Workgroup C1.067

Testing report

24/05/2025

Javier Arellano López – [javarelop@alum.us.es](mailto:javarelop@alum.us.es)

https://github.com/javiarellanoo/Acme-ANS-D04

Table of Contents

[EXECUTIVE SUMMARY: 2](#_Toc199028227)

[REVISION TABLE 3](#_Toc199028228)

[INTRODUCTION 4](#_Toc199028229)

[FUNCTIONAL TESTING 4](#_Toc199028230)

[Operations of Managers on Flights 4](#_Toc199028231)

[Operations by Managers on Legs 8](#_Toc199028232)

[PERFORMANCE TESTING 12](#_Toc199028233)

[Mean Confidence Interval 12](#_Toc199028234)

[Contrasting information 14](#_Toc199028235)

[CONCLUSIONS 15](#_Toc199028236)

[BIBLIOGRAPHY 15](#_Toc199028237)

# EXECUTIVE SUMMARY:

This document serves as a comprehensive and complete overview of the testing procedures applied to validate the functionality and performance of the features involved in Student #1 requirements, with the aim of assessing complete security within the application, complying with “Ley Orgánica 3/2018”.

The functional testing section of the document includes brief descriptions of the test procedures performed for each feature available to airline managers, with respect to Flights and Legs. To carry out said testing, a set of legal and hacking scenarios have been recorded and reproduced. Across these tests, particular attention was given to edge cases within the stablished ranges of attributes, input validation, navigation attributes and access control. Given that all read-only attributes of these features are computed attributes, it was established by the lecturer in the last follow-up session that they were not hackable, hence, those test cases have not been recorded.

The second and last part of the report is devoted to performance testing. By computing the performance in two different hardware environments, this reports establishes the required confidence intervals and compares the execution times to determine the most powerful machine.

# REVISION TABLE

|  |  |  |
| --- | --- | --- |
| Revision Number | Date | Description |
| 1.0 | 24/05/2025 | Initial version of the document |
|  |  |  |
|  |  |  |

# INTRODUCTION

The goal of this document is to provide an analysis of the testing procedure of the application for the requirements related to Student #1. The document will be structured in two parts.

The first one includes information relative to the functional testing process, including a list of all test cases implemented, grouped by feature. For each test case, a succinct description of the tests conducted and an indication of whether bugs have been discovered will be provided.

The second section includes the information relative to performance testing, including charts regarding this performance as well as a 95% confidence interval for the time taken by the project to serve the requests of the functional tests in two different hardware environments. Finally, the most powerful computer will be determined.

# FUNCTIONAL TESTING

## Operations of Managers on Flights

List

The following safe cases have been implemented:

* Go into the list of flights of each user and check every page to make sure that entries render correctly. This has been done across different test cases, since it was also required to be able to access the “show” feature of each Flight.
  + Detection of bugs: No bugs were detected.

The hacking cases include:

* Request the flight listing feature as any other realm, for example non authenticated or administrator. An authorization exception was thrown.
  + Detection of bugs: No bugs were detected.

Show

The safe cases include:

* Request the feature for all flights in the sample data, by the manager that created them, checking that everything renders properly.
  + Detection of bugs: No bugs were detected
* Request the feature for a published flight by a manager that did not create it, checking that it is visible.
  + Detection of bugs: No bugs were detected.

The hacking cases implemented include:

* Request the feature for a flight in draft mode with a realm distinct to the one associated with said flight (i.e a different manager or a non authenticated user) . An authorization exception is thrown
  + Detection of bugs: None were detected
* Request the feature for a non existent flight. An authorization exception is thrown.
  + Detection of bugs: None were detected.

Create

The following safe test cases were performed:

* Sending the form with all null values to check that no exceptions are thrown.
  + Detection of bugs: No bugs were detected.
* For each attribute of the form, as many invalid and valid data as possible have been taken into account. Said variations were taken from the “Sample-Data” file.
  + Detection of bugs: None were detected.
* Submit a form with all valid data to see that the flight is created properly:
  + Detection of bugs: None were detected.

The hacking cases implemented include:

* Requesting the feature with other realms. Checking that an authorization error is thrown
  + Detection of bugs: None were detected
* Hack the “airline” navigation attribute using the “DevTools”. Changed the value for a non-existing airline id (i.e 999 or -1). Check that the application returned an authorization exception.
  + Detection of bugs: No bugs detected.

