Workgroup C1.067

Testing report

25/05/2025

Tudor Cristian Lacatus Cosma – [ionlac@alum.us.es](mailto:ionlac@alum.us.es)

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

Table of Contents

[EXECUTIVE SUMMARY: 2](#_Toc199108928)

[REVISION TABLE 3](#_Toc199108929)

[INTRODUCTION 4](#_Toc199108930)

[FUNCTIONAL TESTING 5](#_Toc199108931)

[Operations by Customers on Bookings 5](#_Toc199108932)

[Operations by Customers on Booking Records 8](#_Toc199108933)

[Operations by Customers on Passengers 10](#_Toc199108934)

[PERFORMANCE TESTING 14](#_Toc199108935)

[Mean Confidence Interval 14](#_Toc199108936)

[Contrasting information 15](#_Toc199108937)

[CONCLUSIONS 17](#_Toc199108938)

[BIBLIOGRAPHY 18](#_Toc199108939)

# EXECUTIVE SUMMARY:

This document provides a thorough and complete overview of the testing procedures used to validate the functionality and performance of features within Student #2's requirements. The primary goal is to ensure complete security within the application, in compliance with "Ley Orgánica 3/2018."

The functional testing section details the test procedures conducted for each feature available to airline customers, specifically regarding Bookings, Booking Records and Passengers. This testing involved recording and reproducing a variety of legal and hacking scenarios. Particular attention was paid to edge cases within established attribute ranges, input validation, navigation attributes, and access control. The lecturer confirmed in the last follow-up session that read-only attributes, being computed, were not susceptible to hacking, so these test cases were not documented.

The final section of this report focuses on performance testing. By calculating performance across two distinct hardware environments, this report establishes the necessary confidence intervals and compares execution times to identify the more powerful machine.

# REVISION TABLE

|  |  |  |
| --- | --- | --- |
| Revision Number | Date | Description |
| 1.0 | 25/05/2025 | Initial version of the document |
|  |  |  |
|  |  |  |

# INTRODUCTION

This document aims to present an analysis of the testing methodology applied to the application, specifically addressing the requirements assigned to Student #2. The content of this document is organized into two primary sections.

The initial section is dedicated to the functional testing process. It will encompass a comprehensive listing of all implemented test cases, categorized by feature. For every test case, a concise explanation of the executed tests will be given, along with an indication of any defects discovered.

The subsequent and final section will cover performance testing. This includes charts illustrating performance metrics, as well as a 95% confidence interval for the time required by the project to process the requests from the functional tests across two distinct hardware configurations. Ultimately, this section will conclude with the identification of the superior performing machine.

# FUNCTIONAL TESTING

## Operations by Customers on Bookings

List

Safe Scenarios:

* We accessed the list of bookings, systematically reviewing every page to confirm accurate rendering of entries.
  + Detection of bugs: No bugs were detected.

Hacking Scenarios:

* Attempts were made to access the booking listing feature from unauthorized realms (e.g., unauthenticated users or administrators). An authorization exception was successfully triggered.
  + Detection of bugs: No bugs were detected.

Show

Safe Scenarios:

* The "show" feature was requested for all bookings within the sample dataset by their respective creating customers. All elements were verified to render correctly.
  + Detection of bugs: No bugs were detected

Hacking scenarios:

* The feature was requested with a realm different from the one associated with the booking (e.g., a distinct customer or an unauthenticated user). An authorization exception was successfully thrown.
  + Detection of bugs: None were detected
* The feature was requested for a booking that did not exist. An authorization exception was successfully thrown.
  + Detection of bugs: None were detected.

Create

Safe scenarios:

* The creation form was submitted with all null values to ensure that no unexpected exceptions were generated.
  + Detection of bugs: No bugs were detected.
* For each form attribute, a comprehensive range of both valid and invalid data, sourced from the "Sample-Data" file, was considered and tested.
  + Detection of bugs: None were detected.
* A form with entirely valid data was submitted to confirm the correct creation of the flight.
  + Detection of bugs: None were detected.

