

**Informe de TESTING INDIVIDUAL**

Grupo 1-C1.020 | https://github.com/PDJ6975/Acme-ANS-D04-25.5.0



Tabla

El contenido generado por IA puede ser incorrecto.

23 de mayo de 2025

ADRIAN RAMIREZ GIL| Student #5

**Tabla de Contenido**

[**1. Testing Funcional** 1](#_Toc199189053)

[1.1 Introducción 1](#_Toc199189054)

[1.2 Metodología 2](#_Toc199189055)

[1.3 Casos de prueba por característica 3](#_Toc199189056)

[1.3.1 Para MaintenanceRecord 3](#_Toc199189057)

[1.3.2 Para Tasks 6](#_Toc199189058)

[1.4 Calidad de la cobertura 10](#_Toc199189059)

[1.5 Conclusiones finales 11](#_Toc199189060)

[**2. Testing de Rendimiento** 11](#_Toc199189061)

[2.1 Entorno y protocolo 11](#_Toc199189062)

[2.2 Recogida de datos 11](#_Toc199189063)

[2.3 Estadísticas Descriptivas 12](#_Toc199189064)

[2.4 Gráficos de eficiencia por característica 12](#_Toc199189065)

[2.5 Hipótesis y conclusión 14](#_Toc199189066)

[**3. Tabla de Revisión** 14](#_Toc199189067)

# **1. Testing Funcional**

## 1.1 Introducción

En esta sección del documento se pretende describir un listado con los casos de prueba realizados, agrupados por características, para las entidades **MaintenaceRecord** y **Task**.

Para cada caso de prueba, se va a proporcionar una descripción y cuál es su eficacia para detectar errores.

## 1.2 Metodología

Para probar las entidades obligatorias del estudiante cinco, se han aplicado pruebas **“.safe”** y **“.hack”**. Para las primeras se ha probado que las entidades se listen, muestren y permitan un CRUD válido y funcional, permitiendo todo tipo de valores válidos y rechazando aquellos que no cumplen con la validación.

Por otro lado, para las segundas, generalmente se han dividido en pruebas de **“url”** y en pruebas de **valores**, principalmente por dos motivos. El primero es que solía dar problemas hacer las dos pruebas juntas, y el segundo es que permitía ser mucho más específico en cada una de las pruebas.

Del mismo modo, se ha buscado obtener la mayor variabilidad posible en los casos de prueba. Para ello, se han realizado un paso importante, en muchos casos se han realizado los casos de prueba con usuarios distintos para asegurar que todos los elementos se renderizan y funcionan correctamente tanto para usuarios nuevos como ya existentes.

Por último, se muestra una tabla con la cobertura total alcanzada. En lo que respecta a los paquetes globales:

|  |  |
| --- | --- |
| Paquete | Cobertura alcanzada |
| technician.maintenanceRecord | 98.8% |
| technician.tasks | 98.3% |

Por otro lado, en lo que respecta a cada servicio específico de FlightAssignment:

|  |  |
| --- | --- |
| Servicio | Cobertura alcanzada |
| TechnicianMaintenanceRecordCreateService | 98.1% |
| TechnicianMaintenanceRecordListService | 100% |
| TechnicianMaintenanceRecordShowService | 100% |
| TechnicianMaintenanceRecordUpdateService | 98,1% |
| TechnicianMaintenanceRecordPublishService | 98,9% |

Para terminar, para los servicios específicos de ActivityLog:

|  |  |
| --- | --- |
| Servicio | Cobertura alcanzada |
| TechnicianTaskListService | 95,0% |
| TechnicianTaskShowService | 100% |
| TechnicianTaskCreateService | 98,4% |
| TechnicianTaskUpdateService | 98,9% |
| TechnicianTaskPublishService | 98,2% |
| TechnicianTaskDeleteService | 100,0% |

Para los controladores de ambas entidades se alcanza un 100% de la cobertura, lo que significa que se prueban todas las líneas de código contenidas en ellos (es decir, todos los servicios que ofrecen).