Update

The safe cases implemented were:

* Submit an empty form to check that no exceptions were thrown and errors were reported in the correct attributes.
  + Detection of bugs: No bugs were detected.
* For each attribute of the form, as many invalid and valid data as possible have been taken into account. Said variations were taken from the “Sample-Data” file.
  + Detection of bugs: None were detected.
* Submit a form with all valid data to see that the flight is updated properly:
  + Detection of bugs: None were detected.

The hacking cases implemented include, among others:

* Requesting the feature for a Flight that has already been published (with both the manager that created it and another one). Check that an authorization exception is thrown.
  + Detection of bugs: None were detected
* Requesting the feature for a Flight in draft mode that does not belong to the logged in manager. Check that an authorization exception is thrown.
  + Detection of bugs: None were detected
* Request the feature for a non existing Flight.
  + Detection of bugs: None were detected.
* Hack the “airline” navigation attribute using the “DevTools”. Changed the value for a non-existing airline id (i.e 999 or -1). Check that the application returned an authorization exception.
  + Detection of bugs: No bugs detected.

Publish:

The safe cases implemented were:

* Submit an empty form to check that no exceptions were thrown and errors were reported in the correct attributes.
  + Detection of bugs: No bugs were detected.
* For each attribute of the form, as many invalid and valid data as possible have been taken into account. Said variations were taken from the “Sample-Data” file. For a flight to be published, it must have Legs, and all of the Legs must be published. So that tests with a Flight with no legs, legs in draft mode and published legs were conducted so that all possible combinations were covered, checking that no cases lead to bugs.
  + Detection of bugs: None were detected.
* Submit a form with all valid data and published legs to see that the flight is published properly:
  + Detection of bugs: None were detected.

The hacking cases implemented include, among others:

* Requesting the feature for a Flight that has already been published (with both the manager that created it and another one). Check that an authorization exception is thrown.
  + Detection of bugs: None were detected
* Requesting the feature for a Flight in draft mode that does not belong to the logged in manager. Check that an authorization exception is thrown.
  + Detection of bugs: None were detected
* Request the feature for a non existing Flight.
  + Detection of bugs: None were detected.
* Hack the “airline” navigation attribute using the “DevTools”. Changed the value for a non-existing airline id (i.e 999 or -1). Check that the application returned an authorization exception.
  + Detection of bugs: No bugs detected.

Delete:

The following safe cases were implemented:

* Requesting the feature normally, with a flight in draft mode. Check that the flight is properly deleted
  + Detection of bugs: None were detected

The hacking cases include, among others:

* Requesting the delete feature for a non existing flight. Check that an authorization exception is thrown.
  + Detection of bugs: No bugs were found
* Request the feature for an already published flight. Check that an authorization error is thrown.
  + Detection of bugs: No bugs were found.
* Request the feature directly through the URL. Check that the GET phase of the delete request has been restricted with an authorization error so that only requests coming from the application are valid.
  + Detection of bugs: No bugs were detected
* - Request the feature for a flight in draft mode with a different principal than the one who created it. Check that it returns an authorization error.
  + Detection of bugs: No bugs were found.

## Operations by Managers on Legs

List

The following safe cases have been performed:

* For each Flight, it has been checked that the listing of legs associated with said flight is visible by the manager that created it.
  + Detection of bugs: No bugs were detected
* For a published Flight, it has been checked that the listing of legs associated with that flight is visible by other managers.
  + Detection of bugs: No bugs were detected

The hack cases include:

* Trying to list the legs of a non-existing Flight. Check that an authorization exception is thrown.
  + Detection of bugs: No bugs were detected
* Trying to list the legs of a non-published Flight with any other realm than the one that created it.
  + Detection of bugs: No bugs were detected.

Show

The following safe cases have been performed:

* For each Leg in the sample data, it has been checked that they can be correctly seen, by their creator if they are in draft mode and by all managers if they flight that they belong to is published, (for the Flight to be published, the leg must have also been published).
  + Detection of bugs: None were detected

The following hack cases have been implemented, among others:

* Trying to show a non existing leg. It has been checked that an authorization exception is thrown.
  + Detection of bugs: None were found.
* Trying to show a published leg of a non published flight by another manager. It has been checked that an authorization exception is thrown.
  + Detection of bugs: None were found
* Trying to show a non published leg of a non published flight by another manager. It has been checked that an authorization exception is thrown.
  + Detection of bugs: None were found.

