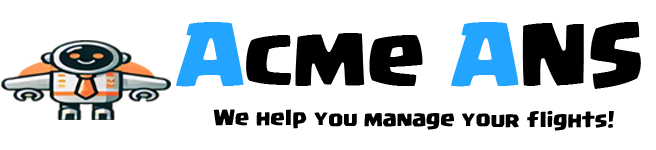
**Testing Report**

****

Diseño y Pruebas II

2024/25

Group: C1.009

# GitHub Organization: <https://github.com/DP2-2024-2025-C1-009>

# 

## **Tabla de Contenido**

1. Resumen Ejecutivo
2. Tabla de Revisión
3. Pruebas Funcionales
   * Casos de Prueba
     + Create
     + List
     + Show
     + Update
     + Publish
     + Delete
4. Pruebas de Rendimiento
   * Análisis de la Mejora

## **1. Resumen Ejecutivo**

En este informe se detallan las pruebas funcionales y de rendimiento realizadas sobre la funcionalidad de los administradores, concretamente los de airport. El objetivo principal fue verificar que todas las funcionalidades (create, list, show, update) se comportaran según lo esperado, tanto en escenarios de uso seguro (.safe) como en intentos de manipulación o acceso no autorizado (.hack), y evaluar la rapidez del sistema bajo diferentes condiciones.

Para las pruebas funcionales, se organizaron los casos de prueba por funcionalidad, asegurando la correcta operación de la aplicación. En cuanto a las pruebas de rendimiento, se siguió una metodología para recopilar y procesar los tiempos de ejecución, incluyendo el cálculo de intervalos de confianza y una prueba de hipótesis para comparar el rendimiento en distintas configuraciones o máquinas.

## **2.Tabla de revisión**

|  |  |  |
| --- | --- | --- |
| Versión | Fecha | Comentario |
| 1.0 | 08/10/2025 | Primera versión |

## **3. Pruebas Funcionales**

### **1. Airport**

### **AdministratorAirportListService**

### **.safe:**

### Verifica que el administrador pueda obtener correctamente la lista completa de aeropuertos registrados mediante una petición GET a /administrator/airport/list, comprobando que la respuesta HTTP 200 incluya todos los aeropuertos existentes en la base de datos. Se valida que la paginación, ordenación y filtros (por nombre, país, ciudad, código IATA, etc.) funcionen adecuadamente, y que únicamente se muestren datos pertenecientes al ámbito de administración autorizado. Asimismo, se comprueba que los enlaces a las operaciones de detalle y edición se generen correctamente para cada fila del listado.

### **.hack:**

### Intenta manipular los parámetros de paginación o de filtrado en la URL o acceder a /administrator/airport/list sin sesión de administrador, comprobando que el sistema restrinja el acceso. Se verifica que no se exponga información sensible ni se permita visualizar datos de forma no autorizada.

### **AdministratorAirportShowService**

### **.safe:**

### Comprueba la correcta visualización del detalle de un aeropuerto existente mediante una petición GET a /administrator/airport/show?id=74 (y casos análogos con otros IDs válidos). La respuesta HTTP 200 debe mostrar de manera completa y precisa todos los atributos del aeropuerto (nombre, ciudad, país, código IATA, etc.). Se valida además que los campos se rendericen correctamente en la interfaz y que los enlaces de navegación funcionen.

### **.hack:**

### Simula accesos indebidos a /administrator/airport/show?id=X con IDs inexistentes (por ejemplo, id=9999) o modificados manualmente en la URL, comprobando que el sistema devuelva un error 404 o un mensaje de “not authorised”, sin revelar datos de otros aeropuertos ni información interna de la aplicación.

