# **Testing Report**



|  |
| --- |
| **Group:** C1.031 |
| **Repository:** https://github.com/aaronma300604/DP2-C01-31 |
| **Members:**  **Student #1:** López Ramos, Daniel ([danlopram@alum.us.es](mailto:danlopram@alum.us.es))  **Student #2:** Ángel Postigo, Estrella ([estangpos@alum.us.es](mailto:estangpos@alum.us.es))  **Student #3:** Miranda Balastegui, Iván ([ivamirbal@alum.us.es](mailto:ivamirbal@alum.us.es))  **Student #4:** Terrón Hernández, Diego ([dieterher@alum.us.es](mailto:dieterher@alum.us.es))  **Student #5:** Mayoral Ansias, Aarón ([aarmayans@alum.us.es](mailto:aarmayans@alum.us.es)) |
| **Date:** Seville May 20, 2025 |

Table of Contents

[**Testing Report** 1](#_Toc199008524)

[**Revision Table** 3](#_Toc199008525)

[**Introduction** 3](#_Toc199008526)

[**Contents** 3](#_Toc199008527)

[**Functional Testing** 3](#_Toc199008528)

[**Performance Testing** 5](#_Toc199008529)

[**Conclusion** 10](#_Toc199008530)

[**Bibliography** 10](#_Toc199008531)

# **Revision Table**

|  |  |  |
| --- | --- | --- |
| **Revision Number** | **Date** | **Description** |
| 1.0 | 25/05/2025 | Intial version – all sections added |

# **Introduction**

The purpose of this document is to provide an analysis of the various tests that have been carried out to assess code coverage related to Student 5, as well as an analysis of the system's performance with respect to the requirements associated with said student. In order to do so, statistics and graphs regarding the response time will be provided. Also, a performance comparison between two different computers is going to be displayed, with its corresponding analysis and z-test.

# **Contents**

## **Functional Testing**

The following is a list of functional tests performed over the services for the Maintenance Record entity:

* create-mr-0.safe and create-mr-1.safe: These tests attempt to create Maintenance Records with invalid data and finally create a valid entity. For the text fields, the invalid data consists of too short and long texts. In these fields specifically, valid but malicious data, such script and SQL injections are also tested. For the numeric fields, values over their respective maximum and under their respective minimum are tested. For Date fields, too small and large dates are tested. Also, as the next inspection has to be after the maintenance record date, invalid value pairs have been introduced.
* delete-mr.safe: These tests access to the /delete URL in order to make a GET request and execute the unbind method. Also, they delete valid maintenance records of the database, with and without tasks associated.
* list-mr-2.safe/ showlist-mr.safe: These tests list and show details of the maintenance records, distinguishing between self and published records.
* publish-mr.safe/publish-mr-2.safe: These tests try to publish not publishable records which have either invalid data, no tasks or not published tasks linked. Finally, they publish a valid, publishable record. For the invalid data, the same data as in the create tests has been used.
* update-mr-2.safe/ update-mr.safe: These tests attempt to update maintenance record with invalid data. The same data as in previous .safe tests has been used. Finally, they update a record with valid values.
* get-mr.hack/ get-mr-2.hack/get-mr-3.hack/ delete-mr-get.hack: These tests are general get hacking tests. They try to access maintenance records which cannot be accessed by changing the URL. These include: show not published nor own records, update/delete/publish already published records, update/delete/publish records that belong to another technician and update/delete/publish non existing maintenance records.
* create-mr-post-v2.hack/ delete-mr-post.hack/ publish-mr.hack/update-mr-post.hack: These tests try to create, publish, delete and update records with invalid aircrafts. As any aircraft is susceptible to maintenance, even inactive ones, only non-existing aircrafts are considered to cause an illegal request.

Coverage for the Maintenance record services is 98.6%.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

With a few exceptions, all lines have been completely executed. These exceptions are:



Lines which only execute when no data is present in the database in the unbind methods. As the tester populates the database, these lines in the unbind method do not fully execute.

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Unreachable if branches in authorise methods. The first if statement cannot reach the branch where the conditions evaluate to true && false, since when an aircraftId is 0, aircraft is always null. Regarding the else if clause, it never should be evaluated to true, since all existing aircrafts are available for a technician to create a MaintenanceRecord of. However, this is kept as a double check in case something goes wrong.



This line in the validate method of the publish service. The record being validated will always be draftMode, since it is being checked in the authorise method. Since, this line cannot fully execute.

