University of Sevilla

Higher Technical School of Computer Engineering

**D04 – Student #2 Testing Report**

**Un dibujo de una persona

El contenido generado por IA puede ser incorrecto.**

Degree in Computer Engineering - Software Engineering

Desing and Testing II

Course 2024 – 2025

|  |  |
| --- | --- |
| **Date** | **Version** |
| 15/10/2025 | v1.0 |

|  |  |
| --- | --- |
| **Practice Group: C1.040** | |
| **Members** | **Email** |
| José María Portela Huerta | josporhue@alum.us.es |

**Repository:** https://github.com/DP2-2025-C1-040/Acme-ANS

Table of contents

[1. Revision Table 3](#_Toc199178204)

[2. Execution summary 4](#_Toc199178205)

[3. Introduction 5](#_Toc199178206)

[4. Functional Testing 6](#_Toc199178207)

[5. Performance testing 9](#_Toc199178208)

[6. Conclusions 12](#_Toc199178209)

[7. Bibliography 13](#_Toc199178210)

# 1. Revision Table

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Delivery |
| 15/10/2025 | V1.0 | First report’s versión for C3 | C3 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 2. Execution summary

For this testing document, each of the implementations present in the mandatory requirements of the D04 deliverable have been developed and explained.

Both the system performance and the functional performance of each and every one of the requested functions have been evaluated, in this case, those related to ‘booking’ and ‘passenger’. The methodology provided in Lesson 04, ‘S01 - Formal testing’ and ‘S02 - Performance testing’ was followed.

The system performs robustly in terms of functionality, although there are areas that could use a little attention.

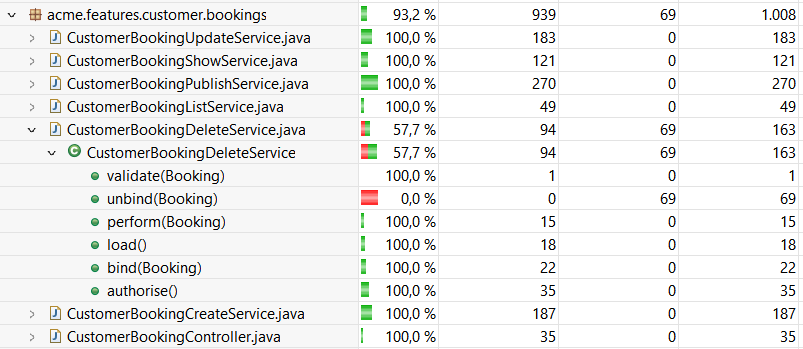
# 3. Introduction

This document is divided into two distinct sections:

1. Functional testing: a list of implemented test cases, grouped by functionality, will be presented. For each one, a description and an indication of how effective it is in detecting errors will be given. For effectiveness, the code coverage and a screenshot of the code will be used to check that all possible decisions have been tested during the execution of the program to avoid the existence of errors.
2. Performance testing: appropriate graphs and a 95% confidence interval will be provided for the time taken for the requests in the tests on two different computers. In addition, after the tests on the different computers, it will be indicated which of these is the most powerful and offers the best performance.

# 4. Functional Testing

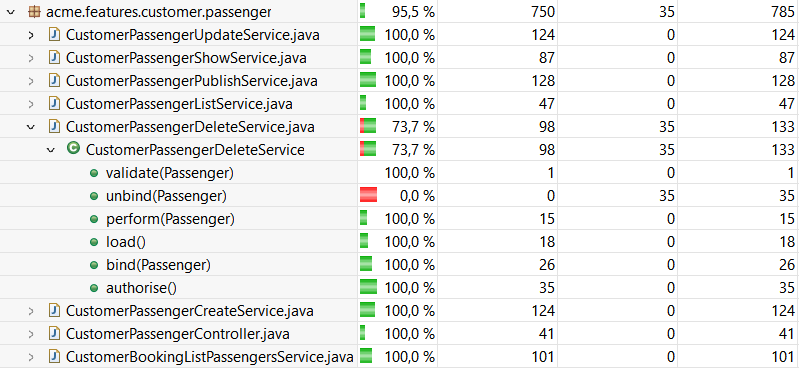
Tests cases related to bookings:



* Create booking:
  + Description: the restrictions of all the fields of the booking creation form are tested with values related to positive, negative and hacking cases.
  + Coverage: 100%
  + Effectiveness: perfect coverage.
* Delete booking:
  + Description: the elimination of a booking is tested, with positive, negative and hacking cases. In addition, its associated activity logs are checked for deletion.
  + Coverage: 57.7%
  + Effectiveness: in this case, the coverage is 57.7% due to the unbind method, as in no case this method is entered but it is necessary for the other services.
* List bookings:
  + Description: the list of bookings associated with the currently logged-in customer is tested. The different hacking cases are also verified.
  + Coverage: 100%
  + Effectiveness: perfect coverage.
* Publish booking:
  + Description: the publication of a booking is tested. It is checked that if there aren’t passengers associated, or all of them are not published, it is not posible to publish the booking.
  + Coverage: 100%
  + Effectiveness: perfect coverage.
* Show booking:
  + Description: the display of a booking, for its corresponding customer and associated flight, is tested. In addition, the different hacking cases are tested.
  + Coverage: 100%
  + Effectiveness: perfect coverage.
* Update booking:
  + Description: the constraints of all fields of a booking update form are tested with values for positive, negative and hacking cases.
  + Coverage: 100%
  + Effectiveness: perfect coverage.

In short, the only red thing that can be seen is the unbind method of the delete service, since it is not executed at any time, as it is impossible to enter this condition. For this reason and for having an average coverage of 93.2% in booking, it is considered that the existence of potential failures or bugs is negligible.

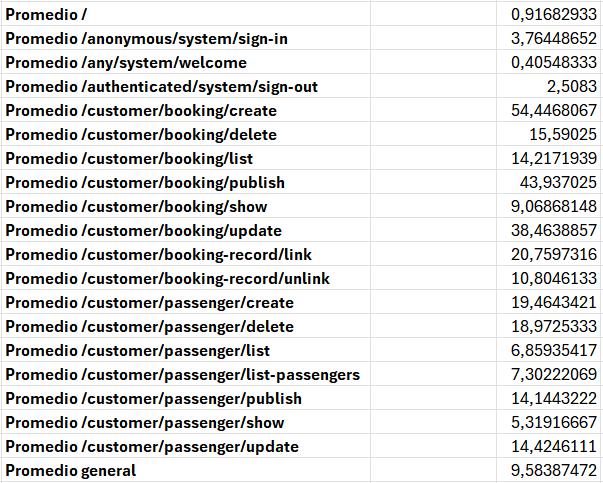
Test cases relating to passengers:

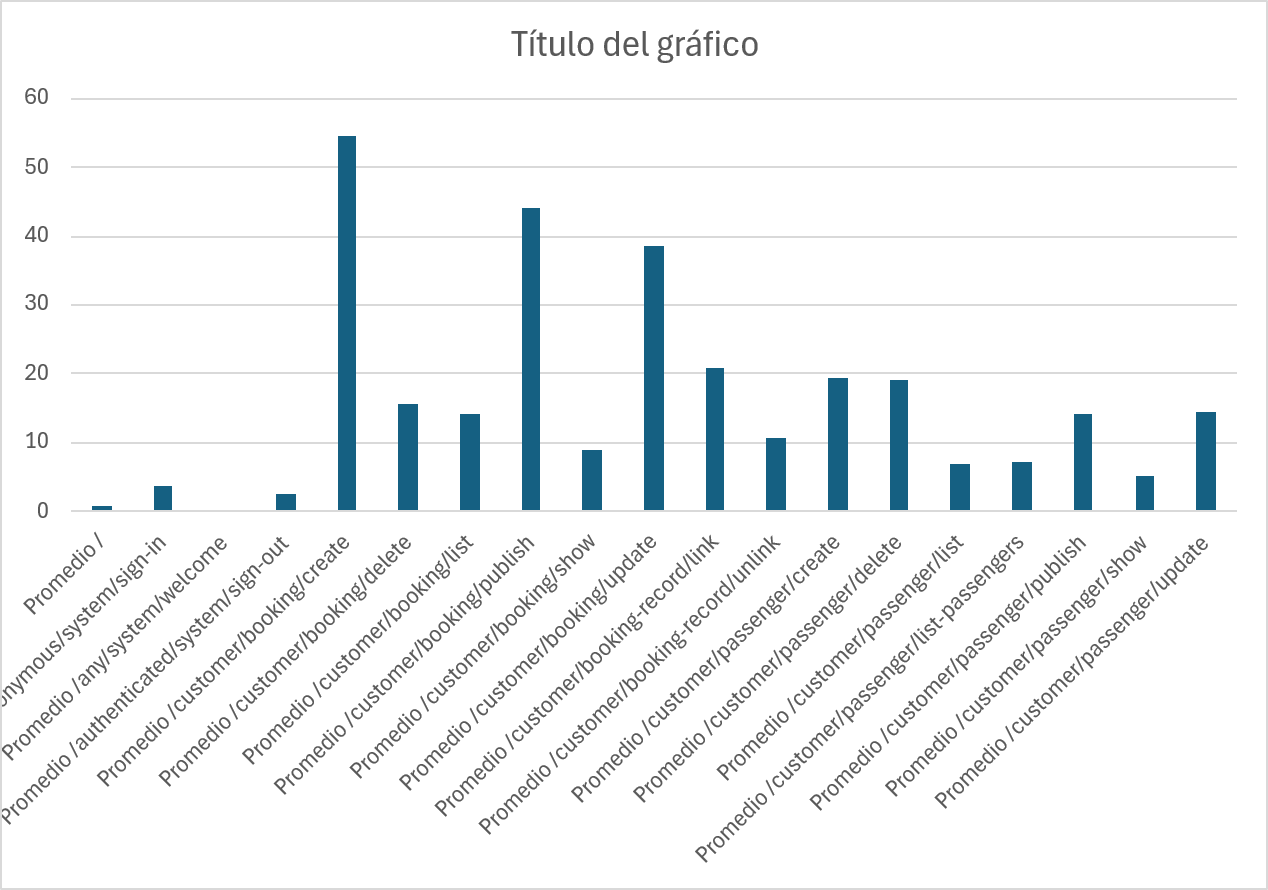


* Create passenger:
  + Description: the restrictions of all the fields of the passenger creation form are tested with values related to positive, negative and hacking cases.
  + Coverage: 100%
  + Effectiveness: perfect coverage.
* Delete passenger:
  + Description: the elimination of a booking is tested, with positive, negative and hacking cases.
  + Coverage: 73.7%
  + Effectiveness: coverage is around 73.7% due to the unbind method, as in the delete booking. This is not a cause for concern as the unbind method is the same as in other services and is already proven to work correctly in those services.
* List passengers:
  + Description: the list of passengers associated with the currently logged-in customer is tested. The different hacking cases are also verified.
  + Coverage: 100%
  + Effectiveness: perfect coverage.
* Publish passenger:
  + Description: the publication of a passenger is tested.
  + Coverage: 100%
  + Effectiveness: perfect coverage.
* Show passenger:
  + Description: the display of a passenger, with its corresponding booking, is tested. In addition, the different hacking cases are tested.
  + Coverage: 100%
  + Effectiveness: perfect coverage.
* Update passenger:
  + Description: the constraints of all fields of a passenger update form are tested with values for positive, negative and hacking cases.
  + Coverage: 100%
  + Effectiveness: perfect coverage.

In short, the only red thing that can be seen is the unbind method of the delete service, since it is not executed at any time, as it is impossible to enter this condition. For all these reasons and for having an average coverage of 95.5% in passenger, it is considered that the existence of potential failures or bugs is negligible.

# 5. Performance testing

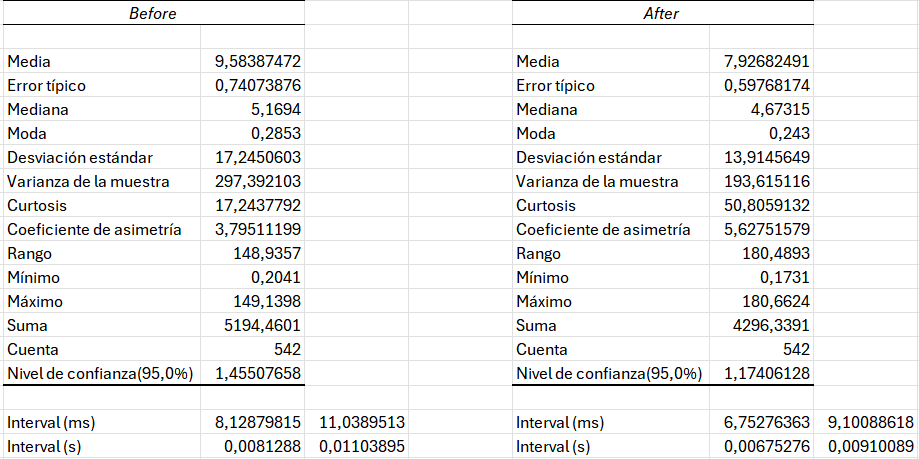
After performing the set of tests for the appropriate functionalities, all the steps shown in ‘S02 - Performance testing’ have been carried out, obtaining the following results:



As can be seen in the images above, the average time taken by the system to perform a request is approximately 9.58 ms, i.e. 0.01 seconds, which is quite fast.

