Insertando imagen...

ACME-SF

G1.007

**Testing report**

26/05/2024



# Cover

|  |  |
| --- | --- |
| **Repository:** <https://github.com/DP2-C1-07/Acme-SF-D04> | |
| **Student #1**  **ID:** 31878881F  **UVUS:**  pabcabmar3  **Name:** Caballero María, Pablo **Roles:** manager, developer, tester | **Student #2**  **ID Number:**49034820Q  **UVUS:** mararnmon  **Name:** Arnáiz Montero, Marco Antonio  **Roles:** developer, operator , tester |
| **Student #3**  **ID Number:** 77865211E  **UVUS:**  alfalolan  **Name:** Alonso Lanzarán, Alfonso Luis  **Roles:** developer, tester | **Student #4**  **ID Number:** 53932912M  **UVUS:** albsanmim  **Name:** Sánchez Mimbrero, Alberto  **Roles:** developer , tester |
| **Student #5**  **ID Number:** 48123111G  **UVUS:** juagarcar4  **Name:** Garcia Carballo, Juan  **Roles:** developer , tester |  |

Table of contents

[Cover 2](#_Toc167651754)

[Executive summary 3](#_Toc167651755)

[Revision table 3](#_Toc167651756)

[Introduction 3](#_Toc167651757)

[Contents 3](#_Toc167651758)

[Functional testing 3](#_Toc167651759)

[Performance testing 6](#_Toc167651760)

[Statistical analysis 6](#_Toc167651761)

[Profiling 10](#_Toc167651762)

[Software profiling 10](#_Toc167651763)

[Hardware profiling 10](#_Toc167651764)

[Conclusions 10](#_Toc167651765)

[Bibliography 10](#_Toc167651766)

# Executive summary

This document is a testing report, in where the testing of the requirements of the whole project is explained and analyzed. More specifically, the present report corresponds to the testing tasks carried by the student #1.

# Revision table

|  |  |  |
| --- | --- | --- |
| Number | Date(dd/mm/yyyy) | Description |
| 1.0 | 27/05/2024 | Document done in its entirety, reviewed by peers. No major errors were found. |

# Introduction

The purpose of this document is to provide a detailed testing report of the functional requirements that involves the student #3, in other words, requirements #6 and #7, from the mandatory requirements from the delivery 3 of the project “Acme-SF”.

The structure of the present document is the following: first, a cover that display our group number, our repository, the names of the workgroup members, their corporate e-mail addresses, and date. Then, there is a table of contents that serves as an index of each section, showing the page where each one starts. Then, an executive summary which explains briefly what this report is; a revision table which includes the revision number, date, and short description of the revision in each entry; an introduction which explains more thoroughly the contents of this report; the content itself, which will be split into 2 main sections (functional testing and performance testing); a conclusion, and a bibliography section.

# Contents

## Functional testing

The proceeding through which the test suite was generated was the following: using the tester#recorder tool from Eclipse, the interaction of the user with the application was recorded, in order to perform end to end testing. The features that were tested are listed below, along with a brief description of each one:

**/developer/training-module/create**

The cases that were tested were as it follows. Firstly, a positive test, sending an empty form, and then introducing in each field of the form all the possible data, according to the methodology studied in class , this is: minimum minus the smallest amount, minimum, maximum minus the smallest amount, maximum, maximum plus the smallest amount, and, in the case of string attributes, in addition to that, 2 exotic charsets, a string of SQL injection, and a string of JavaScript injection; in the case of a link, in addition to that, every singular case from a list provided by the professors. In the case of an unique code, a wrongly formed one, an already existing one and a correct one. Obviously, the cases where some mandatory field was null or had an invalid value were rejected.

Then, a hacking tests, create a training module not being a developer.

**/developer/training-module/delete**

The cases that were tested were as it follows. Firstly, positive and negative test, trying to delete when there are invalid values, and then deleting a training module successfully.

Then, hacking tests that include: trying to delete a training module not being developer, trying to delete a training module from another developer, trying to delete an already published training module, and trying to delete an already published training module from another developer.

**/developer/training-module/list-mine**

The cases that were tested were as it follows. Firstly, a positive test, listing successfully all the training modules from a developer.

Then hacking cases that include trying to list training modules not being a developer.

**/developer/training-module/publish**

The cases that were tested were as it follows. Firstly, positive test, negative test : to publish a training module with no training sessions attached, or with some training session that was not published.

Then, hacking cases that include trying to publish a training module not being developer, trying to publish a training module from another developer, and trying to publish a training module not being a developer.

**/developer/training-module/show**

The cases that were tested were as it follows. Firstly, a positive test, showing a training module successfully.

Then, hacking cases that include: trying to show a training module from other developer, and trying to show a training module not being a developer.

**/developer/training-module/update**

The cases that were tested were as it follows. Firstly, positive and negative tests, sending an empty form, and then introducing in each field of the form all the possible data, according to the methodology studied in class , this is: minimum minus the smallest amount, minimum, maximum minus the smallest amount, maximum, maximum plus the smallest amount, and, in the case of string attributes, in addition to that, 2 exotic charsets, a string of SQL injection, and a string of JavaScript injection; in the case of a link, in addition to that, every singular case from a list provided by the professors. In the case of an unique code, a wrongly formed one, an already existing one and a correct one. Obviously, the cases where some mandatory field was null or had an invalid value were rejected.