Hacking scenarios:

* The feature was requested using alternative realms, and an authorization error was correctly returned.
  + Detection of bugs: None were detected
* The "flight" attribute was manipulated via "DevTools." Values were changed to non-existent flight IDs (e.g., 999 or -1), and also to flights that had already occurred (it's impossible to create a booking for a past flight), and flights in draft mode (creation is not possible for unpublished flights). The application correctly returned an authorization exception in these cases.
  + Detection of bugs: No bugs detected.

Update

Safe scenarios:

* An empty form was submitted to verify that no exceptions were thrown and that errors were correctly reported for the relevant attributes.
  + Detection of bugs: No bugs were detected.
* For each form attribute, a comprehensive range of both invalid and valid data, sourced from the "Sample-Data" file, was considered and tested.
  + Detection of bugs: None were detected.
* A form containing all valid data was submitted to confirm the successful update of the booking.
  + Detection of bugs: None were detected.

Hacking scenarios:

* The feature was requested for a booking that had already been published. An authorization exception was correctly thrown.
  + Detection of bugs: None were detected
* The feature was requested by a realm that did not own the booking (e.g., a different customer).
  + Detection of bugs: None were detected
* The feature was requested for a booking that did not exist (e.g., -1 or 9999).
  + Detection of bugs: None were detected
* The "flight" attribute was manipulated using "DevTools," changing its value to a non-existent flight ID (e.g., 9999 or -1). The application correctly returned an authorization exception.
  + Detection of bugs: No bugs detected.
* The "flight" attribute was manipulated using "DevTools," changing its value to a non-published flight. The application correctly returned an authorization exception.
  + Detection of bugs: No bugs detected.
* The "purchaseMoment" attribute was manipulated using "DevTools" to a custom value. The application correctly ignored this custom moment, as the purchase moment is computed at the time of publication using the current timestamp.
  + Detection of bugs: No bugs detected.

Publish:

Safe scenarios:

* An empty form was submitted to verify that no exceptions were thrown and that errors were correctly reported for the relevant attributes.
  + Detection of bugs: No bugs were detected.
* For each form attribute, a comprehensive range of both invalid and valid data, sourced from the "Sample-Data" file, was considered and tested.
  + Detection of bugs: None were detected.
* A form containing all valid data was submitted to confirm the successful update of the booking.
  + Detection of bugs: None were detected.
* The form was submitted with a booking without passengers (each booking requires at least one). A message was shown.
  + Detection of bugs: None were detected.

Hacking scenarios:

* The feature was requested for a booking that had already been published. An authorization exception was correctly thrown.
  + Detection of bugs: None were detected
* The feature was requested by a realm that did not own the booking (e.g., a different customer).
  + Detection of bugs: None were detected
* The feature was requested for a booking that did not exist (e.g., -1 or 9999).
  + Detection of bugs: None were detected
* The "flight" attribute was manipulated using "DevTools," changing its value to a non-existent flight ID (e.g., 9999 or -1). The application correctly returned an authorization exception.
  + Detection of bugs: No bugs detected.
* The "flight" attribute was manipulated using "DevTools," changing its value to a non-published flight. The application correctly returned an authorization exception.
  + Detection of bugs: No bugs detected.
* The "purchaseMoment" attribute was manipulated using "DevTools" to a custom value. The application correctly ignored this custom moment, as the purchase moment is computed at the time of publication using the current moment.
  + Detection of bugs: No bugs detected.

## Operations by Customers on Booking Records

List

Safe Scenarios:

* We accessed the list of bookings records of some bookings to ensure that the booking records are well rendered
  + Detection of bugs: No bugs were detected.

Hacking Scenarios:

* We attempted to access the booking record listing feature from unauthorized accounts (e.g., customers who did not own the booking). This successfully triggered an authorization exception.
  + Detection of bugs: No bugs were detected.
* Attempts were made to access booking records that did not exist (e.g., 9999 or -1). No bugs were detected, and the application behaved as expected. This successfully triggered an authorization exception.
  + Detection of bugs: No bugs were detected.

Show

Safe Scenarios:

* The "show" feature was requested for all bookings records belonging to the same booking. All elements were verified to render correctly.
  + Detection of bugs: No bugs were detected

Hacking scenarios:

* The feature was requested with a realm different from the one associated with the booking record (e.g., a distinct customer or an unauthenticated user). An authorization exception was successfully thrown.
  + Detection of bugs: None were detected
* The feature was requested for a booking record that did not exist. An authorization exception was successfully thrown.
  + Detection of bugs: None were detected.