## 1.3 Casos de prueba por característica

### 1.3.1 Para MaintenanceRecord

Comencemos especificando los casos de pruebas para las características **“technician/maintenance-record/list?mine=false”** y **“technician/maintenance-record/list?mine=true”**.

|  |  |  |  |
| --- | --- | --- | --- |
| Caso de prueba | Tipo | Descripción | Eficacia |
| list.safe | Renderizado | Se prueba que el listado se muestre correctamente para maintenance record sin publicar y publicadas. | No se detectó ninguna incidencia. |
| list.hack | Hack | Se prueba que:  -Miembros con otro rol no puedan ver el listado. | No se detectó ninguna incidencia ya que se realizaba una validación correcta en el “authorised”. |

Pasemos a **“technician/maintenance-record /show?id=xx”:**

|  |  |  |  |
| --- | --- | --- | --- |
| Caso de prueba | Tipo | Descripción | Eficacia |
| show.safe | Renderizado | Se prueba que un maintenance record se renderice correctamente sin ningún error. | No ha detectado ninguna incidencia ya que el código del servicio estaba correctamente implementado. |
| show.hack | Hack | Permite probar:  -Miembros con otro rol no puedan ver los detalles de una maintenance record.  -Miembros con mismo rol no puedan ver maintenance record que no le corresponden, pero si las publicadas. | No se detectó ninguna incidencia ya que se realizaba una validación correcta en el “authorised”. |

En lo que respecta a **“technician/maintenance-record/create” (GET y POST):**

|  |  |  |  |
| --- | --- | --- | --- |
| Caso de prueba | Tipo | Descripción | Eficacia |
| create.safe | Casos positivos y negativos del formulario | Se prueba que todos los campos del formulario estén correctamente validados, rechazando los valores no admitidos por el modelo. | No detectó ninguna incidencia ya que todos los campos estaban correctamente validados. |
| create.hack | Hack (url y valores) | Permite probar:  -Miembros con otro rol no puedan acceder al formulario de creación  -Atributos read-only no puedan ser modificados.  - Atributos de selección no puedan ser modificados con valores ilegales . | Se detectó que los valores del next inspection due no se comprobaban correctamente. |

En lo que respecta a **“technician/maintenance-record/update” (GET y POST):**

|  |  |  |  |
| --- | --- | --- | --- |
| Caso de prueba | Tipo | Descripción | Eficacia |
| update.safe | Casos positivos y negativos del formulario | Se prueba que todos los campos del formulario estén correctamente validados, rechazando los valores no admitidos por el modelo. | Se detecto un mensaje de error en español cuando debería de ir en inglés. |
| update-url.hack | Hack (url) | Permite probar:  -Miembros con otro rol no puedan acceder a los detalles de la asignación por medio del “update”.  - Miembros del mismo rol no puedan acceder a los detalles de un maintenance record que no les pertenece por medio del “update”.  - Maintenance Record ya publicadas no puedan ser actualizadas. | No detectó ninguna incidencia porque todo estaba bien validado e implementado. |
| update-values.hack | Hack (values) | Permite probar que:  -Atributos read-only no puedan ser modificados.  - Atributos de selección no puedan ser modificados con valores ilegales. | Se arreglo el Aircraf con valor null ya que se detectaba el cambio. |

En lo que respecta a **“technician/maintenance-record/publish” (GET y POST):**

|  |  |  |  |
| --- | --- | --- | --- |
| Caso de prueba | Tipo | Descripción | Eficacia |
| publish.safe | Casos positivos y negativos del formulario | Se prueba que todos los campos del formulario estén correctamente validados, rechazando los valores no admitidos por el modelo. | No detectó ninguna incidencia ya que todos los campos estaban correctamente validados. |
| publish-url.hack | Hack (url) | Permite probar:  -Miembros con otro rol no puedan acceder a los detalles del maintenance record por medio del “publish”.  - Miembros del mismo rol no puedan acceder a los detalles de una maintenance record que no les pertenece por medio del “publish”.  - Maintenance record ya publicadas no puedan volver a publicarse. | No detectó ninguna incidencia porque todo estaba bien validado e implementado. |
| publish -values.hack | Hack (values) | Permite probar que:  -Atributos read-only no puedan ser modificados.  - Atributos de selección no puedan ser modificados con valores ilegales. | No detectó ninguna incidencia porque todo estaba bien validado e implementado. |

