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](#_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 of the whole project is explained and analyzed. More specifically, the present report corresponds to the testing tasks of the entity Banner and requirement #11.

# 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 group, in other words, requirement #11, 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:

**/administrator/banner/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 plus and 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 dates, in addition to the minimum minus the smallest amount, minimum, maximum plus and minus the smallest amount, maximum and maximum plus the smallest amount, and taking into account that the minimum gap is 7 days, it was tried the minimum gap, the minimum plus and minus the smallest amount, the maximum, the maximum plus and minus the smallest gap. Also, trying when the end date is before the beginning date. 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 banner not being an administrator.

**/administrator/banner/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 banner successfully.

Then, negative tests that include: trying to delete a banner not being administrator.

**/administrator/banner/list**

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

Then negative cases that include trying to list banners not being an administrator.

**/administrator/banner/show**

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

Then, negative cases that include trying to show a banner not being a administrator.

**/administrator/banner/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 plus and 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 dates, in addition to the minimum minus the smallest amount, minimum, maximum plus and minus the smallest amount, maximum and maximum plus the smallest amount, and taking into account that the minimum gap is 7 days, it was tried the minimum gap, the minimum plus and minus the smallest amount, the maximum, the maximum plus and minus the smallest gap. Also, trying when the end date is before the beginning date. Obviously, the cases where some mandatory field was null or had an invalid value were rejected.

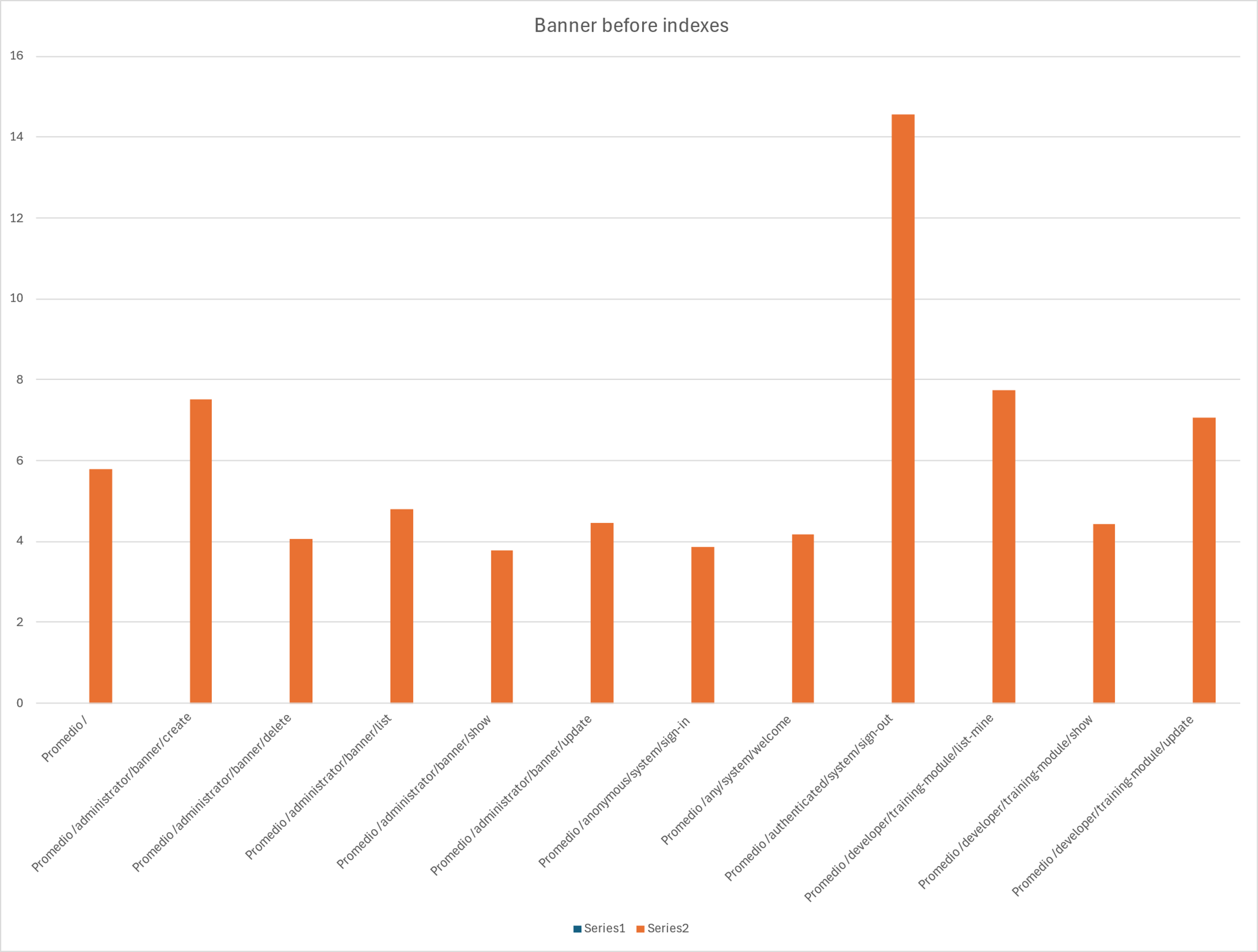
Then, some negative tests, including trying to create a banner not being an administrator.