Create

Safe scenarios:

* The creation form was submitted with no passenger to ensure that no unexpected exceptions were generated.
  + Detection of bugs: No bugs were detected.
* The form was submitted with a valid passenger to confirm the correct creation of the booking record.
  + Detection of bugs: None were detected.

Hacking scenarios:

* The feature was requested using realms not authorized to perform this action (e.g., a customer trying to add a passenger to a booking they don't own). An authorization error was correctly returned.
  + Detection of bugs: None were detected
* When this functionality is requested via a GET method, the system verifies whether the booking ID exists and, if it does, whether it belongs to the specified realm. If either of these conditions is not satisfied, an authorization error is returned.
  + Detection of bugs: None were detected
* The "passenger" attribute was manipulated using "DevTools," changing its value to a non-published passenger. The application correctly returned an authorization exception.
  + Detection of bugs: None were detected
* The "passenger" attribute was manipulated using "DevTools," changing its value to a non-existing passenger. The application correctly returned an authorization exception.
  + Detection of bugs: None were detected
* The "passenger" attribute was manipulated using "DevTools," changing its value to a passenger that does not belong to the current realm. The application correctly returned an authorization exception.
  + Detection of bugs: None were detected
* The "passenger" attribute was manipulated using "DevTools," changing its value to a passenger that it is already associated with the booking. The application correctly returned an authorization exception.
  + Detection of bugs: None were detected

Delete

Safe scenarios:

* When the delete is requested, it submits the booking record for deletion in order to confirm that the record has been successfully removed.
  + Detection of bugs: None were detected.

Hacking scenarios:

* The feature was requested for a booking record that had already been published. An authorization exception was correctly thrown.
  + Detection of bugs: None were detected
* The feature was requested by a realm that did not own the booking (e.g., a different customer).
  + Detection of bugs: None were detected
* The feature was requested for a booking record that did not exist (e.g., -1 or 9999).
  + Detection of bugs: None were detected
* The "passenger" attribute was manipulated using "DevTools," changing its value to a non-existent passenger ID (e.g., 9999 or -1). The application did not take into account this, since the delete service takes only into account the ID of the booking record.
  + Detection of bugs: No bugs detected.

## Operations by Customers on Passengers

List

Safe Scenarios:

* We accessed the list of passengers, systematically reviewing every page to confirm accurate rendering of entries.
  + Detection of bugs: No bugs were detected.

Hacking Scenarios:

* Attempts were made to access the passenger listing feature from unauthorized realms (e.g., unauthenticated users or administrators). An authorization exception was successfully triggered.
  + Detection of bugs: No bugs were detected.

Show

Safe Scenarios:

* The "show" feature was requested for all passengers within the sample dataset by their respective creating customers. All elements were verified to render correctly.
  + Detection of bugs: No bugs were detected

Hacking scenarios:

* The feature was requested with a realm different from the one associated with the passenger (e.g., a distinct customer or an unauthenticated user). An authorization exception was successfully thrown.
  + Detection of bugs: None were detected
* The feature was requested for a passenger that did not exist. An authorization exception was successfully thrown.
  + Detection of bugs: None were detected.

Create

Safe scenarios:

* The creation form was submitted with all null values to ensure that no unexpected exceptions were generated.
  + Detection of bugs: No bugs were detected.
* For each form attribute, a comprehensive range of both valid and invalid data, sourced from the "Sample-Data" file, was considered and tested.
  + Detection of bugs: None were detected.
* A form with entirely valid data was submitted to confirm the correct creation of the flight.
  + Detection of bugs: None were detected.

Hacking scenarios:

* The feature was requested using alternative realms, and an authorization error was correctly returned.
  + Detection of bugs: None were detected

Update

Safe scenarios:

* An empty form was submitted to verify that no exceptions were thrown and that errors were correctly reported for the relevant attributes.
  + Detection of bugs: No bugs were detected.
* For each form attribute, a comprehensive range of both invalid and valid data, sourced from the "Sample-Data" file, was considered and tested.
  + Detection of bugs: None were detected.
* A form containing all valid data was submitted to confirm the successful update of the passenger..
  + Detection of bugs: None were detected.