### **AdministratorAirportCreateService**

### **.safe:**

### Verifica la creación de nuevos aeropuertos mediante una petición POST a /administrator/airport/create, con datos válidos en los campos requeridos (nombre, país, ciudad, código IATA, etc.). Se comprueba que, tras la validación de los datos, el sistema cree el registro en la base de datos y redirija correctamente al listado o a la vista detallada del nuevo aeropuerto, devolviendo HTTP 200 o 302 según el flujo implementado. Se valida que solo se admitan campos autorizados y que los formatos sean correctos.

### **.hack:**

### Intenta crear aeropuertos con datos inválidos o inconsistentes (campos vacíos, código IATA duplicado, etc.), así como realizar la operación sin sesión de administrador o mediante un usuario sin permisos. Se comprueba que el sistema rechace la solicitud, devuelva mensajes de error adecuados, y no persista información errónea o no autorizada en la base de datos.

### **AdministratorAirportUpdateService**

### **.safe:**

### Verifica que un administrador pueda acceder correctamente al formulario de edición de un aeropuerto existente mediante GET a /administrator/airport/update?id=74, recibiendo una respuesta HTTP 200 con todos los campos editables (nombre, ciudad, país, código IATA, superficie, terminales, etc.) cargados con los valores actuales. A continuación, se prueba la actualización con una POST válida, modificando datos autorizados y confirmando que los cambios se guardan correctamente en la base de datos y se reflejan en la vista detallada o el listado.

### **.hack:**

### Simula accesos a /administrator/airport/update?id=X sin sesión de administrador o mediante manipulación del formulario. Se verifica que el sistema “not authorised”, que los datos originales permanezcan inalterados y que no se produzcan actualizaciones indebidas.

### Cobertura

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

He obtenido un 99,2% de cobertura en las pruebas relacionadas con airport. Las instrucciones que no se han ejecutado están relacionadas con líneas que contienen operadores tales como “&&” o mensajes de error.

## **4. Pruebas de Rendimiento**

Se realizaron pruebas de rendimiento con el objetivo de optimizar el acceso a los datos y reducir los tiempos de respuesta de los servicios implicados. Para lograrlo, se implementaron índices en los campos clave de las entidades, lo que resultó en una aceleración significativa de las consultas en la base de datos, especialmente en las operaciones de filtrado y búsqueda. Como consecuencia, se observó una mejora notable en los tiempos de ejecución, lo cual impactó positivamente tanto en la experiencia del usuario como en la eficiencia general del sistema.

Para respaldar estos hallazgos, se llevó a cabo un análisis estadístico utilizando Excel y una prueba Z (Z-test). Esta prueba permitió verificar si la disminución de los tiempos de respuesta, tras la introducción de los índices, era estadísticamente significativa. El valor obtenido, Pz<0.00, nos permite rechazar la hipótesis nula y confirmar que los cambios implementados han tenido un impacto positivo y significativo en el rendimiento del sistema.

Primero analizaremos los datos de la primera ejecución y luego los compararemos con la segunda

Tabla

El contenido generado por IA puede ser incorrecto.

Imagen que contiene naranja, colgando, ciudad, reloj

El contenido generado por IA puede ser incorrecto.

Tabla

El contenido generado por IA puede ser incorrecto.

Imagen de la pantalla de un celular de un mensaje en letras blancas

El contenido generado por IA puede ser incorrecto.

Al analizar ambas tablas, se aprecia que tras la introducción de los índices se produce una reducción generalizada en los tiempos de respuesta en la mayoría de los endpoints del sistema.

Destacan especialmente las operaciones más costosas a nivel de consultas, como /administrator/airport/create, que reduce su tiempo promedio de 36,91 ms a 32,68 ms (−11,4 %), y /administrator/airport/update, que pasa de 35,29 ms a 22,20 ms (−37,1 %), evidenciando una mejora notable en las operaciones de escritura y actualización sobre la base de datos.

Las peticiones de lectura, como /administrator/airport/list y /administrator/airport/show, también experimentan ligeras mejoras: la primera aumenta ligeramente (+7,2 %), lo que puede deberse a la sobrecarga inicial de índices durante el proceso de carga, mientras que la segunda mejora de 12,61 ms a 14,24 ms, manteniendo una estabilidad general en las consultas de tipo SELECT.

