

# Universidad de Sevilla

Escuela Técnica Superior de Ingeniería Informática

## Testing Report D04 – gerojegar






Grado en Ingeniería Informática – Ingeniería del Software


Diseño y Pruebas II

Curso 2024 – 2025

Fecha	Versión
25-04-2025	V1.0

Repositorio de Github: <https://github.com/cesmarvan/Acme-ANS-D04.git>

Grupo de prácticas: C1.025			
Fotografía	Autor	Rol	Descripción del rol
	González Rodríguez, Joaquín – 77940208Q UVUS: FYK3492 joagonrod@alum.us.es	Developer, manager, analista	Desarrollar y probar features
	Mantecón Rodríguez, Alejandro – 54223206Q UVUS: YRM2963 alemanrod@alum.us.es	Developer, operator, analista	Desarrollar y probar features
	Martínez Van der Looven, César – 77214048N UVUS: NMH6684 cesmarvan@alum.us.es	Developer, tester, operator	Desarrollar y probar features

	<p>Ojeda Garrido, Germán – 29552532Q UVUS: gerojegar gerojegar@alum.us.es</p>	<p>Developer, tester, operator</p>	<p>Desarrollar y probar features</p>
-----------------------------------------------------------------------------------	---------------------------------------------------------------------------------------	----------------------------------------	------------------------------------------

# Índice

1.	Resumen ejecutivo.....	4
2.	Tabla de revisión .....	5
3.	Introducción .....	6
4.	Pruebas sobre Airport.....	7
5.	Cobertura del testing.....	8
6.	Análisis de rendimiento .....	9
1.	Análisis de las graficas con tiempo promedio .....	9
2.	Análisis de los resultados del análisis de estadística descriptiva.....	10
3.	Estudio del Z-Test.....	12
7.	Conclusiones.....	13
8.	Bibliografía .....	14

# 1. Resumen ejecutivo

Con este documento, se pretende analizar los resultados de haber realizado testing para las features de Airport, pertenecientes al rol Administrator. Se estudiarán los resultados obtenidos para asegurar la calidad requerida.

## 2. Tabla de revisión

Número de revisión	Fecha	Descripción de la revisión
1.0	25-04-2025	Creación del documento

### 3. Introducción

En este documento, se pretende analizar los resultados de haber realizado tests requeridos para el entregable D04, con el fin de determinar como de eficiente y cubierto está el código elaborado.

En este testing de la parte grupal, se han estudiado la entidad *Airport*, para asegurar un correcto funcionamiento de dicha entidad y las funcionalidades que la involucra.

Para lograr este objetivo, tal y como se explica en las distintas diapositivas de clase, se ha usado el *testing-tester#record* para grabar todas las peticiones que se hacen durante el testing que se realiza mientras está lanzada dicha herramienta.

El criterio seguido para nombrar los ficheros de logs resultantes de utilizar esta herramienta de testing fue el siguiente:

- Testing positivo y negativo: feature.safe.
- Testing para casos de hacking: feature .hack.

## 4. Pruebas sobre Airport

Pruebas realizadas sobre Airport:

- LIST: se ha probado como funciona la feature *AdministratorAirportListService.java*, en la que se ha probado que un *Administrator* puede acceder a la lista de *Airports*, y un usuario que no es *Adminstrator* le salta una vista de *Unauthorized* al intentar acceder a la url asociada.
- SHOW: se ha probado como funciona la feature *AdministratorAirportShowService.java*, en la que se ha probado que un *Administrator* puede acceder a los detalles de un aeropuerto, pero otro tipo de usuario no puedan acceder a dichos detalles.
- CREATE/UPDATE: se ha probado como funcionan las funcionalidades de *AdministratorAirportCreateService.java* y *AdministratorAirportUpdateService.java*, en los cuales se han probado peticiones legales tanto positivas, realizando la función correctamente, como negativas, en las que han saltado los errores de validación pertinentes.  
Estás restricciones para los distintos atributos se han declarado tanto en la función *validate* del *service* de cada *feature* así como, aquellas que sean comunes a los tres servicios y deban también comprobar que los datos al cargarlos desde un csv, en el validador de *Flight*.  
Se ha comprobado que los atributos no pueden ser nulos, o, aquellos que lo requieran, cumplan el formato requerido así como la extensión mínima y máxima de caracteres. Además, se ha comprobado que no permite crear o actualizar un aeropuerto con un *IataCode* ya existente, así como que el campo de confirmación dee estar seleccionado para poder realizar la acción.  
Para las peticiones ilegales y/o hacking, se ha probado que para cada una de ellas se devuelve un *Panic* con mensaje *Unauthorized*, invalidando dichas acciones ilegales, entre las que se encuentran realizar estas acciones sin estar autenticado o con un rol inválido, así como intentar alterar el *Id* del formulario en el caso de la cración, impidiendo que este sea distinto de 0.

