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 | 26/05/2024 | 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 #1, 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:

**/manager/project/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, some negative tests, including trying to create a project not being a manager.

**/manager/project/delete**

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

Then, negative tests that include: trying to delete a project not being manager, trying to delete a project from another manager, trying to delete an already published project, and trying to delete an already published project from another manager.

**/manager/project/list-mine**

The cases that were tested were as it follows. Firstly, a positive test, listing successfully all the projects from a manager.

Then negative cases that include trying to list projects not being a manager.

**/manager/project/publish**

The cases that were tested were as it follows. Firstly, a positive test, trying to publish a project with no user stories attached, or with some user story that was not published (these 2 cases were not allowed by the application),and then publishing a project successfully.

Then, negative cases that include: trying to publish a project not being manager, trying to publish a project from another manager, and trying to publish a project not being a manager.

**/manager/project/show**

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

. Then, negative cases that include: trying to show a project from other manager, and trying to show a project not being a manager.

**/manager/project/update**

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, some negative tests, including trying to update a project not being a manager, trying to update a project from other manager, trying to update an already published project, and trying to update an already published project from another manager.

**/manager/project-user-story/create**

The cases that were tested were as it follows. Firstly, a positive test, sending a ProjectUserStory entity with both fields (project and user story) empty, and then each one individually, which were rejected by the app; and then creating one successfully.

Then, negative tests cases including trying to create one not being a manager.

**/manager/project-user-story/delete**

The cases that were tested were as it follows. Firstly, a positive test, deleting a ProjectUserStory entity successfully. Then, negative tests cases including trying to delete one not being a manager, and trying to delete one that belongs to another manager.

**/manager/project-user-story/list**

The cases that were tested were as it follows. Firstly, a positive test, listing the ProjectUserStory entities successfully.

Then, negative cases including trying to access the feature not being a manager.

**/manager/project-user-story/show**

The cases that were tested were as it follows. Firstly, a positive test, showing the ProjectUserStory entity successfully.

Then, negative cases including trying to access the feature not being a manager, or trying to show a ProjectUserStory from another manager.

**/manager/user-story/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. Obviously, the cases where some mandatory field was null or had an invalid value were rejected.

Then, some negative tests, including trying to create a user story not being a manager.

**/manager/ user-story /delete**

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

Then, negative tests that include: trying to delete a user story not being manager, trying to delete a user story from another manager, trying to delete an already published user story, and trying to delete an already published user story from another manager.

**/manager/ user-story /list**

The cases that were tested were as it follows. Firstly, a positive test, listing successfully all the user stories from a manager.

Then negative cases that include trying to list user stories not being a manager.

**/manager/ user-story /publish**

The cases that were tested were as it follows. Firstly, publishing a user story successfully. Then, negative cases that include: trying to publish a user story not being manager, trying to publish a user story from another manager, and trying to publish a user story not being a manager.

**/manager/ user-story /show**

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

Then, negative cases that include: trying to show a user story from other manager, and trying to show a user story not being a manager.

**/manager/ user-story /update**

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. Obviously, the cases where some mandatory field was null or had an invalid value were rejected.

Then, some negative tests, including trying to update a user story not being a manager, trying to update a user story from other manager, trying to update an already published user story, and trying to update an already published user story from another manager.

**/manager/ user-story /list-mine**

The cases that were tested were as it follows. Firstly, a positive test, listing successfully all the user stories from a manager’s project.

Then negative cases that include trying to list user stories from a project, not being a manager; and trying to list the user stories from another manager’s project.

## 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 20 miliseconds decrease in the feature /manager/project/create, and a 10 miliseconds increase in the feature /manager/project/delete. Nevertheless, upon a further insight onto the statistical analysis that was performed, we find that this changes are of no relevance, since they were run on a 5 year-old, low-end laptop. What is more, even the features provided by the base project (/, /system/sign-in and /system/welcome took more time during the second run).

These are the different statistical metrics computed:

Before adding the indexes:

Tabla

Descripción generada automáticamente

After adding the indexes:

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamente

The intervals computed were as they follow:

Before adding the indexes:



After adding the indexes:



In average, the second run took around 1 millisecond more than the second run. But taking into account that during this run the variance was higher, this 5% increase is deemed irrelevant.

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

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

This means that the z value is slightly above the ideal range, whose maximum would be the alpha value (0,05). It is worth mentioning that the replaying of the tests took more than an hour on my laptop, so it would not be surprising finding than background processes running on it could alter slightly this result. Anyways, no other application was opened, and therefore, theoretically, no other foreground processes whatsoever were actively running simultaneously.

### 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:

Interfaz de usuario gráfica, Aplicación

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 6 methods from 3 to 9, do not take a lot of self time (0 milliseconds in every one of them), but in total time (between 0,1 and 0,2 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: current disk queue length (which monitors the disk usage, showed in black), “% committed bytes in use (which monitors how much memory is used, showed in blue) 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:

Diagrama

Descripción generada automáticamente

This means that there are some peaks in the disk queue length, and a constant 90% of committed bytes 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.