Workgroup C1.067

Testing report

25/05/2025

Raúl Calero Capote – [raucalcap@alum.us.es](mailto:raucalcap@alum.us.es)

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

Table of Contents

[EXECUTIVE SUMMARY: 2](#_Toc199187165)

[REVISION TABLE 2](#_Toc199187166)

[INTRODUCTION 3](#_Toc199187167)

[FUNCTIONAL TESTING 4](#_Toc199187168)

[Operations of Flight Crew Members on Flight Assignments 4](#_Toc199187169)

[Operations by Flight Crew Member on Activity Logs 9](#_Toc199187170)

[PERFORMANCE TESTING 12](#_Toc199187171)

[Mean Confidence Interval 13](#_Toc199187172)

[Contrasting information 14](#_Toc199187173)

[CONCLUSIONS 15](#_Toc199187174)

[BIBLIOGRAPHY 15](#_Toc199187175)

# 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 #3 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 flight crew members, with respect to Flight Assignments and Activity Logs. 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, read-only attributes and access control.

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

# REVISION TABLE

|  |  |  |
| --- | --- | --- |
| Revision Number | Date | Description |
| 1.0 | 25/05/2025 | 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 #3. 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 Flight Crew Members on Flight Assignments

List

The following safe cases have been implemented:

* Go into the list of completed and incomplete assignments 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 Assignment.
  + Detection of bugs: No bugs were detected.

The hacking cases include:

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

Show

The safe cases include:

* Request the feature for all assignments in the sample data, by the crew member that created them, checking that everything renders properly.
  + Detection of bugs: No bugs were detected

The hacking cases implemented include:

* Request the feature for an assignment with a realm distinct to the one associated with said assignment (i.e. a different crew member). An authorization exception is thrown
  + Detection of bugs: None were detected
* Request the feature for a non-existent assignment. 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 considered. 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 assignment 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 “leg” navigation attribute using the “DevTools”. Changed the value for a non-existing leg id (i.e. 999 or -1) or for a non-valid leg (i.e. one that is not published). Check that the application returned an authorization exception.
  + Detection of bugs: No bugs detected.
* Hack the “lastUpdate” read-only attribute using the “DevTools”. Check that the changed value does not persist.
  + 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 considered. 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 assignment is updated properly:
  + Detection of bugs: None were detected.

The hacking cases implemented include, among others:

* Requesting the feature for a Flight Assignment that has already been published (with both the member associated to it and another one). Check that an authorization exception is thrown.
  + Detection of bugs: None were detected
* Requesting the feature for an Assignment in draft mode that does not belong to the logged in member. Check that an authorization exception is thrown.
  + Detection of bugs: None were detected
* Request the feature for a non-existing Flight Assignment.
  + Detection of bugs: None were detected.
* Hack the “leg” navigation attribute using the “DevTools”. Changed the value for a non-existing leg id (i.e. 999 or -1) or for a non-valid leg (i.e. one that is not published). Check that the application returned an authorization exception.
  + Detection of bugs: No bugs detected.
* Hack the “lastUpdate” read-only attribute using the “DevTools”. Check that the changed value does not persist.
  + 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 considered. Said variations were taken from the “Sample-Data” file. For an assignment to be published, it must be associated with a leg that has not occurred, the member associated must have “AVAILABLE” status and cannot be assigned to simultaneous legs. So that tests with different crew members and legs were performed to cover these restrictions, checking that no cases lead to bugs.
  + Detection of bugs: None were detected.
* Submit a form with all valid data to see that the assignment is published properly:
  + Detection of bugs: None were detected.

The hacking cases implemented include, among others:

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

Delete:

The following safe cases were implemented:

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

The hacking cases include, among others:

* Requesting the delete feature for a non-existing assignment. Check that an authorization exception is thrown.
  + Detection of bugs: No bugs were found
* Request the feature for an already published assignment. 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 an assignment 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.

Show assigned Flight Crew Member:

The safe cases include:

* Request the feature for all crew members associated to an assignment in the sample data, by the same crew member, checking that everything renders properly.
  + Detection of bugs: No bugs were detected

The hacking cases implemented include:

* Request the feature for a crew member with a different realm. An authorization exception is thrown
  + Detection of bugs: None were detected
* Request the feature for a non-existent crew member. An authorization exception is thrown.
  + Detection of bugs: None were detected.

Show associated Leg:

The safe cases include:

* Request the feature for all legs associated to an assignment in the sample data, by the crew member that is assigned to that assignment, checking that everything renders properly.
  + Detection of bugs: No bugs were detected

The hacking cases implemented include:

* Request the feature for a leg to which the logged-in crew member is not assigned. An authorization exception is thrown
  + Detection of bugs: None were detected
* Request the feature for a non-existent leg. An authorization exception is thrown.
  + Detection of bugs: None were detected.