### 1.3.2 Para Tasks

Comencemos especificando los casos de pruebas para las características **“technician/task/list?maintenanceRecordId=xx y technician/task/list?mine=false y technician/task/list?mine=true”**

|  |  |  |  |
| --- | --- | --- | --- |
| Caso de prueba | Tipo | Descripción | Eficacia |
| list.safe | Renderizado | Se prueba que el listado de tasks se muestre correctamente para maintenance record publicadas sin publicar y todas las tareas. | No se detectó ningún fallo |
| list.hack | Hack | Se prueba que:  -Miembros con otro rol no puedan ver un listado de tasks.  -Miembros del mismo rol no puedan ver un listado de tasks de un maintenance record que no les pertenece. | No se detectó ninguna incidencia ya que se realizaba una validación correcta en el “authorised”. |

Pasemos a **“technician/task/show?id=xx”:**

|  |  |  |  |
| --- | --- | --- | --- |
| Caso de prueba | Tipo | Descripción | Eficacia |
| show.safe | Renderizado | Se prueba que una task se renderice correctamente sin ningún error. | No ha detectado ninguna incidencia ya que el código del servicio estaba correctamente implementado . |
| show.hack | Hack | Permite probar:  -Miembros con otro rol no puedan ver los detalles de una task.  -Miembros con mismo rol no puedan ver tasks de maintenance records que no le corresponden. | No se detectó ninguna incidencia ya que se realizaba una validación correcta en el “authorised”. |

En lo que respecta a **“technician/task/create?maintenanceRecordId =xx” (GET y POST):**

|  |  |  |  |
| --- | --- | --- | --- |
| Caso de prueba | Tipo | Descripción | Eficacia |
| create.safe | Casos positivos y negativos del formulario | Se prueba que todos los campos del formulario estén correctamente validados, rechazando los valores no admitidos por el modelo. | No detectó ninguna incidencia ya que todos los campos estaban correctamente validados. |
| create.hack | Hack (url y valores) | Permite probar:  -Miembros con otro rol no puedan acceder al formulario de creación de un task.  - Miembros del mismo rol no puedan acceder al formulario de creación de un task asociado a una maintenance record que no les corresponde.  - Miembros del rol correcto no puedan crear tareas para maintenance records ya publicadas | Se comprobó que los maintenance records influían en la petición por lo que se caparon para que se evadieran. |

En lo que respecta a **“technician/task/update” (GET y POST):**

|  |  |  |  |
| --- | --- | --- | --- |
| Caso de prueba | Tipo | Descripción | Eficacia |
| update.safe  (para en y es) | Casos positivos y negativos del formulario | Se prueba que todos los campos del formulario estén correctamente validados, rechazando los valores no admitidos por el modelo. | No detectó ninguna incidencia ya que todos los campos estaban correctamente validados. |
| update-url.hack | Hack (url) | Permite probar:  -Miembros con otro rol no puedan ver los detalles de una task mediante el “update”.  -Miembros con mismo rol no puedan ver tasks maintenance records que no le corresponden mediante el “update”.  - Miembros del rol correcto no puedan tratar de actualizar tasks ya publicadas. | Se comprobó que los maintenance records influían en la petición por lo que se caparon para que se evadieran. |
| update-values.hack | Hack (values) | Permite probar que:  Los desplegables salten con errores de invalid value al ser enumerados | No se detectó ninguna incidencia. |

En lo que respecta a **“technician/task/publish” (GET y POST):**

|  |  |  |  |
| --- | --- | --- | --- |
| Caso de prueba | Tipo | Descripción | Eficacia |
| publish.safe | Casos positivos y negativos del formulario | Se prueba que todos los campos del formulario estén correctamente validados, rechazando los valores no admitidos por el modelo. | Se detecto que los campos pasaban a read only cuando una validación fallaba pero no se aplicaba el publish porque al volver a acceder a esta entidad los campos ya volvían a ser editables. |
| publish-url.hack | Hack (url) | Permite probar:  -Miembros con otro rol no puedan ver los detalles de una task mediante el “publish”.  -Miembros con mismo rol no puedan ver tasks de maintenance records que no le corresponden mediante el “publish.”  - Miembros del rol correcto no puedan tratar de publicar tasks ya publicadas. | No detectó ninguna incidencia porque todo estaba bien validado e implementado. |
| publish -values.hack | Hack (values) | Permite probar que:  -Atributos read-only (una fecha) no puedan ser modificados. | No se detectó ninguna incidencia ya que la fecha no estaba en el “bind” del servicio. |

