**Testing Report Group**

**C2.019**

| **Grupo** | C2.019 |
| --- | --- |
| **Repositorio** | <https://github.com/adolfoborrego/Acme-ANS> |
| **Student #2** | **ID**: 77873179D  **UVUS**: SSK0456  **Nombre**: Martínez Díaz, Ignacio  **Roles**: Developer, Analyst, Tester |
| **Student #4** | **ID**: 52077055H  **UVUS**: TCP2748  **Nombre**: Sánchez Carmona, Germán  **Roles**: Project Manager, Developer , Analyst, Tester |
| **Student #5** | **ID**: 54794337B  **UVUS**: CFV7375  **Nombre**: Regidor García, Miguel  **Roles**: Operator, Developer , Analyst, Tester |

**Índice**

[**1. Introducción 2**](#_uaak4lrj701f)

[1.1. Propósito del documento 2](#_vhw0ph3g883v)

[**2. Functional Testing 3**](#_2o7winp4n81o)

[2.1. Introducción 3](#_u0mptw7qxets)

[2.2. Casos de prueba 4](#_miyy5k1dkxh7)

[2.3. Cobertura de las pruebas 5](#_fx9tgbgyal9z)

[2.4. Conclusiones 5](#_l9jap5oujti4)

[**3. Performance Testing 6**](#_e8k0vhxse4bq)

[3.1. Introducción 6](#_okqbw7wqfat6)

[3.2. Gráficos de eficiencia medios 7](#_waevn69g30k4)

[3.3. Estadísticas descriptivas 8](#_tjedc5gtr4xl)

[3.4. Hipótesis y conclusiones 9](#_tzzfw1l1kz1z)

[**4. Historial de versiones 10**](#_ber4r6t8b5ec)

[**5. Bibliografía 11**](#_u8bxkmmvpl1i)

# **1. Introducción**

## 1.1. Propósito del documento

El propósito de este documento es presentar de forma estructurada y rigurosa los resultados obtenidos durante el proceso de pruebas del sistema desarrollado por el equipo. El informe tiene como objetivo principal verificar que las funcionalidades implementadas cumplen correctamente con los requisitos definidos y que el sistema ofrece un comportamiento estable, seguro y eficiente tanto a nivel funcional como de rendimiento.

A través de este documento se detalla la ejecución de pruebas funcionales, diferenciando entre versiones inseguras (.hack) y versiones protegidas (.safe), así como el análisis del rendimiento del sistema en diferentes condiciones, incluyendo comparativas estadísticas e intervalos de confianza. Este análisis permite no solo identificar errores o vulnerabilidades, sino también justificar decisiones técnicas tomadas durante el desarrollo para mejorar la calidad final del producto.

# **2. Functional Testing**

## 2.1. Introducción

En este apartado se documenta la metodología seguida para realizar las pruebas funcionales (functional testing) de las distintas características implementadas. El objetivo principal de este tipo de pruebas es comprobar que las funcionalidades del sistema devuelven los resultados esperados, tanto en condiciones normales como anómalas, garantizando así la calidad del software desde el punto de vista del usuario final.

Siguiendo la metodología formal estudiada, se han diseñado y ejecutado casos de prueba positivos, negativos (.safe) y de hacking (.hack). Cada uno de estos tipos de pruebas ha sido elaborado respetando los principios de repetibilidad, control de datos y cobertura. Se ha prestado especial atención a la verificación de formularios de edición, la visualización de datos en listados y formularios, la gestión de entradas inválidas y la resistencia del sistema ante intentos de uso indebido.

## 2.2. Casos de prueba

A continuación se detallan los casos de prueba realizados sobre la entidad *Airport*. La tabla incluye tanto pruebas positivas como pruebas de hacking, con su correspondiente descripción y una valoración de su eficacia.

| **Caso de prueba** | **Descripción** | **Eficacia** |
| --- | --- | --- |
| list.safe | Comprueba que el listado funciona correctamente. | No se han detectado errores. |
| list.hack | Comprueba que no se puedan hacer GET hacking en el listado. | No se han detectado errores. |
| show.safe | Comprueba que se muestran las  propiedades de airport correctamente. | No se han detectado errores. |
| show.hack | Comprueba que el airport no sea nulo, e intenta acceder desde otro realm. | No se han detectado errores. |
| create.safe | Comprueba que se pueda crear  correctamente un airport, contemplando así las validaciones de  cada propiedad. | No se han detectado errores. |
| create.hack | Comprueba que el que esté creando  el airport sea un administrator, y que no se introduzca un operationalScope inválido. | No se han detectado errores. |
| update.safe | Comprueba que se puedan actualizar  los datos de un airport, teniendo en  cuenta las validaciones de dichos | No se han detectado errores. |
| update.hack | Comprueba que el que esté  actualizando el airport sea un  administrator, y que no se introduzca un operationalScope inválido. | No se han detectado errores. |