Then, some hacking tests, including trying to update a training module not being a developer, trying to update a training module from other developer, trying to update an already published training module, and trying to update an already published training module from another developer.

**/developer/training-session/create**

The cases that were tested were as it follows. Firstly, positive and negative test, sending an empty form, and then introducing in each field of the form all the possible data, according to the methodology studied in class , this is: minimum minus the smallest amount, minimum, maximum minus the smallest amount, maximum, maximum plus the smallest amount, and, in the case of string attributes, in addition to that, 2 exotic charsets, a string of SQL injection, and a string of JavaScript injection; in the case of a link, in addition to that, every singular case from a list provided by the professors. Obviously, the cases where some mandatory field was null or had an invalid value were rejected.

Then, some hacking tests, including trying to create a training session not being a developer.

**/developer/training-session/delete**

The cases that were tested were as it follows. Firstly, a positive test deleting a training session successfully.

Then, hacking tests that include: trying to delete a training session not being developer, trying to delete a training session from another developer, trying to delete an already published training session, and trying to delete an already published training session from another developer.

**/developer/training-session/list**

The cases that were tested were as it follows. Firstly, a positive test, listing successfully all the training sessions from a developer.

Then hacking cases that include trying to list training sessions not being a developer.

**/developer/training-session/publish**

The cases that were tested were as it follows. Firstly, positive and negative test, publishing a training session successfully and sending the form with incorrect data. Then, hacking cases that include: trying to publish a training session not being developer, trying to publish a training session from another developer, and trying to publish a training session not being a developer.

**/developer/training-session/show**

The cases that were tested were as it follows. Firstly, a positive test, showing a training session successfully.

Then, hacking cases that include: trying to show a training session from other developer, and trying to show a training session not being a developer.

**/developer/training-session/update**

The cases that were tested were as it follows. Firstly, positive and negative test, sending an empty form, and then introducing in each field of the form all the possible data, according to the methodology studied in class , this is: minimum minus the smallest amount, minimum, maximum minus the smallest amount, maximum, maximum plus the smallest amount, and, in the case of string attributes, in addition to that, 2 exotic charsets, a string of SQL injection, and a string of JavaScript injection; in the case of a link, in addition to that, every singular case from a list provided by the professors. Obviously, the cases where some mandatory field was null or had an invalid value were rejected.

Then, some hacking tests, including trying to update a training session not being a developer, trying to update a training session from other developer, trying to update an already published training session, and trying to update an already published training session from another developer.

## Performance testing

### Statistical analysis

After launching the tester#replayer, and analyzing with Excel tools the time taken in average by each feature among the aforementioned ones, these are the results:

Before adding the indexes to the entities:

After adding the indexes to the entities:

The most significant change has been a 10 miliseconds decrease in the feature /developer/training-module/create, a 5 miliseconds decrease in the feature /developer/training-module/delete.

These are the different statistical metrics computed:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

In average, the second run took around 0,6 millisecond less than the first run. Which means a 5% decrease in time.

Finally, when we compute the two tails z value upon our gathered data, we obtain this summary:

Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente

This means that the z value is between alpha(0,05) and 1 so we can conclude that adding the index didn’t make a significant improvement.

### Profiling

Carrying on with this performance testing, a hardware and software profiling was performed, and this were the results:

#### Software profiling

Using the VisualVM tool, and monitoring the CPU time consumed during the replaying of the tests, filtering by those methods in classes related to the aforementioned explained features, and finally sorting by time in order to find those that take the most, we find the following 10 hot spots, sorted by CPU time: Tabla

Descripción generada automáticamente con confianza media

According to what was studied in class and bearing in mind that the first 2 are not even implemented by me, this screenshot means that the 10 methods, do not take a lot of self-time (0 milliseconds in every one of them), but in total time methods from 3 to 10, implemented by me takes between 0,2 and 1 seconds. Therefore, it is not they that consume time, but instead, the ultimate methods that they invoke from the framework.

The applications that were open when this screenshots was taken were Windows cmd (to run MariaDB), Eclipse (to run the tests) and VisualVM (to measure the CPU time).

#### Hardware profiling

Opening the performance monitor with administrator privileges, and adding the following performance counters: CPU time(black),current disk queue length (which monitors the disk usage, red), “% committed bytes in use (which monitors how much memory is used, showed in yellow) and bytes total/sec (which monitors the bandwidth that your network card is consuming, showed in green), we can observe this metrics during the replaying of the tests:

Gráfico

Descripción generada automáticamente con confianza media

This means that there are some peaks in the disk queue length, and a constant 15% of CPU time in use. The applications that were open when this screenshots was taken were Windows cmd (to run MariaDB), the performance monitor (to measure the aforementioned 3 performance indicators), and Eclipse (to run the tests).

Nevertheless, and according to the methodology studied in class, it is not generally a good idea to change suddenly the hardware or the software environment with which a project is being developed, as the client perception will worsen significantly.

# Conclusions

According to what was studied in class, the metrics observed imply that the changes made were not relevant enough to lower the mean times that the different features took to execute. But, since the self time of the 2 methods that consume more time were the same in the machine where they were ran as in another student’s machine, after the indexes for Banner were created, we conclude that they are just not very powerful generally. On top of that, some methods in the hot spots that take more than 0 milliseconds use queries with attributes for which there are not even an index. In other words, this means that there exists several time consuming methods that cannot be accelerated by using the indexes.

# Bibliography

Intentionally blank.