para este caso las siguientes medidas de tiempo (en milisegundos) para cada URI, correspondiente a la petición que representa:



A continuación, representamos estos datos de manera gráfica:



Por tanto, de estos datos obtenemos la siguiente estadística:



Como se puede observar, obtenemos un nivel de confianza de **1.766573703** y por tanto podemos establecer el siguiente intervalo de confianza:

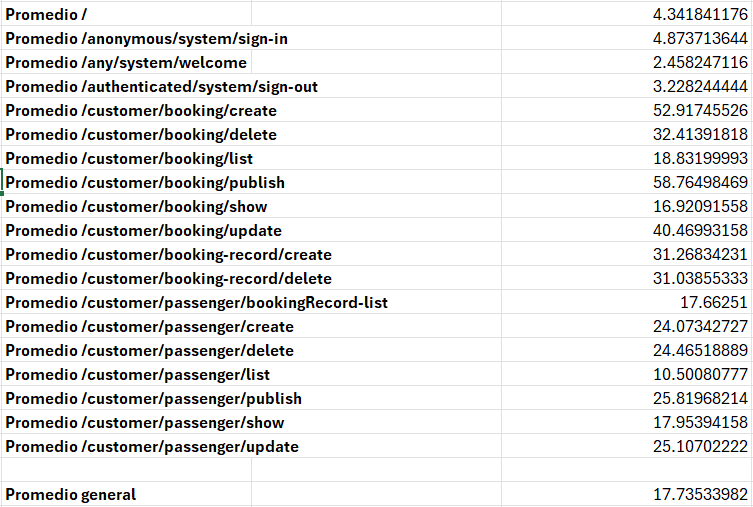


### **Resultados de la segunda prueba (sin índices en ordenador local):**

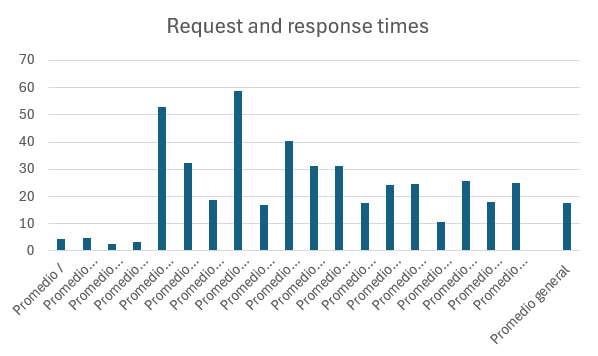
Estos son los mismos resultados que los del apartado 5.1.

### **Resultados de la segunda prueba (sin índices en otro ordenador):**

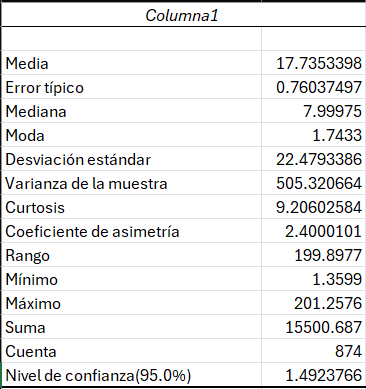
Se muestran para este caso las siguientes medidas de tiempo (en milisegundos) para cada URI correspondiente a la petición que representa:

****

A continuación, representamos estos datos de manera gráfica:

****

Por tanto, de estos datos obtenemos la siguiente estadística:

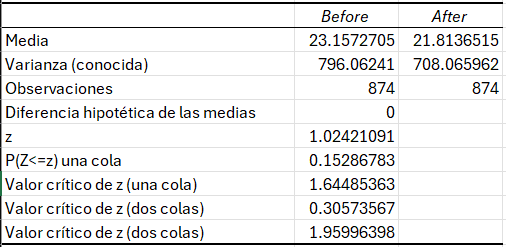
****

Como se puede observar, obtenemos un nivel de confianza de **1.4923766** y por tanto podemos establecer el siguiente intervalo de confianza:



### **Primera pruebaZ (mismo ordenador sin índices – con índices):**

Estos son los resultados obtenidos al contrastar ambas pruebas. Esta prueba recibe el nombre de Prueba Z.

****

Estos son los intervalos de confianza del antes (sin índices):



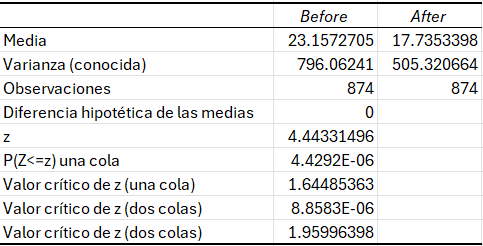
Y estos son los intervalos de confianza del después (con índices):



Podemos observar que el valor de P es menor que 0.5, esto quiere decir que los cambios realizados son significantes, es decir, tienen un impacto importante en el rendimiento de mi aplicación. Por tanto, podemos deducir que el sistema, en cuanto a rendimiento, se comporta mejor con índices implementados que sin ellos.

### **Segunda pruebaZ (distinto ordenador sin índices)**

Estos son los resultados obtenidos al contrastar ambas pruebas:

****

Estos son los intervalos de confianza del antes (ordenador 1 sin índices):

****

Y estos son los intervalos de confianza del después (ordenador 2 con índices):

****

Podemos observar que el valor de P es mucho menor que 0.5, esto quiere decir que la diferencia en cuanto al rendimiento entre un ordenador y otro es suficientemente significativa y por tanto podemos comparar ambos valores. Se demuestra a través de la media que el ordenador 2 (After) es más potente que el ordenador 1 (Before).

### **Conclusiones**:

El testing es una fase clave del desarrollo, ya que asegura que el sistema cumpla con los requisitos, funcione correctamente y brinde buena experiencia al usuario. Sin embargo, no basta con validar funcionalidades y rendimiento, también es vital probar la seguridad para detectar vulnerabilidades y proteger los datos más vulnerables. Por ello, el testing debe entenderse como un proceso integral que combina calidad, eficiencia y protección, garantizando un producto robusto y confiable.

Personalmente esta fase de testing ha sido una experiencia muy enriquecedora, me ha hecho entender situaciones muy importantes que debo contemplar a la hora de abordar un proyecto de forma profesional. En ocasiones ha sido un completo desafío, pero gracias a los contenidos dados en clase, sesiones de *Follow Up* y por supuesto a la comunicación y ayuda de mis compañeros, hemos podido completar esta última fase del proyecto. De nuevo, como ha sido costumbre a lo largo del desarrollo de Acme-ANS, el trabajo en equipo ha sido la clave para lograr completar todos los requisitos esperados.

### **Bibliografía**:

Intencionalmente en blanco.