## 2.3. Cobertura de las pruebas

La siguiente tabla muestra la cobertura alcanzada por el paquete *Airport* asociado a las funcionalidades desarrolladas para *Administrator*. Esta cobertura se ha obtenido a partir de la ejecución de los casos de prueba descritos.

| **Paquete** | **Cobertura** |
| --- | --- |
| administrator.airport | 94.2 % |

A continuación se presenta la cobertura obtenida por servicio dentro del paquete administrator.airport, como resultado directo de los casos de prueba realizados sobre la entidad *Airport*.

| **Servicio** | **Cobertura** |
| --- | --- |
| AdministratorAirportListService | 94.6 % |
| AdministratorAirportShowService | 96.8 % |
| AdministratorAirportCreateService | 93.5 % |
| AdministratorAirportUpdateService | 93.2 % |

## 2.4. Conclusiones

La cobertura de pruebas para el paquete Airport ha sido satisfactoria, alcanzando un 94.2% global, con cada uno de los servicios probados superando el 90% de cobertura. Los casos de prueba ejecutados, tanto seguros como de tipo hack, no detectaron fallos, lo que indica un comportamiento robusto del sistema frente a operaciones válidas y accesos no autorizados. Esto demuestra que las funcionalidades implementadas para la gestión de aeropuertos son estables y seguras bajo los escenarios evaluados.

# **3. Performance Testing**

## 3.1. Introducción

El propósito de este apartado es analizar el rendimiento del sistema mediante pruebas de tipo performance testing. A diferencia de las pruebas funcionales, cuyo objetivo principal es verificar el comportamiento correcto del sistema, las pruebas de rendimiento se centran en medir el tiempo de respuesta de las distintas funcionalidades bajo condiciones controladas.

Para llevar a cabo este análisis, se han reutilizado los casos de prueba funcionales ya existentes, reproduciéndolos (replay) en dos equipos portátiles diferentes. De este modo, se ha obtenido un conjunto de datos reales de ejecución a partir de los cuales se ha podido evaluar la eficiencia del sistema y comparar el rendimiento relativo de ambos entornos.

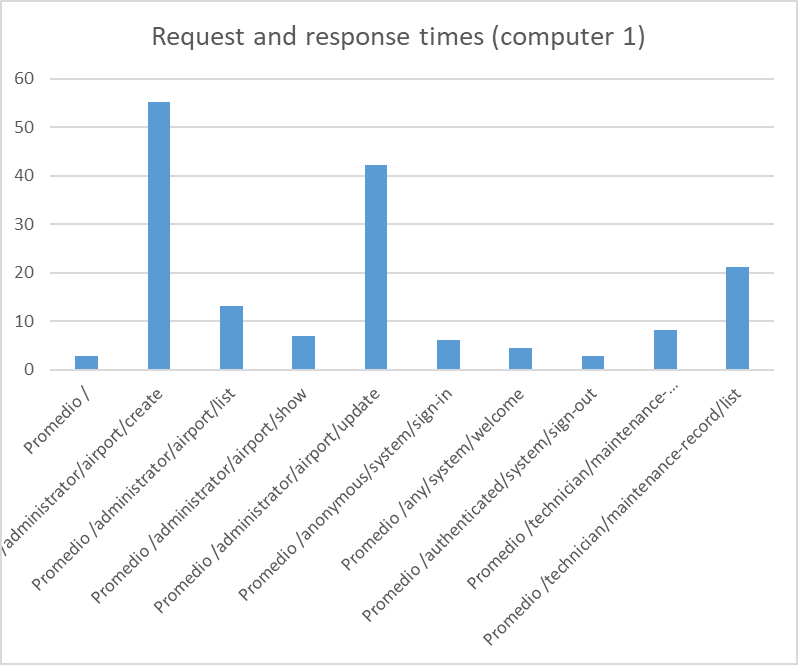
El análisis se ha realizado siguiendo la metodología propuesta, que incluye el cálculo de intervalos de confianza para el tiempo medio de respuesta y la aplicación de una prueba Z para dos muestras independientes, con el objetivo de determinar si las diferencias observadas entre ambos portátiles son estadísticamente significativas.

En las siguientes secciones se presentan los resultados obtenidos, tanto en forma de gráficos de eficiencia media como de análisis estadísticos que permiten extraer conclusiones fundamentadas sobre el rendimiento de la aplicación.