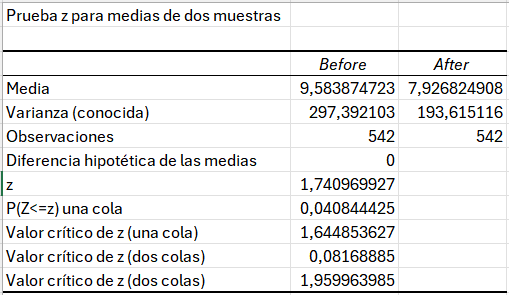
Furthermore, the bar chart clearly shows that the requests that take the longest time are those that handle a larger amount of data and validations, specifically those related to the creation, update and publication of a booking.

Although the performance tests without indexes met the established requirements (with an average of 10 ms), it was decided to add indexes on the most queried fields. The appropriate comparisons are made of the different values obtained in each test and we obtain:

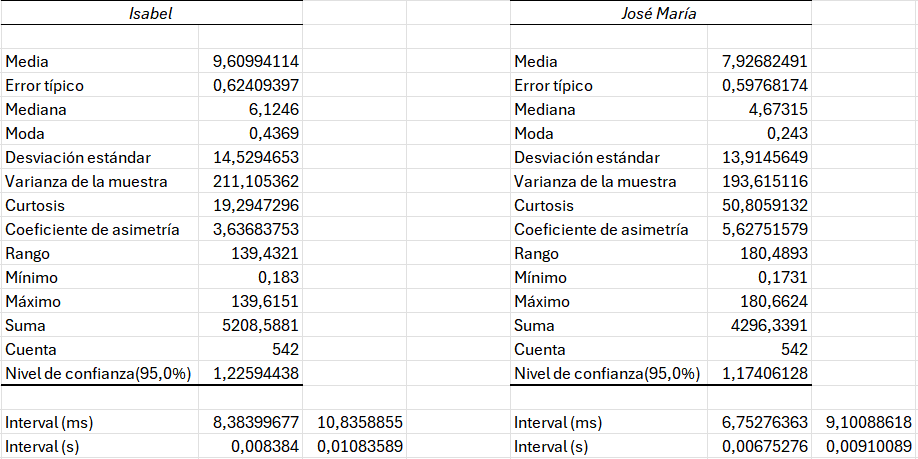


The indexes reduced the average to 7.92 ms (20.9% faster), which improves significately the stability of the system and prepares it for higher data volume scenarios.

To determine whether the average times before and after the changes can be considered the same or not, a z-test was performed with the following results:

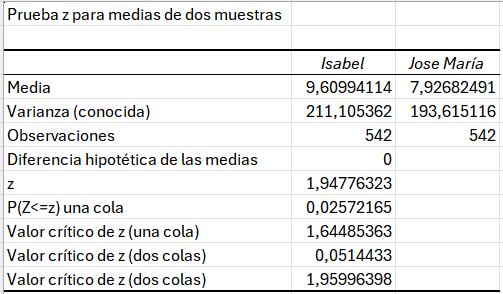


Given this two-tailed p-value result (0,0816), we can conclude that, being very close to de α-value 0,005, the changes doesn’t conclude a better performance in the application.



The performance of the system will now be compared on two different computers. The second (José María’s) computer will be the one on which all the above tests have been performed and the first (Isabel) will be another team member's computer. Here are the results:

We can see the difference in results.



As can be seen, there are little differences among the different computers, keeping the z value of 0,0514 very close to the α-value: 0,05.

# 6. Conclusions

After creating this testing report, it has been concluded that this phase of the project lifecycle is vital. Verifying that all developed functions work correctly and are thoroughly checked to minimise errors or failures, as well as ensuring that performance is optimised as much as possible, are fundamental aspects for the client. A thoroughly tested system allows the end-user to use it quickly and intuitively, avoiding problems that can detract from their experience.

In addition, a meticulous testing process contributes to customer satisfaction and product reputation, ensuring that quality and efficiency expectations are consistently met.

# 7. Bibliography

* 06 Annexes - Material provided in the subject Design and Testing II by the University of Seville.
* L04 - S01 - Formal testing - Material provided in the subject Design and Testing II by the University of Seville.
* L04 - S02 - Performance testing - Material provided in the subject Design and Testing II by the University of Seville.