**INFORME DE PRUEBAS**



**Grado en Ingeniería Informática – Ingeniería del Software**

**Diseño y Pruebas 2**

**Curso 2023-2024**

Lucía Campos Díez

### Índice

1. [Executive summary 4](#_Toc28482)

[2. Table Review 5](#_Toc28483)

[3. Introduction 6](#_Toc28484)

[4. Contents 7](#_Toc28485)

[5.1 Funcionalm Testing 7](#_Toc28486)

[5.2 Performance Testing 14](#_Toc28487)

[6. Conclusions 17](#_Toc28488)

[**7. Bibliography** 18](#_Toc28489)

# 

# Executive summary

Testing is a fundamental element in software development and maintenance, helping development team members verify the proper functioning of the system and detect errors.  
This document presents a report on the testing carried out by Student 5, organized into sections on functional testing and performance testing; it also includes an analysis of the test coverage.

# Review Table

|  |  |  |
| --- | --- | --- |
| **Version** | **Description** | **Date** |
| v1.0 | Initial Version | 02/07/2025 |
| v2.0 | Initial Version | 03/07/2025 |

# Introduction

In the first section, focused on functional testing, the implemented test cases will be discussed grouped by implementation, including their effectiveness in detecting errors.  
The second section addresses the system’s performance through graphs and statistical analysis of the request response times, using 95% confidence intervals before and after refactoring with corresponding metrics, along with a hypothesis test based on the analysis.

# Contents

## Funcional Testing

Below is a brief summary of the implemented test cases, with additional explanations provided for those tests that uncovered errors in the functionality implementation. Please note that in cases where a test failed due to a detected error or produced an unexpected result, the error was fixed and the test was repeated to ensure proper operation.

### InvolvedInNuevo

* **create1Nuevo.safe y create2Nuevo.safe**: Both tests were created to cover all the possible requests that technicians might perform, potentially triggering validation errors. The first test case considered is when all fields are left blank; from there, each field is gradually filled with invalid values that do not meet the form's constraints. These constraints are communicated through small warning messages. In this case, neither of the two dropdowns can be left unselected, and the system notifies whether the input already exists or not.
* **createNuevo.**hack: This test was created to handle cases in which a relationship between a record and a task is created illegally. These cases include changes in the ID received through the form, either from the relationship itself, the selected record, or the selected task, using invalid values (letters or symbols), or in situations that are not allowed (e.g., the selected record is already published, or the selected task belongs to another technician and is not published).
* **deleteNuevo.**hack: This test was created for cases in which a relationship between a record and a task is deleted illegally. These include changes in the ID received in the form, whether from the relationship, the record, or the task, using invalid values (letters or symbols), or in situations that are not allowed (e.g., the selected record is already published).
* **deleteNuevo.**safe: This test was created to cover all the possible requests that technicians might perform, potentially triggering validation errors. The first test case considered is when all fields are left blank; from there, each field is gradually filled with invalid values that do not meet the form's constraints. These constraints are communicated through small warning messages. In this case, neither of the two dropdowns can be left unselected.
* **final.**safe y **final2.**safe: Both tests were created to help ensure full coverage of the three entities due to last-minute changes made by other team members in the CSV files.
* **showNuevo.**hack: This test was created to handle the possibility of someone trying to access a relationship through the URL in the address bar that does not belong to them (i.e., the record is not owned by the user).
* **updateNuevo.**hack: This test was created for cases in which a relationship between a record and a task is updated illegally. These include changes in the ID received through the form, whether from the relationship, the record, or the task, using invalid values (letters or symbols), or in situations that are not allowed (e.g., the selected record is already published or does not belong to the user, the task does not belong to the user and is not published, or the relationship has already been previously saved in the database).
* **updateNuevo.**safe: This test was created to cover all the possible requests that technicians might perform, potentially triggering validation errors. The first test case considered is when all fields are left blank; from there, each field is gradually filled with invalid values that do not meet the form's constraints. These constraints are communicated through small warning messages. In this case, neither of the two dropdowns can be left unselected.

