## **DP2 – Testing Report**

**Group:** C3.027  
**Repository:** <https://github.com/DP2-C1-027/AirNav-Logistics-C3>

**Group Members:** Niza Cobo, Manuel Jesús; Gomez Claraco, Nicolas.

**Corporate Emails:** [mannizcob@alum.us.es](mailto:mannizcob@alum.us.es) , [nicgomcla@alum.us.es](mailto:nicgomcla@alum.us.es)   
**Date:** 10/13/2025

**Table of Contents**

1. Executive Summary
2. Revision Table
3. Introduction
4. Contents
5. Conclusions
6. Bibliography

## **Executive Summary**

This report provides an overview of the testing phase conducted to assess the functionalities developed by the group C3.027. The objective of the testing process was to ensure that the implemented features meet the defined quality standards in terms of functionality and performance.

Functional testing confirmed that the developed requirements generally behave as expected, fulfilling the specified criteria. Performance testing was also conducted to evaluate the system’s efficiency under different load conditions. The results indicate that the software performs within acceptable limits, with some areas identified for potential optimization.

Overall, the tests demonstrate that the implemented functionalities are stable and meet the project’s requirements, although minor improvements could further enhance performance and maintainability.

## **Revision Table**

|  |  |  |
| --- | --- | --- |
| **Revision Number** | **Date** | **Description** |
| 1.0 | 10/14/2025 | Initial version of the document |

## **Introduction**

The purpose of this document is to analyze and evaluate the operation and performance of the requirements developed by the group C3.027. Through various functional and performance tests, the aim is to ensure that the implemented functionalities meet the expected quality criteria, both in terms of correct behavior and efficiency.

First, there is a section dedicated to functional testing, whichcollects the results obtained after running various tests on thedeveloped functionalities. Secondly, there is a section focused on performance testing, which studies the efficiency of the software under certain conditions.

## **Content**

## **Functional Testing**

Functional testing was carried out to verify that #11 requirement behave according to the expected specifications. The tests were executed using .safe and .hack files that represent different input scenarios, including typical and boundary cases.

All test cases executed successfully, and no functional defects were detected during this phase:

11) Operations by **administrators** on **airports**:

* List the airports and show their details.

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | Description | File | Result |
| List airports | - Access the list of airports with administrator account. | list.safe | High (100%) |
| - Attempting to access the list of airports with technician member account. | list.hack |

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | Description | File | Result |
| Show the details of an airport | - Access airport data with administrator account. | show.safe | High (100%) |
| - Attempting to access airport data with technician member account.  - Attempting to access airport data without an id provided.  - Attempting to access airport data with an incorrect id. | show.hack |

* Create and update an airport. Both operations require confirmation.

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | Description | File | Result |
| Create an airport | - Create a valid airport.  - Attempting to create an invalid airport, testing all the fields with incorrect data. | create.safe | High (100%) |
| - Attempting to access the create airport view with technician member account.  - Attempting to create an airport without and id provided.  - Attempting to create an airport with and id provided different from 0. | create.hack |
| Update an airport | - Update a valid airport.  - Attempting to update an invalid airport, testing all the fields with incorrect data. | update.safe | High (100%) |
| - Attempting to access the update airport view with technician member account.  - Attempting to update an airport without and id provided.  - Attempting to update an airport with an invalid id provided. | update.hack |

We will analyze the code coverage we have achieved thanks to the functional tests performed. Code coverage indicates what percentage of the source code has been executed during testing. This allows us to identify the areas that have been verified and those that have not yet been analyzed (which may contain undetected errors):

Gráfico

Descripción generada automáticamente con confianza media

**Figure 1** - Coverage of requirement #11

We can see that classes related with requirement #11 reach close to 100%. This implies that much of the code implemented to perform these functionalities is tested in case there is an error that has not been contemplated.

To justify the remaining 0.2%, I will explain which concepts have not been analyzed or taken into account:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Figure 2** - Coverage of Update functionality

Although exhaustive tests are performed on both .safe and .hack to test the code, Eclipse has not been able to detect the different tests.

## **Performance Testing**

Here we present the results obtained after running the functional tests, developed for requirement #11 of Group C3.027. The main objective is to evaluate the impact of the tests on response times, evaluate the performance of the software, and the performance of the hardware on each of the different devices.

To this end, several tests were carried out, an initial one without indexes and a subsequent one with the optimized indexes applied, measuring the average response times on two different devices: the work laptop and the personal PC.