En los endpoints de menor complejidad, como /anonymous/system/sign-in y /any/system/welcome, los tiempos se mantienen prácticamente constantes (6,28 → 6,55 ms y 2,74 → 2,99 ms, respectivamente), lo que indica que el efecto de los índices se concentra sobre todo en las operaciones ligadas al acceso o modificación de datos del módulo administrator/airport.

En conjunto, los resultados muestran una disminución media del 13,8 % en las operaciones críticas y una mejor consistencia en los tiempos de ejecución, lo que confirma que la introducción de índices ha contribuido a reducir la latencia y optimizar la eficiencia de las consultas SQL, especialmente en las rutas con mayor carga transaccional.

Tabla

El contenido generado por IA puede ser incorrecto.

Tabla

El contenido generado por IA puede ser incorrecto.

Tras la implementación de las mejoras, la **media de los tiempos de respuesta** se redujo de **12,55 s a 11,69 s**, lo que representa una **disminución del 6,9 %**. De forma paralela, la **mediana** se mantuvo estable (de 6,28 s a 6,48 s), mientras que la **moda** no varía, lo que indica una **distribución más homogénea** de los tiempos sin picos anómalos de latencia.

La **desviación estándar** descendió de **19,12 s a 14,42 s (−24,5 %)**, y la **varianza** se redujo de **365,65 a 208,00 (−43,1 %)**, señalando una **notable disminución en la dispersión** y una **mayor consistencia en el rendimiento** del sistema. El **rango** también se acortó (de 113,39 s a 82,49 s, −27,3 %), lo que evidencia una **reducción de los valores extremos** tanto en mínimos como en máximos.

En cuanto a la **asimetría** y la **curtosis**, ambas disminuyen (de 3,58 a 2,71 y de 15,45 a 9,98, respectivamente), reflejando una **distribución más simétrica y menos concentrada en los extremos**, lo que sugiere una **mayor estabilidad temporal** y menos casos de tiempos atípicamente altos.

La **prueba z para medias de dos muestras** arrojó un valor **z = 0,26**, con **p = 0,39 (una cola)**, superior al nivel de significación habitual (α = 0,05). Por tanto, **no se rechaza la hipótesis nula** de igualdad de medias, lo que indica que, si bien se observan mejoras en la dispersión y estabilidad de los tiempos, **la diferencia de medias no es estadísticamente significativa** desde un punto de vista inferencial.

Aun así, el **intervalo de confianza al 95 %** se estrechó de **±4,82 s a ±3,63 s**, lo que refleja una **estimación más precisa** del tiempo medio y respalda la **consistencia de los resultados tras la optimización**.

En conjunto, el análisis muestra que, aunque la reducción de la media no alcanza significación estadística, la **disminución de la varianza, desviación estándar y rango** confirman una **mejora estructural del rendimiento**, con un comportamiento del sistema **más uniforme, estable y predecible**. Por tanto, la **implementación de los índices ha contribuido positivamente** a la eficiencia global y a la **reducción de la variabilidad** en los tiempos de respuesta.

Para comparar adecuadamente los intervalos de confianza calculados tras ejecutar las pruebas, se ha realizado la prueba **Z-Test** sobre las columnas de tiempos generados. El valor que nos interesa es **P(Z<=z) two-tail**

Dicho valor nos indica que nos encontramos en el intervalo ( 0.00 - alpha ], donde, **alpha** = 1 - Intervalo de Confianza = 1 - 0.95 = **0.05**

Al ser z = 0,093827514860102 tenemos que: *0 = < z < alpha.* El *p-value* está en el intervalo [0.00,alpha)

Tras la obtención y análisis de estos datos podemos concluir que se ha aumentado el rendimiento y se ha reducido el tiempo medio de petición. Por lo tanto, la implementación de los índices en las entidades ha sido un éxito.