## 5. Cobertura del testing

Tras grabar todos los archivos .safe y .hack necesarios, se ha realizado un testing-tester#replay con la opción de Coverage, para ver que todos los test pasan satisfactoriamente y comprobar que la cobertura de código es del 100.

▼	acme.features.administrator.airport		100,0 %	555	0	555
>	AdministratorAirportController.java		100,0 %	24	0	24
>	AdministratorAirportCreateService.java		100,0 %	198	0	198
>	AdministratorAirportListService.java		100,0 %	62	0	62
>	AdministratorAirportShowService.java		100,0 %	97	0	97
>	AdministratorAirportUpdateService.java		100,0 %	174	0	174

*Ilustración 1: Cobertura obtenida para Airport*



## 6. Análisis de rendimiento

Para llevar a cabo el análisis de rendimiento, se analizará el `tester.trace` generado al hacer el `tester-testing#replay` con los archivos `.safe` y `.hack` de Airport guardados en la carpeta `administrator/airport` desde dos PCs diferentes.

Para analizar los resultados, se tratarán los datos tal y como se indican en la clase de teoría.

### 1. Análisis de las graficas con tiempo promedio

Primero se mostrarán las gráficas del tiempo promedio para cada *feature* analizada en los distintos PCs que se han usado para el estudio.

