Previous WIS testing knowledge

Acme AirNav Solutions, Inc

Group Number: C1.066
Repository: `https://github.com/mquirosq/DP2-C1.066`

Members:
María Quirós Quiroga, marquiqui@alum.us.es
Guillermo Rodríguez Narbona, guirodnar@alum.us.es
Ignacio Mora Pérez, ignmorper1@alum.us.es
Daniel Herrera Urbano, danherurb@alum.us.es
Alejandro Parody Quirós, aleparqui@alum.us.es

February 20, 2025

Contents

Executive Summary

The main objective of this report is to present the knowledge about Web Information System testing that we have previously acquired before this subject, thus providing a clear insight on the concepts and content we are already familiar with.

Revision History

Revision	Date	Description
1.0	2025-02-18	Initial draft
1.1	2025-02-20	Changes after first correction
1.2	2025-02-20	Changes after second correction

1. Introduction

The act of testing code as part of the development of a Web Information System is not something new, as we have seen it previously during this and the past course. As all members of the team are students from the Software Engineering grade at the University of Seville, we all share the same knowledge about WIS testing from the previous subjects. This is the reason why, in the following sections, we are going to talk as a group.

2. Architecture and Integration of Information Systems

This subject was the first to introduce testing as a way to check that our code behaved as expected. This content was covered during a couple of laboratory classes and as part of a practical project (the second of the two required to pass the subject).

During laboratory classes, we only learned how to create unit tests with JUnit. These were written and run to verify the correct operation of individual units of code, such as functions, methods, or classes that we had worked on in previous lessons.

The second practical project consisted of developing a RESTful API that provided information about data collected from Youtube and Vimeo. For this, we followed a microservices architecture and used the Spring-Boot framework to build the three required systems. The concrete functions that made the collection and retrieval of the videos possible were the SUT in this project. We used JUnit to generate these tests, applying the knowledge acquired from the laboratory classes.

3. Design and Testing I

This subject was part of the first half of our third academic year, which means that the knowledge is still very recent in our minds. In this subject, we have learned to perform unit testing, mock testing, end-to-end tests and a lot more.

First, for the unit tests, we created enough to cover all of the service classes, domain model classes, validators, and custom queries we had implemented. For each test, we included two positive cases, in which we checked that they followed a normal behavior, and one negative case, where we tested abnormal behaviors. However, for SUTs that contained loops, additional test cases were required to check that they performed correctly. All this was done using the JUnit5 testing framework and the @DataJpaTest annotations.

We also studied the application of sociable tests versus solitary tests, as well as test doubles, using the Mockito framework for the latter.

For web layer testing, we studied and applied the Spring @WebMvcTest annotation to verify outputs, such as HTTP status or JSON responses, simulating the corresponding

HTTP requests needed for the tests.

Next, for end-to-end and acceptance tests, we applied a Mock MVC (Model View Controller) to simulate HTTP calls across the whole application. This allowed us to test several elements of the project all the way, such as the registration form.

Lastly, for several of the React components we implemented, we created some tests using Jest, with the help of a mock API which returned test data. In this way, we check that they were in the correct states and that they rendered correctly.

4. Conclusions

We, as software engineering students, know the importance of checking that the code we write works as it should, even though it was not after these two subjects that we learned several ways to do it. Until then, all the code we wrote was tested manually, expecting output from predefined data and omitting edge cases that could break the system.

We had used tests in other subjects, but we did not learn how to implement them in our projects; we simply debugged them to make sure our applications did not have any incorrect behaviors. These tests were also an evaluation method, so we did not understand the real purpose behind them.

Throughout these subjects, mostly in the last one, we have acquired the knowledge and tools to understand how to perform tests, whether it is unit testing, end-to-end, or front-end tests.

With both the theoretical and practical foundations obtained from these two subjects, we hope that they will allow us to approach the related activities of the project proposed on this subject.

References

Intentionally blank.