Insertando imagen...

ACME-SF

G1.007

**Testing report**

27/05/2024



# 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](#_Toc167712519)

[Executive summary 3](#_Toc167712520)

[Revision table 3](#_Toc167712521)

[Introduction 3](#_Toc167712522)

[Contents 3](#_Toc167712523)

[Functional testing 3](#_Toc167712524)

[Performance testing 4](#_Toc167712525)

[Statistical analysis 4](#_Toc167712526)

[Profiling 8](#_Toc167712527)

[Software profiling 8](#_Toc167712528)

[Conclusions 9](#_Toc167712529)

[Bibliography 9](#_Toc167712530)

# Executive summary

This document is a testing report, in where the testing of the requirements assigned to the student #04 is explained and analyzed. More specifically, the present report corresponds to the testing tasks of the entities Sponsorship and Invoice.

# 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 #04, 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 Acme Framework tester#recorder tool from Eclipse, the interaction of the user with the application was recorded, to perform end to end testing. The features that were tested are listed below, along with a brief description of each one:

**/sponsor/sponsorship/create**

The cases that were tested were as it follows. Firstly, 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, that is: minimum minus the smallest amount and maximum plus the smallest amount. In the case of the code property, testing every regex group by separate (minimum – 1 and maximum + 1 characters, type of characters…), and a code of a sponsorship that already exists. In the case of a link, in addition to the length tests of a plain string, a simple string that is obviously not an URI (“asdasdasd”) and a URI without its scheme (for example, without http://, that is, “google.com”). In the case of emails, in addition to the beforementioned length tests, invalid email patterns. In the case of dates, the minimum minus the smallest amount and maximum plus the smallest amount, considering that some dates minimums were another date from the form. In the case of money properties, minimum – 0.01 and maximum + 0.01 and an unsupported currency.

Then, a positive test, with all correct values and a hack test, trying to create a sponsorship while not being a sponsor.

**/sponsor/sponsorship/delete**

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

Then, hack tests that include trying to delete a sponsorship while not being a sponsor, trying to delete another sponsor’s sponsorship and trying to delete a published sponsorship with the right user.

**/sponsor/sponsorship/list-mine**

The cases that were tested were as it follows. Firstly, a positive test, listing successfully all the sponsorships of the sponsor.

Then a hack test trying to list sponsorships while not being a sponsor.

**/sponsor/sponsorship/show**

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

Then, two hack tests, trying to show a sponsorship while not being a sponsor and trying to show another sponsor’s sponsorship.

**/sponsor/sponsorship/update**

The cases that were tested were as it follows. Firstly, negative tests, following the /sponsor/sponsorship/create approach. Also, in the case of money properties, adding a test case trying a different currency than the currency of all the registered sponsorship invoices.

Then, a positive test, with all correct values.

Then, hack tests that include trying to update a sponsorship while not being a sponsor, trying to update another sponsor’s sponsorship and trying to update a published sponsorship with the right user.

**/sponsor/sponsorship/publish**

The cases that were tested were as it follows. Firstly, negative tests, following the /sponsor/sponsorship/update approach, but adding trying to publish a sponsorship with some invoices unpublished.

Then, a positive test, with all correct values and invoices published.

Then, hack tests that include trying to publish a sponsorship while not being a sponsor, trying to publish another sponsor’s sponsorship and trying to publish an already published sponsorship with the right user.

**/sponsor/invoice/create**

The cases that were tested were as it follows. Firstly, 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, that is: minimum minus the smallest amount and maximum plus the smallest amount. In the case of the code property, testing every regex group by separate (minimum – 1 and maximum + 1 characters, type of characters…), and a code of an invoice that already exists. In the case of a link, in addition to the length tests of a plain string, a simple string that is obviously not an URI (“asdasdasd”) and a URI without its scheme (for example, without http://, that is, “google.com”). In the case of dates, the minimum minus the smallest amount and maximum plus the smallest amount, considering that some dates minimums were another date from the form. In the case of money properties, minimum – 0.01 and maximum + 0.01, an unsupported currency and a different currency than the sponsorship that this invoice is part of. In the case of numbers, minimum minus the smallest amount and maximum plus the smallest amount.

Then, a positive test, with all correct values and three hack tests, trying to create an invoice while not being a sponsor, trying to create an invoice for another sponsor’s sponsorship and trying to create an invoice for a published sponsorship.

**/sponsor/invoice/delete**

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

Then, hack tests that include trying to delete an invoice while not being a sponsor, trying to delete another sponsor’s invoice and trying to delete a published invoice or an invoice of a published sponsorship with the right user.

**/sponsor/invoice/list**

The cases that were tested were as it follows. Firstly, a positive test, listing successfully all the invoices of the sponsor and listing all the invoices of a sponsorship of the sponsor.

Then two hack tests trying to list invoices while not being a sponsor and trying to list the invoices of another sponsor’s sponsorships.

**/sponsor/invoice/show**

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

Then, two hack tests, trying to show an invoice while not being a sponsor and trying to show another sponsor’s invoice.

**/sponsor/invoice/update**

The cases that were tested were as it follows. Firstly, negative tests, following the /sponsor/invoice/create approach.

Then, a positive test, with all correct values.