Por último, para **“technician/task/delete” (GET y POST):**

|  |  |  |  |
| --- | --- | --- | --- |
| Caso de prueba | Tipo | Descripción | Eficacia |
| delete.safe  (para en y es) | Casos positivos y negativos del formulario | Se prueba que una task se borre correctamente. | No se detectó ninguna incidencia. |
| delete-url.hack | Hack (url) | Permite probar:  -Miembros con otro rol no puedan acceder a los detalles de la task por medio del “delete”.  - Miembros del mismo rol no puedan acceder a los detalles de una task que no les pertenece por medio del “delete”.  - Task ya publicadas no puedan ser borradas. | No detectó ninguna incidencia porque todo estaba bien validado e implementado. |

## 1.4 Calidad de la cobertura

En esta sección del documento se pretende describir la calidad de la cobertura para los servicios de las entidades Maintenance Record y Task. Tanto para el listado como para los detalles, se alcanza un 100% de cobertura (menos para el listado de tasks donde existe un caso que no tiene mucho sentido de comprobar, pero es conveniente tener todos los casos controlados, con un 95%), lo que significa que se cubren casi todas las líneas de código.

Para los servicios de CRUD, la cobertura gira alrededor del 95-99% debido a los métodos de validación. Si bien en los casos de prueba se cubren prácticamente todos los escenarios posibles, he añadido validaciones extra de seguridad con el objetivo de tener un sistema sólido y fiable, para asegurar que no hay ninguna brecha de seguridad. Por ejemplo, se han añadido validaciones duplicadas en el “authorise” y en el “validate” para asegurar la seguridad en todas las capas. También se han contemplado casos muy extremos que ni si quiera se pueden probar directamente a través del frontend, como que lleguen miembros o “logs” nulos a las validaciones, pues se deberían detectar en el authorise.

Por estos motivos alcanzar el 100% de cobertura en todos los servicios es prácticamente imposible, o por lo menos inviable.

## 1.5 Conclusiones finales

En definitiva, se han desarrollado un conjunto de pruebas que cubre la gran mayoría de líneas de código de las entidades del estudiante cinco. Aunque haya métodos de validación que se pueden optimizar, se ha cumplido el objetivo de obtener un sistema robusto y seguro ante la gran mayoría de escenarios que se dan en el mundo real.

# **2. Testing de Rendimiento**

## 2.1 Entorno y protocolo

Para el desarrollo de esta sección se va a emplear un equipo con las siguientes características:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Modelo | RAM | CPU | GPU | Disco | SO |
| HP Victus | 8.0 GB | AMD Ryzen 5 5600H 3.30 GHz | Nvidia Geforce GTX | 475 GB | Windows 10 |
| HP | 8GB | **CPU:** Intel Pentium 4 1.8 GHz (single-core) | AMD Radeon HD 7310 (integrada) | 500 GB | Windows 10 |

Para este estudio se va a emplear únicamente los casos de prueba que he ejecutado individualmente como estudiante cinco, empleando la versión 25.5.0 del proyecto y del framework. En el primer lanzamiento, no se emplearán índices en mis entidades, para ver si el rendimiento mejora una vez se implementen.

## 2.2 Recogida de datos

Para la obtención de datos, tanto antes como después de implementar índices, se han seguido los pasos explicados en la teoría, analizando la información del “.trace” generado por la aplicación y filtrándola para obtener un fichero limpio del que poder obtener información útil como gráficos.

## 2.3 Estadísticas Descriptivas

Una vez filtradas las peticiones realizadas en los casos de prueba eliminando las irrelevantes, podemos afirmar que, para el estudio, se utiliza un total de 1208 filas de datos. En ambos casos se presenta una media y desviación similares, de los que se pueden derivar, además, los siguientes datos:

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

El contenido generado por IA puede ser incorrecto. Interfaz de usuario gráfica, Texto, Aplicación, Tabla

El contenido generado por IA puede ser incorrecto.