Create

The following safe cases have been performed:

The safe cases implemented were:

* Submit an empty form to check that no exceptions were thrown and errors were reported in the correct attributes.
  + Detection of bugs: No bugs were detected.
* For each attribute of the form, as many invalid and valid data as possible have been taken into account. Said variations were taken from the “Sample-Data” file. Natural intelligence has been used to generate different variations of the unique attributes, such as the Flight Number (repeated, not fitting the pattern, not starting with the Airline’s IATA code)
  + Detection of bugs: None were detected.
* Submit a form with all valid data to see that the leg is created properly:
  + Detection of bugs: None were detected.

The hacking cases implemented include, among others:

* Requesting the feature for a Flight that has already been published (with both the manager that created it and another one). Check that an authorization exception is thrown.
  + Detection of bugs: None were detected
* Requesting the feature for a Flight in draft mode that does not belong to the logged in manager. Check that an authorization exception is thrown.
  + Detection of bugs: None were detected
* Request the feature for a non existing Flight.
  + Detection of bugs: None were detected.
* Hack all navigation attributes (“aircraft”, “destinationAirport”, “arrivalAirport” ) using the “DevTools” and using an Id belonging to a non existing flight (i.e 999 or -1).

Update

The following safe cases have been performed:

The safe cases implemented were:

* Submit an empty form to check that no exceptions were thrown and errors were reported in the correct attributes.
  + Detection of bugs: No bugs were detected.
* For each attribute of the form, as many invalid and valid data as possible have been taken into account. Said variations were taken from the “Sample-Data” file. Natural intelligence has been used to generate different variations of the unique attributes, such as the Flight Number (repeated, not fitting the pattern, not starting with the Airline’s IATA code)
  + Detection of bugs: None were detected.
* Submit a form with all valid data to see that the leg is updated properly:
  + Detection of bugs: None were detected.

The hacking cases implemented include, among others:

* Requesting the feature for a Flight that has already been published (with both the manager that created it and another one). Check that an authorization exception is thrown.
  + Detection of bugs: None were detected
* Requesting the feature for a leg of Flight in draft mode that does not belong to the logged in manager. Check that an authorization exception is thrown.
  + Detection of bugs: None were detected
* Request the feature for a non existing Flight.
  + Detection of bugs: None were detected.
* Hack all navigation attributes (“aircraft”, “destinationAirport”, “arrivalAirport” ) using the “DevTools” and using an Id belonging to a non existing flight (i.e 999 or -1).

Publish

The following safe cases have been performed:

The safe cases implemented were:

* Submit an empty form to check that no exceptions were thrown and errors were reported in the correct attributes.
  + Detection of bugs: No bugs were detected.
* For each attribute of the form, as many invalid and valid data as possible have been taken into account. Said variations were taken from the “Sample-Data” file. Natural intelligence has been used to generate different variations of the unique attributes, such as the Flight Number (repeated, not fitting the pattern, not starting with the Airline’s IATA code)
  + Detection of bugs: None were detected.
* Submit a form with all valid data to see that the leg is published properly:
  + Detection of bugs: None were detected.

The hacking cases implemented include, among others:

* Requesting the feature for a Flight that has already been published (with both the manager that created it and another one). Check that an authorization exception is thrown.
  + Detection of bugs: None were detected
* Requesting the feature for a leg of a Flight in draft mode that does not belong to the logged in manager. Check that an authorization exception is thrown.
  + Detection of bugs: None were detected
* Request the feature for a non existing Flight.
  + Detection of bugs: None were detected.
* Hack all navigation attributes (“aircraft”, “destinationAirport”, “arrivalAirport” ) using the “DevTools” and using an Id belonging to a non existing flight (i.e 999 or -1).

Delete

The following safe cases were conducted:

* Requesting the feature normally, with a flight in draft mode. Check that the leg is properly deleted
  + Detection of bugs: None were detected

The hacking cases include, among others:

* Requesting the delete feature for a non existing leg. Check that an authorization exception is thrown.
  + Detection of bugs: No bugs were found
* Request the feature for an already published leg. Check that an authorization error is thrown.
  + Detection of bugs: No bugs were found.
* Request the feature directly through the URL. Check that the GET phase of the delete request has been restricted with an authorization error so that only requests coming from the application are valid.
  + Detection of bugs: No bugs were detected
* - Request the feature for a leg in draft mode with a different principal than the one who created it. Check that it returns an authorization error.
  + Detection of bugs: No bugs were found.