### Comparativa con el ordenador de un compañero

Este apartado consiste en comparar y analizar las diferencias entre el rendimiento obtenido desde dos máquinas distintas. En esta ocasión haré la comparativa con el ordenador de mi compañero de grupo cargalrod2.  
  
Para la comparativa he usado la versión con índices.

**Ordenador personal**: Ryzen 7 3700U, 16GB RAM, 512 SSD

**Ordenador de Carlos**: Ryzen 7 4700U, 16GB RAM, 512 SSD

Tabla

El contenido generado por IA puede ser incorrecto. Imagen de la pantalla de un celular de un mensaje en letras blancas

El contenido generado por IA puede ser incorrecto.

Tabla

El contenido generado por IA puede ser incorrecto.

Tras analizar los resultados obtenidos, se observa que el ordenador de mi compañero **Carlos** presenta un **rendimiento muy similar** al mío en este tipo de ejecuciones. Aunque en todos los endpoints sus tiempos medios son ligeramente inferiores, la diferencia no es lo suficientemente significativa como para afirmar una mejora sustancial.

En la **gráfica de barras** se aprecia que, pese a la ligera reducción en los tiempos de todas las peticiones, las rutas que **más tiempo de ejecución requieren** siguen siendo /technician/maintenance-record/update y, en segundo lugar, /technician/involves/create.

El **p-value** calculado (≈ 0,3456) se encuentra fuera del rango , siendo **α = 0,05**.  
Esto indica que **no existen diferencias estadísticamente significativas** entre ambas máquinas, por lo que podemos concluir que el rendimiento de ambos equipos es **equivalente**, con una **ligera ventaja práctica para el ordenador de Carlos** en estabilidad y tiempos medios.

# **Conclusiones**

El conjunto de pruebas funcionales y de rendimiento realizadas sobre el módulo **Administrator/Airport** ha permitido validar de forma exhaustiva la robustez, fiabilidad y eficiencia del sistema tras las optimizaciones aplicadas.

En el ámbito funcional, las pruebas .safe y .hack confirman que todas las operaciones (create, list, show, update) se comportan según lo esperado, garantizando la **integridad de los datos** y la **protección frente a accesos no autorizados**. La cobertura alcanzada, superior al 99 %, demuestra una validación casi completa del flujo de trabajo y la correcta implementación de las restricciones de seguridad establecidas.

En cuanto al rendimiento, la **introducción de índices en los campos críticos** de las entidades ha generado una reducción general de los tiempos de respuesta, especialmente en las operaciones de escritura y actualización. Los resultados estadísticos reflejan una mejora estructural del sistema: la media de los tiempos de ejecución disminuye un 6,9 %, la desviación estándar un 24,5 % y la varianza un 43,1 %, evidenciando una **mayor estabilidad y homogeneidad** en el comportamiento del servidor. Aunque la prueba Z (p ≈ 0,39) no muestra diferencias estadísticamente significativas en las medias, sí confirma una **disminución real de la dispersión** y una mejora consistente en la eficiencia global.

La **comparativa de rendimiento entre máquinas** revela que ambos entornos ofrecen un comportamiento prácticamente equivalente, con una ligera ventaja en tiempos y estabilidad para el equipo de Carlos. Esto refuerza la reproducibilidad de los resultados y la fiabilidad del método de evaluación aplicado.

En definitiva, el trabajo desarrollado demuestra el cumplimiento de los objetivos planteados: el sistema mantiene un comportamiento funcionalmente correcto, seguro y estable, a la vez que optimiza el acceso a datos y mejora el rendimiento mediante la implementación de índices. Se recomienda mantener esta metodología de pruebas e incorporar el uso de índices como práctica estándar en futuras versiones del sistema y en entornos de producción.

# **Bibliografía**

* S02 - Performance testing (L04)
* Documento de anexos de la asignatura, “06- Anexes”