Then, hack tests that include trying to update an invoice while not being a sponsor, trying to update another sponsor’s invoice, trying to update a published invoice and trying to update an invoice of a published sponsorship with the right user.

**/sponsor/invoice/publish**

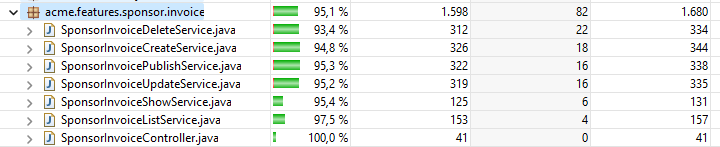
The cases that were tested were as it follows. Firstly, negative tests, following the /sponsor/invoice/update approach.

Then, a positive test, with all correct values and invoices published.

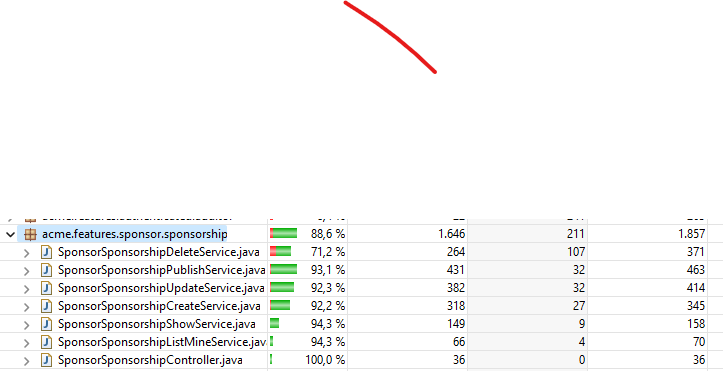
Then, hack tests that include trying to update an invoice while not being a sponsor, trying to update another sponsor’s invoice, trying to update a published invoice and trying to update an invoice of a published sponsorship with the right user.

## Test coverage.

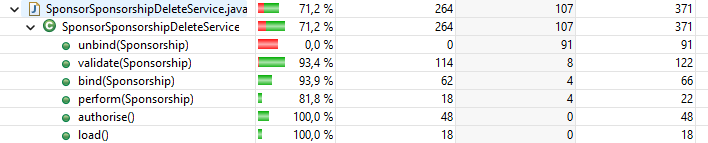
The test suite ensures a coverage of more than 93%. For invoice features:



For sponsorship features:



There is an exception: the delete service. This is due to the unbind method not being tested, because the entity is going to be deleted and not updated.



## 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:

There’s a slight decrement in the execution time, but not a huge difference. This is because most of the queries search by id or code, and they already had an index. The code property had its index because it was specified in the implementation to assure its uniqueness.

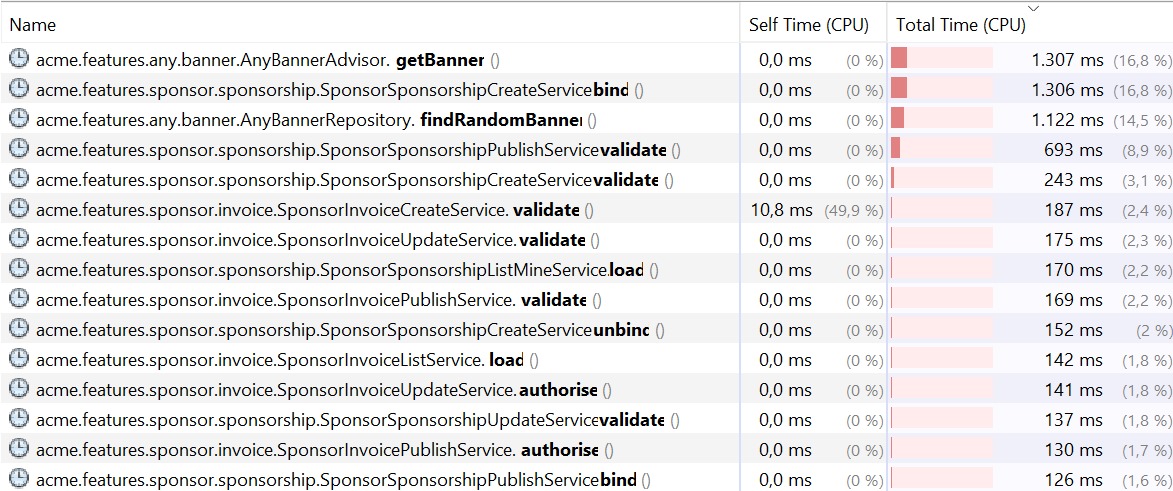
The only indexes added were to foreign keys.

### Profiling

Carrying on with this performance testing, a 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:



According to what was studied in class, every feature has more total time than self time (0 ms in every method). Therefore, it is not they that consume time, but instead, the ultimate methods that they invoke from the framework. The methods that consume more time are #validate() in sponsorship services. That’s normal, because they need to iterate through the sponsorship invoices to check if all match the same currency and, in the case of the publish service, if all sum the sponsorship amount and are published.

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

# Conclusions

The changes made in the code were successful in terms of (somehow) lowering the mean times that the features take to execute. Nevertheless, the same trace in different machines could consume a different amount of time. Also, the features that take the most time can not be improved, because they are either provided by the framework, or provided by Spring.

# Bibliography

Intentionally blank.