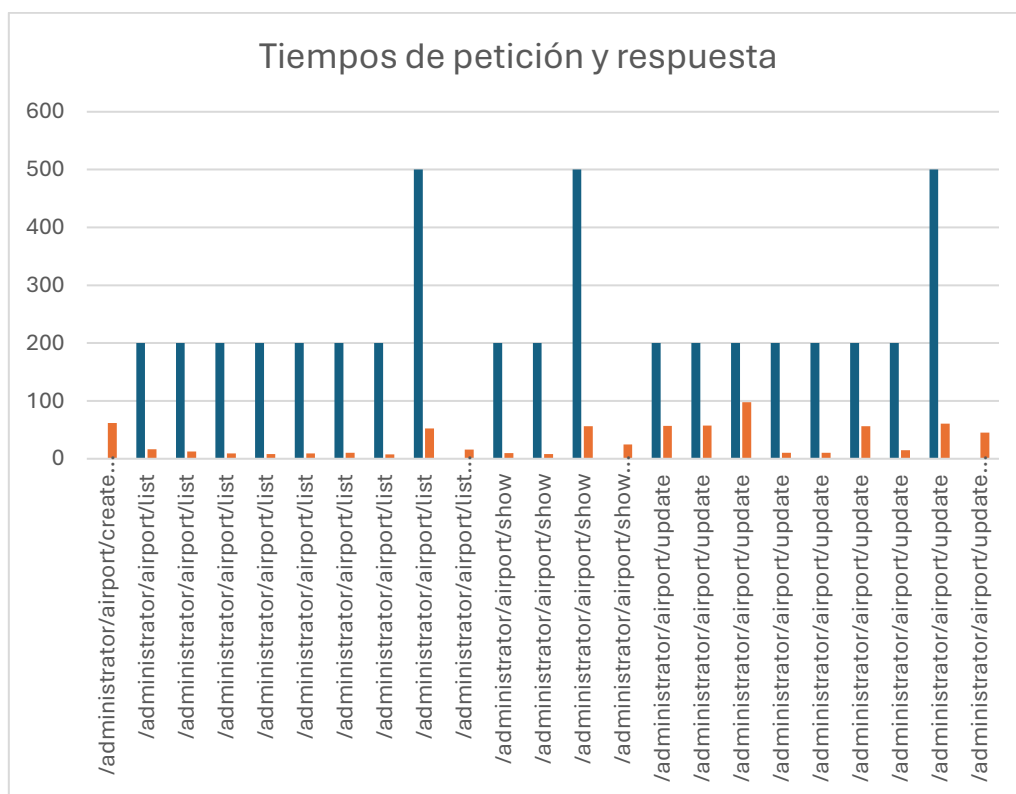
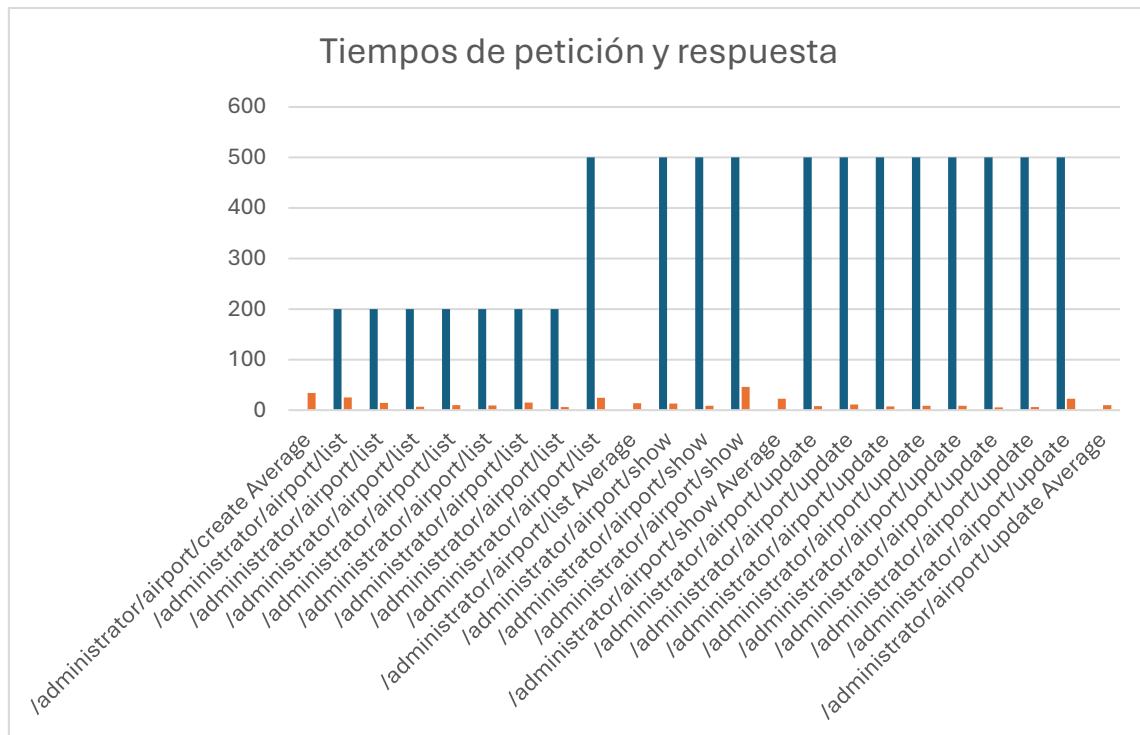


Ilustración 2: Gráfica de tiempo promedio para PC 1



En estas gráficas podemos ver que la *feature* que más tiempo consume en ambos casos es el *create*, siendo el pico más alto en las dos gráficas, ya que, generalmente, sobre todo en las bases de datos relacionales, la insercción de una nueva fila, así como el establecer las nuevas relaciones con otras tablas, consume mas tiempo ya que conlleva mas operaciones que la actualización de un objeto ya existente.

Podemos observar que existe una gran diferencia entre los tiempos de ejecución obtenidos en el PC 1 con los obtenidos en el PC2, siendo los primeros considerablemente más elevados, con una diferencia de, usando la *feature create* como referencia, aproximadamente el doble.

## 2. Análisis de los resultados del análisis de estadística descriptiva

Ahora se mostrarán los resultados obtenidos de realizar el estudio mediante la herramienta que proporciona Excel, *Analysis ToolPak*, para obtener las estadísticas descriptivas para cada PC y estudiar el intervalo de confianza del 95%.