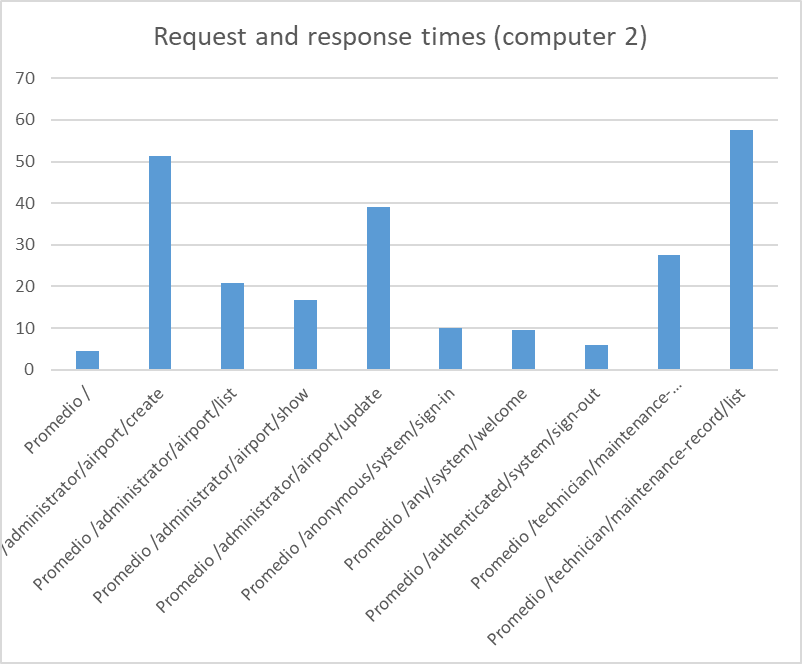
## 3.2. Gráficos de eficiencia medios

Con el objetivo de analizar visualmente el rendimiento del sistema, se han generado gráficos de eficiencia que muestran el tiempo medio de respuesta por funcionalidad (feature).

Estos gráficos permiten identificar de forma clara las funcionalidades que presentan un mayor coste computacional, denominadas MIR (Most Inefficient Request), las cuales podrían convertirse en cuellos de botella en un entorno real. En ambos casos, las funcionalidades create y update de airport.



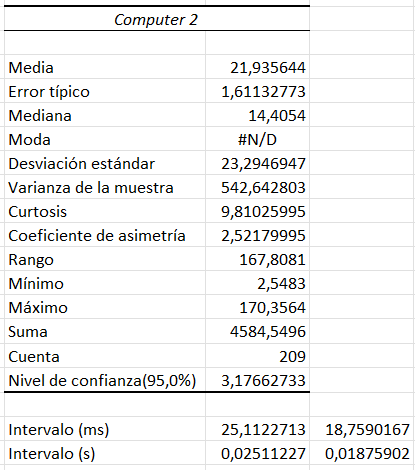
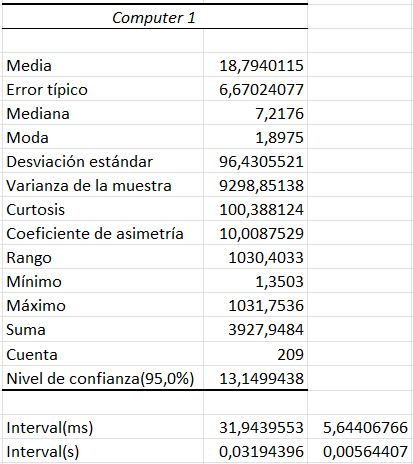
*Computer 1*



*Computer 2*

## 3.3. Estadísticas descriptivas

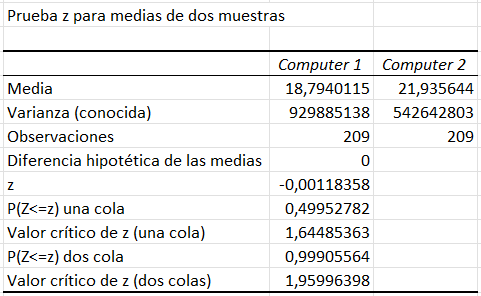
Como paso previo al contraste de hipótesis, se ha realizado un análisis estadístico descriptivo de los tiempos de respuesta registrados al ejecutar los casos de prueba en ambos portátiles. Este análisis permite obtener una visión general del comportamiento del sistema en cada equipo y calcular el intervalo de confianza para el tiempo medio de respuesta.



Ambos portátiles cumplen con el supuesto de que la media de respuesta se mantiene muy por debajo del segundo (1 s), cumpliendo así el requisito de rendimiento utilizado como referencia.

## 3.4. Hipótesis y conclusiones

Tras obtener los intervalos de confianza y las estadísticas descriptivas de ambos portátiles, se ha realizado un contraste de hipótesis mediante una prueba Z para dos muestras independientes, con el objetivo de determinar si la diferencia entre los tiempos medios de respuesta es estadísticamente significativa. La prueba Z se ha aplicado sobre los conjuntos de datos recogidos en Computer 1 y Computer 2, considerando un nivel de significación del 5 % (α = 0.05).



Dado que el valor de P(Z<=z) dos colas se encuentra en el intervalo (α,1], podemos afirmar que no existen grandes diferencias en el rendimiento entre ambos portátiles.

# **4. Historial de versiones**

| **Versión** | **Fecha** | **Descripción** |
| --- | --- | --- |
| 0.0 | 25/05/2025 | Versión inicial. |
| 1.0 | 26/05/2025 | Versión final. |
| 2.0 | 02/07/2025 | Actualización del documento con los resultados de los tests generados para la segunda convocatoria. |

# **5. Bibliografía**

Intentionally Blank.