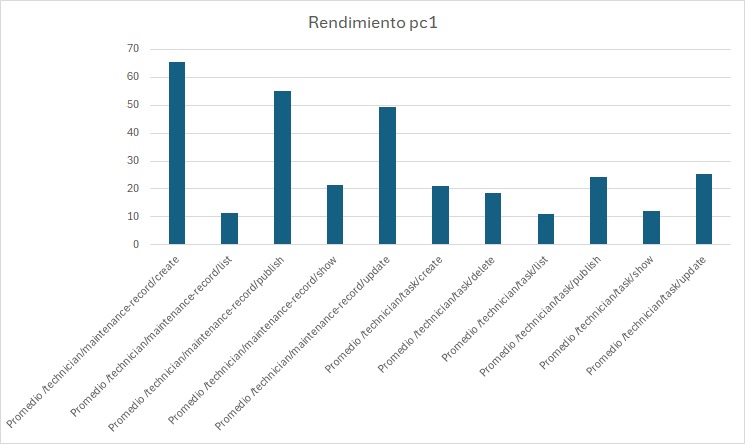
Como podemos observar, las trazas “antes” y “después” presentan medias (≈ 15 ms) con una diferencia, por lo que en un principio la incorporación de índices no introduce cambios significativos en el rendimiento.

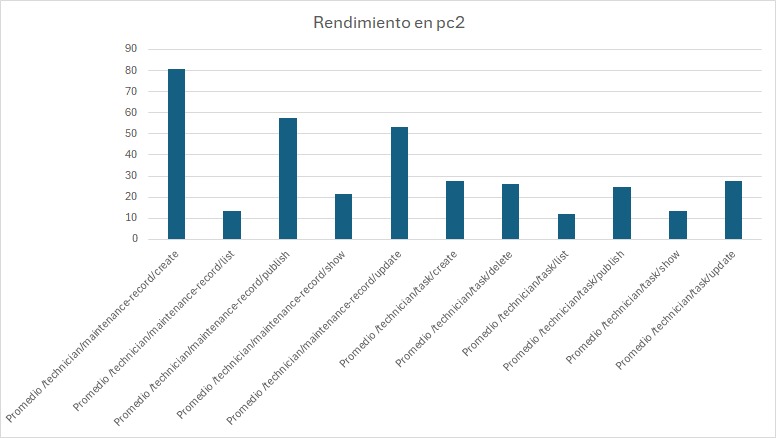
Esto se puede observar también en los intervalos de confianza, que están bastante juntos. Sin embargo, para ambas trazas se cumple con creces el requisito de rendimiento, ya que los límites superiores (≈ 16.41 ms) son ambos menores a 1 segundo.

Por tanto, una de las conclusiones que podemos sacar es que este equipo ejecuta los casos de prueba bastante rápido. Aun así en mi caso no es necesario implementar índices para mis entidades.

## 2.4 Gráficos de eficiencia por característica

Uno de los aspectos que más nos interesa del rendimiento, es comprobar cuál es la petición más ineficiente (MIR), y ver si los índices suponen una mejora real en el rendimiento de la misma.





Claramente hay una petición mucho más ineficiente que el resto, **“technician/maintenance-record/create”**. Esto se debe a que, para crear una maintenance record, se debe recuperar información de los aircrafts, el status que va a tener la hora del sistema y el técnico que la crea, lo que hace que tenga una carga de trabajo grande.

Podemos observar entre los ordenadores que en uno de ellos los programas se ejecutan muchísimo mejor que el otro. Esto se debe a la edad de los ordenadores que son un poco antiguos los dos.

## 2.5 Hipótesis y conclusión

**Conclusión**  
El conjunto de pruebas funcionales muestra una cobertura muy alta (95–100 %) y robustez en validaciones de roles y valores, garantizando un sistema fiable ante escenarios reales. Desde el punto de vista de rendimiento, con medias cercanas a 15 ms y picos inferiores a 17 ms, el requisito de sub-1 s se cumple con holgura sin necesidad urgente de índices. Dado el tamaño de datos actual, los índices no aportan mejoras significativas; sin embargo, es recomendable volver a evaluar su impacto cuando el volumen de registros crezca sustancialmente o se incorporen nuevas consultas complejas. Además en mi caso era innecesario de unos índices.

# **3. Tabla de Revisión**

|  |  |  |
| --- | --- | --- |
| Versión | Fecha | Descripción de los cambios |
| 1.0 | 26/05/2025 | Creación inicial del documento |