# PERFORMANCE TESTING

In this last section, we will evaluate the performance of the project by measuring the elapsed time taken to complete requests during functional testing. The objective is to assess how quickly the system responds under real conditions to determine which computer performs the best.

To collect the data required, we will run all functional tests for manager features. We will be using the following devices:

* MSI Pulse Gl76: 16 GB RAM, 1 TB SSD memory
* HP Victus: 16 GB RAM, 1 TB SSD memory

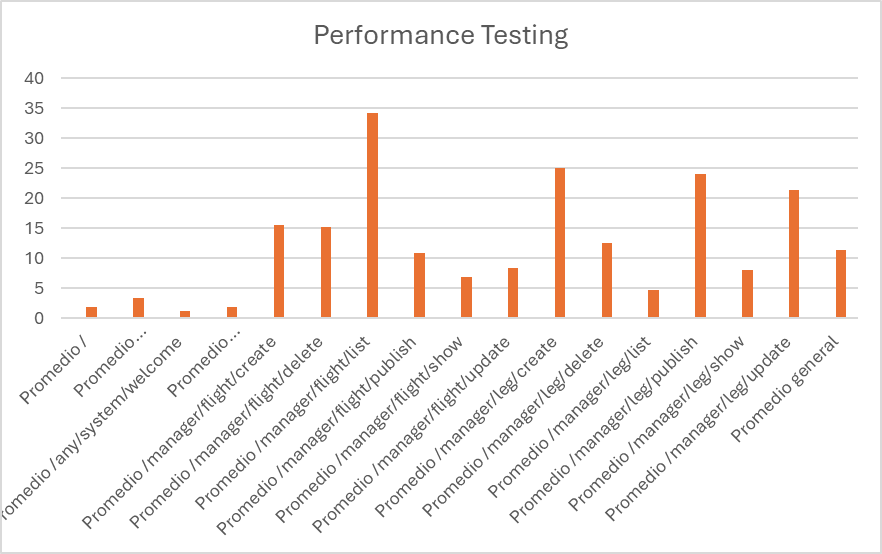
## Mean Confidence Interval

After cleaning up and treating the data obtained from the trace file that is generated as a result of replaying tests, I have performed an analysis using the procedure explained in the subject to obtain an average time for each of the requests executed.



For the first computer, we have obtained a grand average of 13.3 ms. As it can be appreciated, we see that the MIR is the leg update, whose average is of 67.79 ms.

Using the data analyzer extension from Excel, we obtain that the amplitude of the confidence interval at 95% is 1.27 ms. By removing and adding this value from the average, we obtain the confidence interval: [0.014 s – 0.017 s]



When doing the same with the HP Victus PC, we obtained a similar graph. In this case, the grand average stands at 11.31 ms. And as a subtle difference, the apparent MIR is the flight listing feature, with an average of 34.25 ms.

Using the data analyzer, we get that the amplitude of the confidence interval at 95% is 1.043 ms. We would obtain the following confidence interval: [0.015 s– 0.017s]

## Contrasting information

Given the previous results, we can induce that the computer with the best performance is the HP Victus one, as in general, all its average times are lower. Let us use the Z-Test with alpha 0.05 ( one minus confidence) to verify our hypothesis.

Once the Z-Test has been conducted, the results are the following:

Imagen que contiene Texto

El contenido generado por IA puede ser incorrecto.

As it can be clearly seen, the second computer is way faster than the first one, as we initially supposed.

# CONCLUSIONS

This document serves as a means of reporting the results of test cases for future references. The analysis recorded provides a detailed evaluation of the test cases in order to identify possible bugs in the code and make informed decisions to improve the system as well as to ensure protection against possible hacking attempts.

Each test case was recorded systematically, grouped by feature and the bugs found, if any, have been reported. This serves as a transparent reference of the testing process.

Through the analysis of the performance on the two different computers, we have demonstrated that the HP Victus computer has consistently obtained a better performance. As expected, both systems showed similar behavior in terms of request distribution, however, the obtained MIR has been different in each case. Since the difference on average of the MIR obtained for the second PC and the one obtained for the first one in the first case are very close, this can be a punctual case.

This testing report provides concise descriptions of the tests performed, in hopes that they serve as solid foundation for quality assurance in the near future.

# BIBLIOGRAPHY

Intentionally blank