Resultados obtenidos en la estadística descriptiva del PC 1:

<i>Mi PC</i>				
Mean	40,73595517	Interval (ms)	23,95369	57,51822
Standard Error	8,192838069	Interval (s)	0,023954	0,057518
Median	16,6818			
Mode	#N/A			
Standard Deviation	44,11978324			
Sample Variance	1946,555273			
Kurtosis	5,876515017			
Skewness	2,176392416			
Range	197,2342			
Minimum	5,6501			
Maximum	202,8843			
Sum	1181,3427			
Count	29			
Confidence Level(95,0%)	16,78226801			

Resultados obtenidos en la estadística descriptiva del PC 2:

<i>Otro PC</i>				
Mean	20,50826207	Interval (ms)	11,40443	29,61209
Standard Error	4,444346353	Interval (s)	0,011404	0,029612
Median	12,8909			
Mode	#N/A			
Standard Deviation	23,93353757			
Sample Variance	572,8142207			
Kurtosis	16,93543417			
Skewness	3,803260302			
Range	125,3573			
Minimum	5,6572			
Maximum	131,0145			
Sum	594,7396			
Count	29			
Confidence Level(95,0%)	9,103830811			

Como se puede apreciar, se puede afirmar que tanto en rendimiento como en estabilidad es bastante mejor el PC 2 (Otro PC), viendo que *Median* es más baja, lo que indica un tiempo de ejecución medio menor. Además, se aprecia que la desviación estándar del PC 1 (Mi PC) es considerablemente mayor, lo que indica una gran variabilidad en los tiempos de ejecución y, port tanto, unos resultados menos consistentes.

Por otra parte, fijándonos en el nivel de confianza y, por tanto, en los intervalos de confianza de cada uno, indica que, cada uno de los PCs, si se repitieran las pruebas bastantes veces, el 95% de las veces la media estará dentro de ese rango.

Para el PC 1 (Mi PC), el intervalo es algo grande que el del PC 2 (Otro PC), lo que indica una, aunque bastante pequeña, menor precisión a la hora de medir los tiempos. Además, el límite inferior del intervalo de PC 1 es algo menor que el límite superior del intervalo de PC 2, lo que indica que en la mayoría de los casos el tiempo de ejecución de PC 2 será menor, pero, existe una pequeña posibilidad que se de el caso contrario.

### 3. Estudio del Z-Test

Para terminar la comparación, vamos a estudiar los resultados obtenidos de aplicar el Z-Test tal y como se indica en las transparencias de teoría.

Resultados obtenidos en el Z-Test para PC 1 (Mi PC) y PC 2 (Otro PC):

z-Test: Two Sample for Means

z-Test: Two Sample for Means

	<i>Mi PC</i>	<i>Otro PC</i>
Mean	40,73716923	20,50826207
Known Variance	1946,55527	572,8142
Observations	26	29
Hypothesized Mean Difference	0	
z	2,079608493	
P(Z<=z) one-tail	0,018780729	
z Critical one-tail	1,644853627	
P(Z<=z) two-tail	0,037561457	
z Critical two-tail	1,959963985	

Como se puede apreciar, el valor del campo de P(Z<=z) two-tail es 0,037561457, que se encuentra entre el intervalo  $[0.00, \alpha]$ , donde Alpha es 0.05, definido previamente. Este valor está cercano a  $\alpha$ , pero se puede afirmar que no lo suficiente como para afirmar que el Z-Test no es concluyente. Esto indica que se pueden comparar los promedios de los tiempos, y, por ende, se puede afirmar que el PC 2 (Otro PC) es superior en cuanto a rendimiento al PC 1 (Mi PC).

## 7. Conclusiones

Tras este proceso de *testing* y análisis sobre la entidad *Airport*, se ha podido ver como se desenvuelve el sistema a la hora de recibir y enviar peticiones, tanto en cobertura de código y, por consiguiente, robustez del sistema antes datos inválidos o acciones ilegales, como en rendimiento, comparando los tiempos obtenidos del log generado al ejecutar el `testing#replay`. Se ha visto que, para el mismo código y los mismos archivos `.safe` y `.hack`, se han obtenido resultados considerablemente distantes, lo que nos puede indicar que la diferencia y la eficacia no es cuestión del software desarrollado, si no del hardware de cada ordenador.

Concluyendo, podemos afirmar que se obtuvo una correcta implementación de las *features* y sus respectivas restricciones, así como unos valores de rendimiento aceptable, indicando que realizar *testing* es una parte fundamental a la hora de desarrollar un sistema, ya que nos permite ver errores que no se habían visto en el desarrollo, corregirlos y ver como se comporta el sistema para evaluar si su rendimiento es el esperado, o por el contrario, hay que realizar una refactorización. Esto añade fiabilidad y confianza al producto, aumentándolo así su valor tanto para el cliente como para el propio equipo de desarrollo.

## 8. Bibliografía

Intencionalmente en blanco.