Hacking scenarios:

* The feature was requested for a passenger that had already been published. An authorization exception was correctly thrown.
  + Detection of bugs: None were detected
* The feature was requested by a realm that did not own the passenger (e.g., a different customer).
  + Detection of bugs: None were detected
* The feature was requested for a passenger that did not exist (e.g., -1 or 9999).
  + Detection of bugs: None were detected

Publish:

Safe scenarios:

* An empty form was submitted to verify that no exceptions were thrown and that errors were correctly reported for the relevant attributes.
  + Detection of bugs: No bugs were detected.
* For each form attribute, a comprehensive range of both invalid and valid data, sourced from the "Sample-Data" file, was considered and tested.
  + Detection of bugs: None were detected.
* A form containing all valid data was submitted to confirm the successful update of the booking.
  + Detection of bugs: None were detected.

Hacking scenarios:

* The feature was requested for a passenger that had already been published. An authorization exception was correctly thrown.
  + Detection of bugs: None were detected
* The feature was requested by a realm that did not own the passenger (e.g., a different customer).
  + Detection of bugs: None were detected
* The feature was requested for a passenger that did not exist (e.g., -1 or 9999).
  + Detection of bugs: None were detected

# PERFORMANCE TESTING

In this final section, we will evaluate the system's performance by measuring the elapsed time required to complete requests during functional testing. The aim is to determine the responsiveness of the application under realistic conditions and identify which device delivers superior performance.

To obtain the necessary data, we will execute the complete suite of functional tests related to the manager-specific features. The tests will be conducted on the following machines:

* MSI Pulse GL76: 16 GB RAM, 1 TB SSD
* Lenovo Ideapad 5: 32 GB RAM, 512 GB SSD

The Lenovo Ideapad 5 it is more likely to have better results, since it is a newer computer with better hardware.