**Exhibition on the results**

The results obtained by analyzing the log trace when running the launcher replay. During this process, the average response times of the different features were recorded, allowing statistics to be generated.

Based on the data obtained, the average response times showed anincrease, going from 20 ms before optimization to 24.64 ms after the implementation of indexes. However, given that the standard deviations in both cases are greater than 50 ms, the difference was not statistically significant at a 95% confidence level.

The two-tailed p-value is 0.275, which reinforces the assertion that there is insufficient evidence to claim that response times improved significantly, as this value is above the alpha threshold of 0.05 on device 1.

The 95% confidence intervals overlap:

Before the change: [16 ms, 24 ms]

After the change: [15 ms, 23 ms]

In contrast, device 2 shows that the average response times were drastically reduced compared to those of device 1. Regarding the initial case and the final case, with indices going from 13.41 ms before optimization to 13.08 ms after the implementation of indexes. However, given that the standard deviations in both cases are greater than 400 ms, the difference is not statistically significant at a 95% confidence level.

The two-tailed p-value is 0.696, which confirms that there was almost noimprovement from the initial version to the final version with indexes, since this value is above the alpha threshold of 0.05 on device 2.

The 95% confidence intervals overlap:

Before the change: [12.02 ms, 14.81 ms]

After the change: [11.64 ms, 14.4 ms]

The conclusions we can draw are that in neither of the two devices is there a substantial difference in improvement from the previous version to the later one with indices. In device 1, it is somewhat more noticeable but still above the alpha threshold of 0.05. Regarding response times, device 2 shows an improvement that affects the confidence intervals.

**Software performance with VisualVM**

To perform this test, VisualVM was used to identify the methods and classes that consume the most resources during execution, allowing the detection of possible bottlenecks that can be optimized through refactoring or index improvements. The results of this test show that the application is running smoothly and efficiently, with no significant bottlenecks or performance issues.

A detailed analysis of resource consumption at the class and method level can be observed during project execution. It allows you to visualize which parts of the code involve a greater workload, facilitating the identification of critical points in the system, including the unbind of ListPlanned (Flight Assignment), the unbind of Create (Flight Assignment), and the load/unbind of ListComplete (Flight Assignment). These methods can be improved through refactoring or improvements in database queries thanks to indexes, all with the aim of optimizing the application's performance.

**Device performance**

Finally, the overall performance of the system, specifically the hardware, was monitored during the execution of the tests, which provides a more complete view of the performance and load of the system on the two devices.

As can be seen, the performance on device 2 is similar to that of device 1 during the execution of the system functional tests. Both devices have no difficulty in coping with the tests, as there is no bottleneck in their hardware.

Based on the various tests performed, it can be concluded that device 2 (PC) performs substantially better than device 1 (laptop). The metrics obtained also show that response times are shorter and that it offers greater stability during the execution of functional tests. This shows that the PC, with its better hardware performance, responds much better to the workload of functional tests, making it a more suitable option for running this type of test.

## **Conclusions**

Since we want to evaluate the impact of functional tests and system performance, multiple functional tests were carried out, satisfying 99.9% coverage for the FlightAssignment and ActivityLog entities and their functionalities.

In performance testing, it can be stated that neither of the two devices showed statistically significant improvements in response times after the inclusion of indexes. Although in performance testing it can be stated that neither of the two devices showed statistically significant improvements in response times after the inclusion of indexes, in performance testing it can be stated that neither of the two devices showed statistically significant improvements in response times after the inclusion of indexes.

In performance testing, it can be stated that neither of the two devices showed statistically significant improvements in response times after the inclusion of indexes. Although the reduction was more noticeable on average on device 1, it was not sufficient to exceed the statistical confidence threshold of 0.05. On the other hand, device 2, thanks to its greater On the other hand, device 2, thanks to its greater processing capacity, offered lower overall response times, with less dispersion, thus confirming its technical superiority in terms of performance.

The 95% confidence intervals achieved by both devices are:

**Device 1:**

Before the change: [16 ms, 24 ms]

After the change: [15 ms, 23 ms]

**Device 2:**

Before the change: [12.02 ms, 14.81 ms]

After the change: [11.64 ms, 14.4 ms]

In addition, thanks to VisualVM, some MIRs (resource-intensive methods) were identified, representing additional opportunities for improvement, such as possible refactorings or redesign of the indexes applied in the entities as possible suggestions.

## **Bibliography**

Intentionally blank.