## Operations by Flight Crew Member on Activity Logs

List

The following safe cases have been performed:

* For each Flight Assignment, it has been checked that the listing of activity logs associated with said assignment is visible by the member associated with it.
  + Detection of bugs: No bugs were detected

The hack cases include:

* Trying to list the activity logs of a non-existing Flight Assignment. Check that an authorization exception is thrown.
  + Detection of bugs: No bugs were detected
* Trying to list the activity logs of a Flight Assignment with another Flight Crew Member than the one that is associated with it.
  + Detection of bugs: No bugs were detected.

Show

The following safe cases have been performed:

* For each Activity Log in the sample data, it has been checked that they can be correctly seen by their associated member.
  + Detection of bugs: None were detected

The following hack cases have been implemented, among others:

* Trying to show a non-existing activity log. It has been checked that an authorization exception is thrown.
  + Detection of bugs: None were found.
* Trying to show an activity log of an assignment by another member that the one associated. 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 considered. 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 activity log is created properly:
  + Detection of bugs: None were detected.

The hacking cases implemented include, among others:

* Requesting the feature for a Flight Assignment that does not belong to the logged in member. Check that an authorization exception is thrown.
  + Detection of bugs: None were detected
* Request the feature for a non-existing Flight Assignment.
  + Detection of bugs: None were detected.
* Hack the “registrationMoment” read-only attribute using the “DevTools”. Check that the changed value does not persist.
  + Detection of bugs: No bugs detected.

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 considered. 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 activity log is updated properly:
  + Detection of bugs: None were detected.

The hacking cases implemented include, among others:

* Requesting the feature for an Activity Log that has already been published (with both the member that is associated with it and another one). Check that an authorization exception is thrown.
  + Detection of bugs: None were detected
* Requesting the feature for an activity log in draft mode that does not belong to the logged in member. Check that an authorization exception is thrown.
  + Detection of bugs: None were detected
* Request the feature for a non-existing Activity Log.
  + Detection of bugs: None were detected.
* Hack the “registrationMoment” read-only attribute using the “DevTools”. Check that the changed value does not persist.
  + Detection of bugs: No bugs detected.

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 considered. Said variations were taken from the “Sample-Data” file. For an activity log to be published, its Flight Assignment must be published. So that tests with different assignments were performed to cover this restriction, checking that no cases lead to bugs.
  + 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 an Activity Log that has already been published (with both the member that is associated with it and another one). Check that an authorization exception is thrown.
  + Detection of bugs: None were detected
* Requesting the feature for an activity log in draft mode that does not belong to the logged in member. Check that an authorization exception is thrown.
  + Detection of bugs: None were detected
* Request the feature for a non existing Activity Log.
  + Detection of bugs: None were detected.
* Hack the “registrationMoment” read-only attribute using the “DevTools”. Check that the changed value does not persist.
  + Detection of bugs: No bugs detected.

Delete

The following safe cases were conducted:

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

The hacking cases include, among others:

* Requesting the delete feature for a non-existing activity log. Check that an authorization exception is thrown.
  + Detection of bugs: No bugs were found
* Request the feature for an already published activity log. 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 an activity log in draft mode with a different memberthan the one who created it. Check that it returns an authorization error.
  + Detection of bugs: No bugs were found.

# PERFORMANCE TESTING

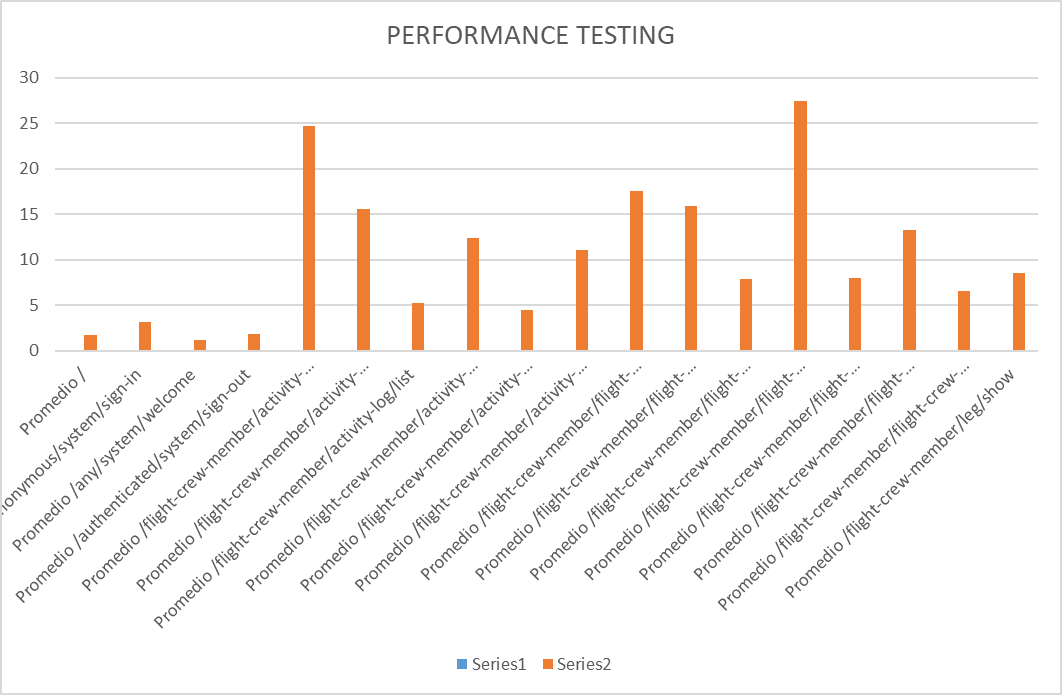
In this last section, we will evaluate the performance of the project by measuring the elapsed time taken to complete requests during functional testing. The objective is to assess how quickly the system responds under real conditions to determine which computer performs the best.