### MaintanenceRecord

* **create1Nuevo.hack y create2Nuevo.hack:** Both tests were created for cases in which a record is created illegally. These cases include modifications to the ID received in the form, either for the record or for the selected aircraft, using invalid values (letters or symbols). Tests were also conducted by changing enumerated values and submitting data that should not be accepted by the form, such as setting a past date for a field that must be in the future. Since notes are optional, they may be left blank. Additionally, the test verifies that the creation date of the record cannot be altered manually.
* **deleteNuevo.hack:** This test was created for cases in which a record is deleted illegally. These cases include changes in the ID received through the form when the record either does not exist, does not belong to the user, or has already been published.
* **deleteNuevo.safe:** This test was created to cover all the possible requests that a technician might make, potentially triggering validation errors. The first test considered is leaving all fields blank; from there, each field is progressively filled with invalid values that do not meet the form’s constraints. These constraints are communicated through small warning messages. In this case, neither of the two dropdowns can be left unselected, the date must be set in the future, and the monetary amount must follow the correct format.
* **nuevoCreate.safe:** This test was created to cover all the possible requests that a technician might make, potentially triggering validation errors. The first test considered is leaving all fields blank; from there, each field is progressively filled with invalid values that do not meet the form’s constraints. These constraints are communicated through small warning messages. In this case, neither of the two dropdowns can be left unselected, the date must be set in the future, and the monetary amount must follow the correct format.
* **publishNuevo.hack:** This test was created for cases in which a record is published illegally. These cases include changes in the ID received through the form (record does not exist, does not belong to the user, or is already published), attempts to change the creation date, or malicious attempts to manipulate aircraft IDs. It also checks that a record cannot be published unless it contains at least one published task, or if all the associated tasks are published.
* publishNuevo.safe: This test was created to cover all the possible requests that a technician might make, potentially triggering validation errors. The first test considered is leaving all fields blank; from there, each field is progressively filled with invalid values that do not meet the form’s constraints. These constraints are communicated through small warning messages. In this case, neither of the two dropdowns can be left unselected, the date must be set in the future, and the monetary amount must follow the correct format.
* **showNuevo.hack: T**his test was designed considering the possibility that someone might try to access a record that does not belong to them by manipulating the URL in the browser’s address bar.
* **updateNuevo.hack:** This test was created for cases in which a record is updated illegally. These cases include changes to the ID received in the form (record does not exist, does not belong to the user, or is already published), attempts to change the creation date, or malicious modifications of aircraft IDs.
* **updateNuevo.safe:** This test was created to cover all the possible requests that a technician might make, potentially triggering validation errors. The first test considered is leaving all fields blank; from there, each field is progressively filled with invalid values that do not meet the form’s constraints. These constraints are communicated through small warning messages. In this case, neither of the two dropdowns can be left unselected, the date must be set in the future, and the monetary amount must follow the correct format.

### Task

* **create1Nuevo.hack y create2Nuevo.hack:** Both tests were created for cases in which a task is created illegally. These include changes to the ID received in the form using invalid values (letters or symbols). Additionally, the test explores modifications to enumerated values.
* **createNuevo.safe:** This test was created to cover all the possible requests that technicians might make, potentially triggering validation errors. The first test considered is leaving all fields blank; from there, each field is gradually filled with invalid values that do not meet the form’s constraints. These constraints are communicated through small warning messages.
* **deleteNuevo.hack:** This test was created for cases in which a task is deleted illegally. These include changes to the ID received in the form when the task does not exist, does not belong to the user, or has already been published.
* **deleteNuevo.safe:** This test was created to cover all the possible requests that technicians might make, potentially triggering validation errors. The first test considered is leaving all fields blank; from there, each field is gradually filled with invalid values that do not meet the form’s constraints. These constraints are communicated through small warning messages.
* **publishNuevo.hack:** This test was created for cases in which a task is published illegally. These include changes to the ID received in the form when the task does not exist, does not belong to the user, or has already been published.
* **publishNuevo.safe:** This test was created to cover all the possible requests that technicians might make, potentially triggering validation errors. The first test considered is leaving all fields blank; from there, each field is gradually filled with invalid values that do not meet the form’s constraints. These constraints are communicated through small warning messages.
* **showNuevo.hack:** This test was created to handle the possibility that someone might try to access a task that does not belong to them by manipulating the URL in the browser’s address bar.
* **updateNuevo.hack:** This test was created for cases in which a task is updated or published illegally. These include changes to the ID received in the form when the task does not exist, does not belong to the user, or has already been published.
* **updateNuevo.safe: T**his test was created to cover all the possible requests that technicians might make, potentially triggering validation errors. The first test considered is leaving all fields blank; from there, each field is gradually filled with invalid values that do not meet the form’s constraints. These constraints are communicated through small warning messages. In this case, the dropdown cannot be left unselected, and care must be taken with the allowed range of each attribute.