## Performance testing

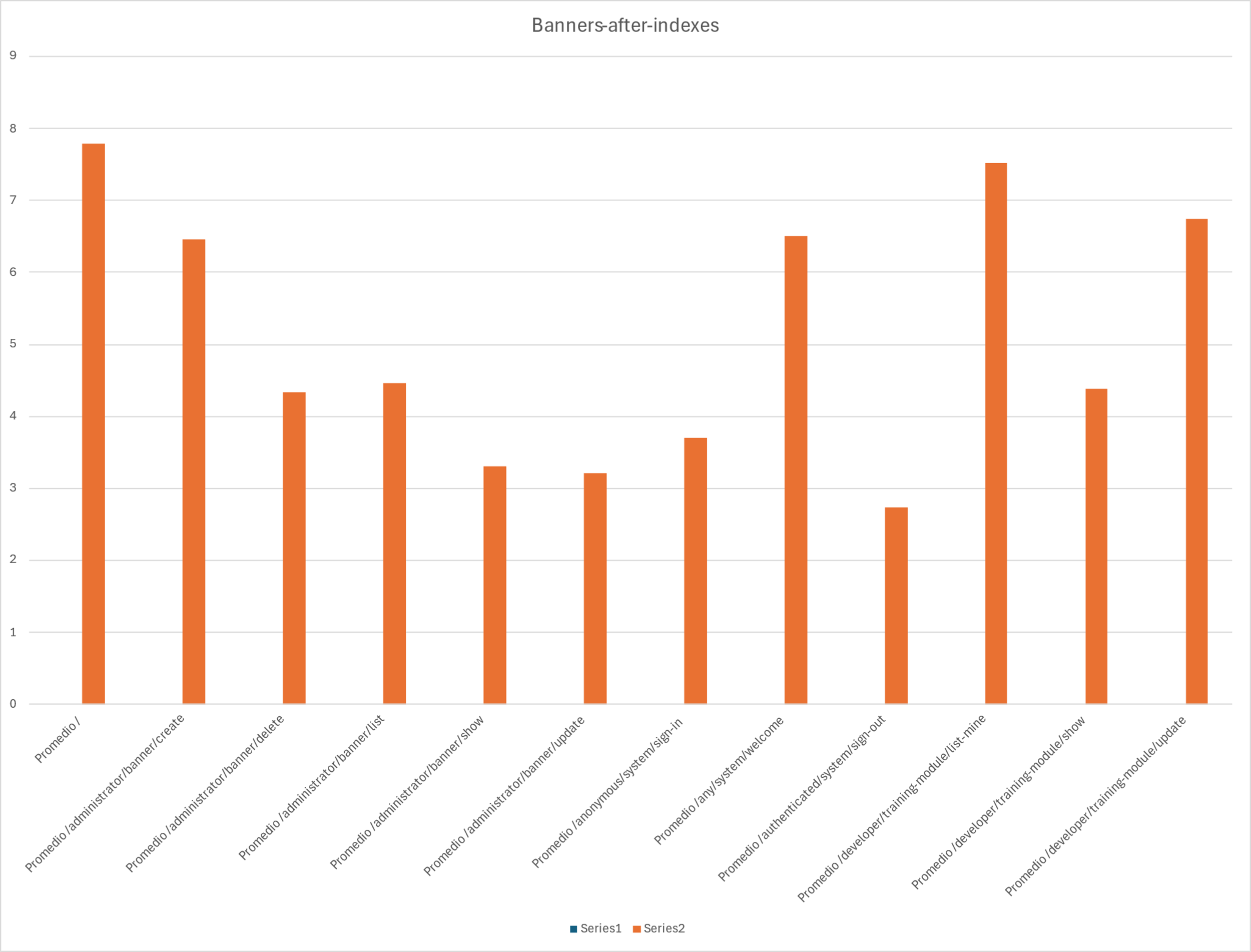
### Statistical analysis

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 decrease in the feature to sign out from 15 ms to 3 ms, and an increase in the feature “/” from 6 ms to 8 ms. Nevertheless, and focusing on those requirements developed by us, there is a 1 ms decrease in the features to create and to update the banners. The rest of them are roughly the same.

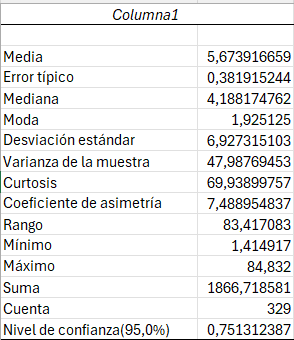
These are the different statistical metrics computed:

Before adding the indexes:

After adding the indexes:

The intervals computed were as they follow:

Before adding the indexes:



After adding the indexes:

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamente

In average, the second run took around 1 millisecond less than the second run. The variance is also lower during the second run, when the indexes were utilized, meaning that a higher percentage of times, the times were lower, because the minimum and the maximum are roughly the same and we mentioned before that the features that take the most time are those not related directly to banners themselves.

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

The two-tail z value is 0,101, which is almost 0 and therefore lower than the alpha value, meaning than the changes in our code were significant and thus, we can compare the mean times.

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 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, 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 the ones to get a banner and retrieve it from the database (around 1 second each).

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 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.