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

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

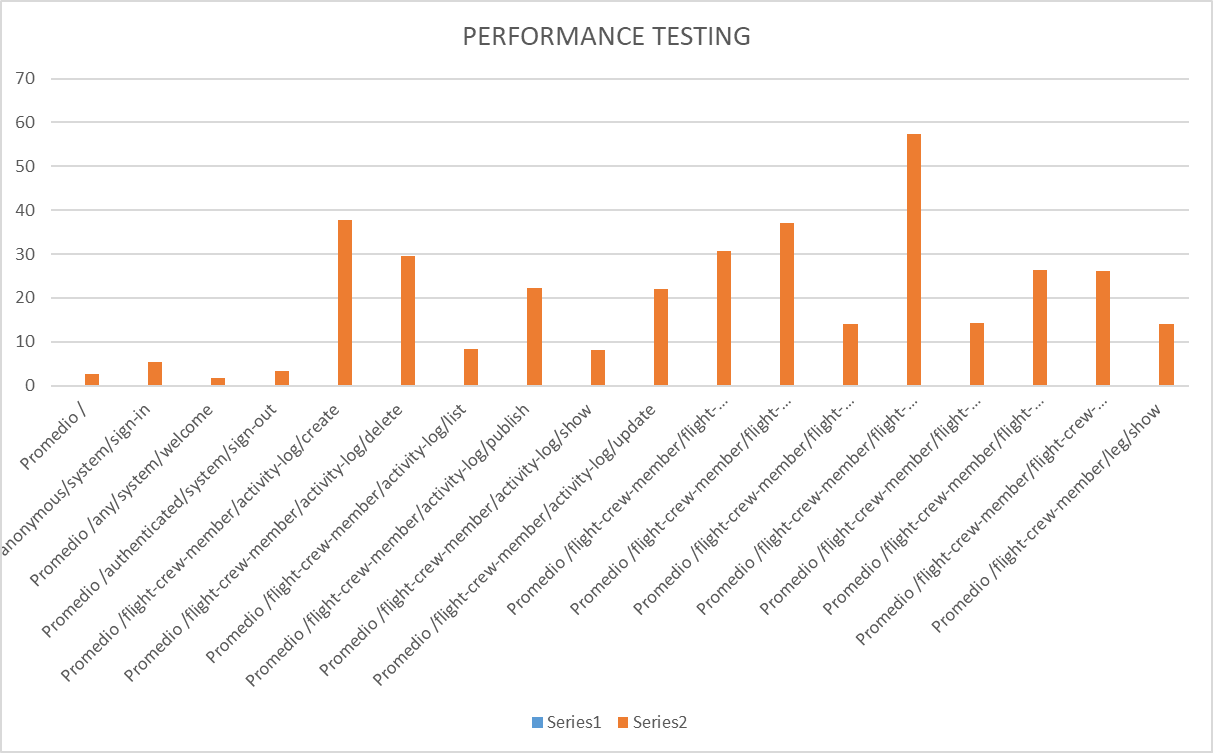
## Mean Confidence Interval

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 6.88 ms. As it can be appreciated, we see that the MIR is the flight assignment publish, whose average is of 27.47 ms.

Using the data analyzer extension from Excel, we obtained that the amplitude of the confidence interval at 95% is 0.43 ms. By removing and adding this value from the average, we obtain the confidence interval: [0.0064 s – 0.0073 s]



When doing the same with the MSI Pulse Gl76 PC, we this similar graph. In this case, the grand average stands at 12.39 ms. We can see the MIR is the same as the previous one, the flight assignment publish, with an average of 57.44 ms.

Using the data analyzer, we get that the amplitude of the confidence interval at 95% is 0.853 ms. We would obtain the following confidence interval: [0.011 s– 0.013s]

## 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:



As it can be clearly seen, the first computer is way faster than the second 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