## Mean Confidence Interval

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

#### MSI Pulse GL76

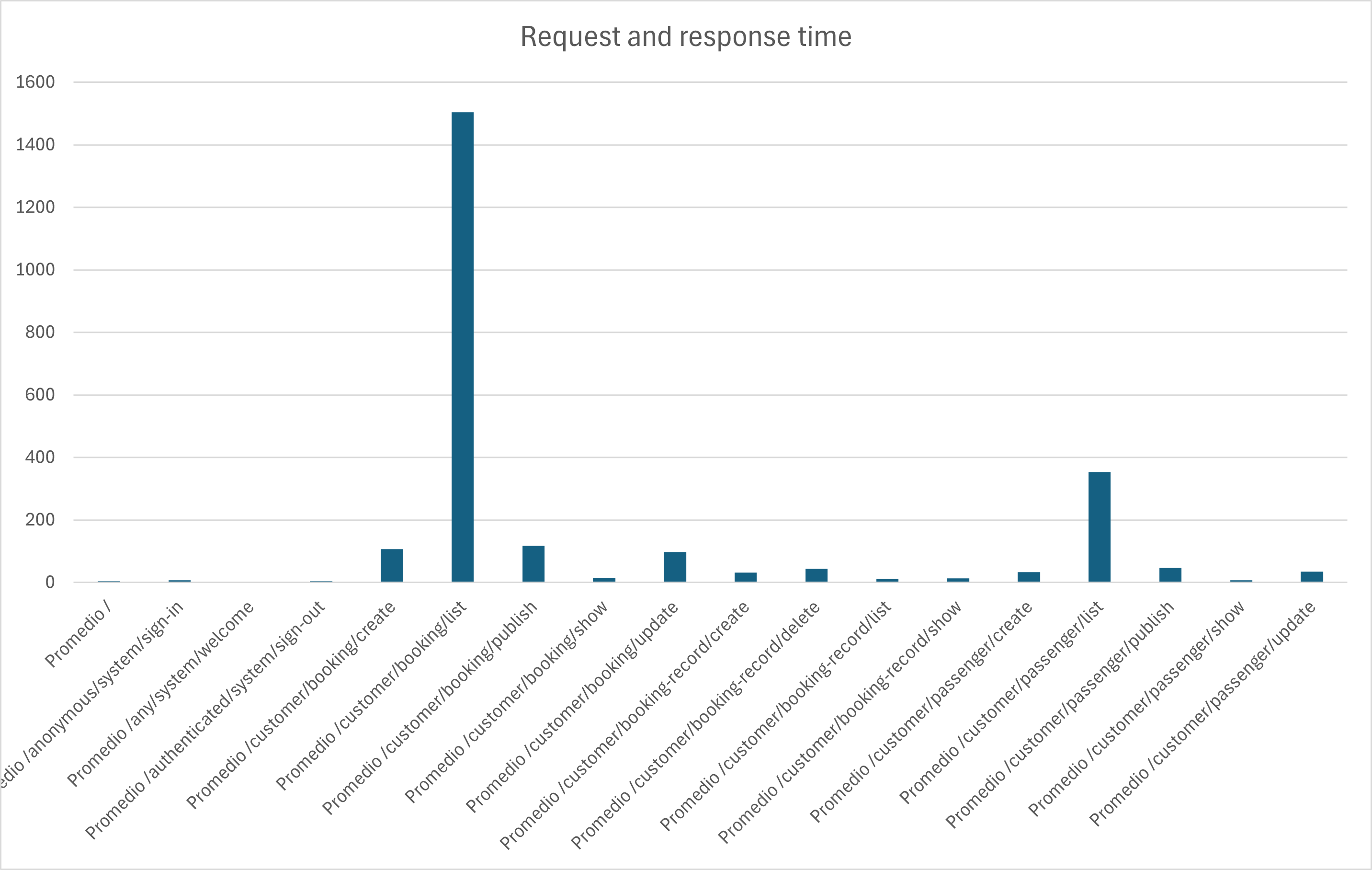
Gráfico, Gráfico en cascada

El contenido generado por IA puede ser incorrecto.

For the first computer, we have obtained a grand average of 172.30 ms. As it can be appreciated, we see that the MIR is the booking list, whose average is of 1446,33 ms.

Using the data analyzer extension from Excel, we obtain that the amplitude of the confidence interval at 95% is 26,4 ms. By removing and adding this value from the average, we obtain the confidence interval: [0.145 s – 0.198 s]

#### Lenovo Ideapad 5

When doing the same with the Lenovo Ideapad 5, we obtained a similar graph. In this case, the grand average stands at 163,05 ms. The apparent MIR is the booking listing feature as in the previous, with an average of 1469,59 ms.

Using the data analyzer, we get that the amplitude of the confidence interval at 95% is 26,12 ms. We would obtain the following confidence interval: [0.136 s – 0.189s]

## Contrasting information

Given the previous results, we can induce that the computer with the best performance is the Lenovo Ideapad 5, as in general, all its average times are lower (but the difference is minimal). Let us use the Z-Test with alpha 0.05 to verify our hypothesis.  
Once the Z-Test has been conducted, the results are the following:

Tabla

El contenido generado por IA puede ser incorrecto.

As can be appreciated, the p-value, which is 0.62, is in interval (0.05, 1.00], then there is no significant improvement between each other and the differences amongst the times are not significantly different since they are globally the same.

# CONCLUSIONS

This document serves as a formal record of the results obtained from the execution of test cases, with the objective of enabling future reference. The recorded analysis offers a detailed evaluation of each test case, aiming to detect potential bugs in the code and to support informed decision-making to improve the system and ensure protection against possible hacking attempts.

Each test case was documented systematically and organized by feature. Any bugs encountered during the process have been duly reported. This structure ensures transparency and traceability of the testing procedure.

The performance analysis conducted on two different computers demonstrates that the Lenovo device consistently outperformed the MSI. As anticipated, both systems exhibited similar behavior regarding request distribution. Nevertheless, the MSI showed an average MIR that was 20 ms faster. Given that the average MIR difference between the two machines in the initial case is minimal, this may be interpreted as an isolated or punctual result.

This testing report presents clear and concise descriptions of the procedures followed and results obtained, with the aim of serving as a solid foundation for future quality assurance efforts.

# BIBLIOGRAPHY

Intentionally blank