## Performance Testing

Pre-refactoring tests with metrics

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

El contenido generado por IA puede ser incorrecto.

Post-refactoring tests with metrics

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

El contenido generado por IA puede ser incorrecto.

It can be observed by comparing both graphs that the requests requiring the most time are the update of the entity maintenanteRecord.

Additionally, it is evident that, overall, before refactoring, the requests took longer to process compared to after the refactoring, where the average response time is significantly lower. However, the operations that previously took the most time now take slightly longer.

The confidence intervals with a 95% confidence level are (29, 34) before refactoring and (32, 38) after refactoring.

**Hipótesis de contraste**

Tabla

El contenido generado por IA puede ser incorrecto.

With this results we can say that the use of indixes makes the performance worse because

# Conclusions

The report includes all the tests carried out by Student 5, which have been useful for detecting some errors in the code and verifying its correct functionality, as well as for measuring performance. This has allowed us to draw conclusions about the efficiency of our system in handling requests. These were the latest results obtained in the most recent replay carried out:

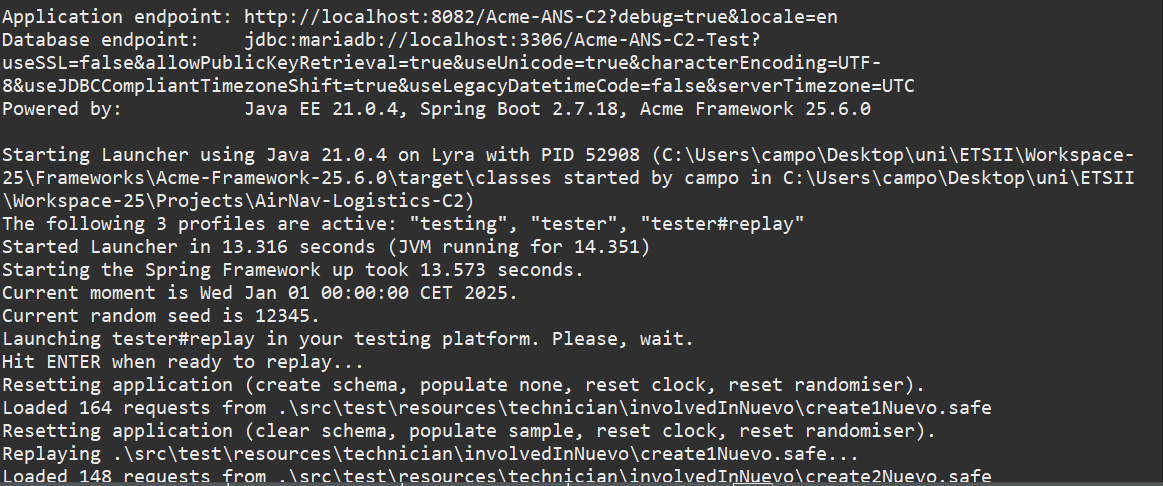


Imagen que contiene texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

Imagen que contiene texto, sostener, hombre

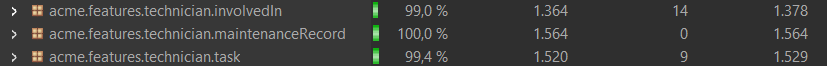
El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

And the final coverage percentage obtained:

# **Bibliography**

Intentionally in blank.