Following, the tests for the Task entity:

-create-t.safe: As in the maintenance record version, these tests aim to try and create a task containing invalid data. However, these are much simpler, since Task only has numeric and text fields. The same invalid data as in the maintenance records’ same type fields have been tested.

-delete-t.safe: These tests delete a task and execute the /delete unbind for this entity.

-list-t.safe: These tests list tasks under all conditions. These include: published plus own tasks, own tasks only and tasks assigned to a maintenance record.

-publish-t.safe.:These try to publish tasks with invalid data and, finally publish a correct task.

-update-t.safe: These try to update a task with invalid data. Finally, a proper update is performed.

As the show method is so simple for tasks, it does not need a separate test, since it gets fully executed while running the above ones.

-update-t-get.hack/show-t-get.hack/publish-t-get.hack/delete-t-get.hack/list-t-get.hack: These tests try to access and modify tasks where for which the user does not have permission. In the case of the list tests, a variety of parameters are used in the URI to try and access the tasks listed for a record the user is not allowed to see nor edit.

Coverage for Task related services is 100%:

Tabla

El contenido generado por IA puede ser incorrecto.

All lines in these services have been completely executed, as Taks is much simpler than Maintenance Record and does not need complex double checks.

Finally, the tests for the Involves entity (the middle entity between Task and Maintenance record), are the following:

-create-i.safe: These create a valid Involves. As the only thing in the form for creating them is a dropdown for selecting a Task, no invalid data can be introduced without post-hacking.

- delete-i.safe: These delete a valid Involves. Same as in create, no invalid data can be legally introduced.

- create-delete-i.hack/create-delete-i-2.hack: These tests do both get and post hacking, trying to access the creation/deletion form of involves whose record the owner is not allowed to modify and introducing invalid tasks in the form of those which the user is allowed to modify.

Coverage for Involves related services is 100%:

Tabla

El contenido generado por IA puede ser incorrecto.

All lines have been fully executed. Again, validations and double checks are not that necessary in this entity as it is much simpler than Maintenance Record.

## **Performance Testing**

A performance analysis will be now carried out by extracting data out of the execution of the above-mentioned tests. For analysis’ sake, the test will be executed in two different computers, and I will compare the results as I would do after a refactoring of the code:

* In other computer. The excel file “tester-performance-clean-other.xlsx” contains the data collected during the execution of the tests in my partner’s computer:

Gráfico

El contenido generado por IA puede ser incorrecto.

As we can see, the most time-consuming operation on average is the maintenance record update operation (120ms), which makes sense since it is likely the most computationally complex.

As for some statistics regarding the executions, we can see the Excel file “tester-performance-statistics.xlsx”:

Tabla

El contenido generado por IA puede ser incorrecto.Tabla, Excel

El contenido generado por IA puede ser incorrecto.

As we can observe, in my partner’s computer, the confidence interval (95%) of the request ranges from 21.42ms to 25.86ms.

* In my computer. The excel file “tester-performance-clean-my-pc.xlsx” contains the data collected during the execution of the tests in my partner’s computer:

Gráfico, Gráfico en cascada

El contenido generado por IA puede ser incorrecto.

In this case, the most time-consuming operation is the creation of maintenance records (45ms). This is reasonable, since it was the second slowest request in the other computer, and it is fairly complex.

Here there are some statistics extracted from the same file as before:

Tabla

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Tabla, Excel

El contenido generado por IA puede ser incorrecto.

In my computer, the confidence interval ranges from 8.48ms to 9.98ms.

Next, using both confidence intervals, a hypothesis test will be conducted using a z-test. This test is documented in the file “tester-perfromance-statistics.xlsx”, with supporting screenshots provided below:

Aplicación, Tabla, Excel

El contenido generado por IA puede ser incorrecto.

The two tail p-value obtained in the z-test is 0. This is due to the huge difference in times and the approximation limit in Excel, so it is not actually 0, but a very close number to it. Nevertheless, as the value is below alpha (0.05), it means that the difference is actually significant. Comparing the means, we can see that, as the time mean on my PC is lower than in my partner’s (9.23ms<23.64ms), the efficiency is higher in my PC.

# **Conclusion**

Most of the code (over 99%) related to Student 5 services has been tested, which has allowed the detection and correction of bugs. Specially, the get and post hacking tests have been especially useful to correct mistakes in the authorise methods, which is, in my opinion, the most important, since it is the primary contention against hacking. Additionally, performance has been statistically analysed, demonstrating that the system has a pretty good response time (peaking at around 120ms in a pretty old computer).

# **Bibliography**

Intentionally blank.