Qualidade de Software - Badge 4

*

M13939 João Martins
Informatics Department
UBI
Covilhã, Portugal

M13239 Vasco Senra
Informatics Department
UBI
Covilhã, Portugal

M13943 Rui Simões
Informatics Department
UBI
Covilhã, Portugal

***Abstract*—This document as written in the context of the practical exercise "Badge 4 - Talk a bit" of class Software Quality, lectured by the professor Nuno Pombo.**

I. Introduction

This practical project pretends too illustrate a practical example of the implementation of automated testing of a web application. This specific web application, made by a 3rd party, consists of a animal hotel management application, and such, it performs the basic CRUD(create, read, update and delete) functionalities that manage the rooms, the pets and users in question. This badge project will explain how the implement automated tests using selenium [1] where implemented and made testing all the functionalities easier and faster. Lastly, the results of the tests made will be numbered and discussed, and apart from the tests themselves, we will discuss the effectiveness of the automation tool in question used.

II. Automated Testing

The purpose of automated testing is to s to ensure that a software system works as intended with minimal human intervention, this might be be functional testing, like the tests present on this report, or it may also be non functional testing (like performance or compatibility testing) and either case the execution consist on the same processes.

Generally, all automated testing tools use the same methods, they run scrips created to simulate the activities that the app would have to run during production, that can be achieved though test frameworks,test scripts other kinds of software.

A. Automated Testing vs Unit Testing

Automated tests encompass any testing activities that are executed with the assistance of automation tools or scripts, unit testing, on the other hand, specifically focuses on testing individual units or components of a software application in isolation, and as such, there are differences like:

- Automated tests can encompass a broad range of testing activities, like unit testing, integration testing and system testing, so the scope of automated tests is broader then unit tests, since unit testing focuses on testing individual units or components in isolation.
- Automated tests, like previously mentioned, have a global reach, therefor, this have a more definitive value, because even if individual unit tests came out positive, an application wide test can show that one functionality of the application breaks another one.
- Automated tests can be performed at various stages of the development and testing process, but unit testing is typically done by developers during the coding phase, before the code is integrated into the larger codebase.

Identify applicable funding agency here. If none, delete this.

III. Selenium - Automated Testing Framework

A. Overview of Selenium

1) Introduction to Selenium: Selenium [1] stands out as a robust **open-source** automated testing framework widely employed for ensuring the quality of web applications. Its pivotal role lies in automating browser actions, offering testers the capability to simulate user interactions with web elements.

2) Key Features: Selenium boasts a range of features that contribute to its popularity in the testing community. With **cross-browser compatibility**, **platform independence**, and support for multiple programming languages, Selenium provides a versatile and comprehensive solution for web application testing.

3) Components of Selenium: Comprising **Selenium WebDriver**, Selenium IDE, and Selenium Grid, the Selenium suite offers a modular approach to automated testing. WebDriver, in particular, enables dynamic interaction with browsers, allowing testers to script actions that simulate user behavior effectively.

B. Selenium WebDriver

1) Introduction to WebDriver: Selenium WebDriver [2] serves as the powerhouse behind automated browser testing. By directly interacting with web browsers, it replicates user actions, facilitating the creation of intricate test scenarios.

Listing 1. Example of WebDriver Selenium initializacion with Python

```
os.environ['PATH'] += r"C:\SeleniumDrivers"
driver = webdriver.Chrome()

driver.get('http://localhost:8080/')
```

2) Supported Browsers: WebDriver extends its support to a wide array of browsers, enabling testers to ensure consistent functionality across various platforms. This capability is crucial for maintaining a high standard of **cross-browser compatibility** in web applications.

3) WebDriver APIs: Selenium WebDriver provides APIs for different programming languages, including **Java**, **Python**, and **C#**. These APIs empower testers to write expressive and efficient test scripts, catering to the preferences and expertise of the development team.

C. Writing Selenium Test Scripts

1) Setting Up Test Environment: Configuring the testing environment with Selenium involves tasks such as installing WebDriver, configuring browsers, and managing dependencies. This preparatory phase is critical for ensuring the seamless execution of test scripts.

2) Creating Test Cases: In the process of writing Selenium test scripts, testers focus on fundamental actions like navigating to URLs, interacting with web elements, and verifying expected outcomes. These scripts serve as blueprints for automated testing scenarios.

Listing 2. Example of navigation to URLs and interacting with web elements with Python

```
# Login
driver.get('http://localhost:8080/')

email_input = driver.find_element(By.NAME, '
    email')
password_input = driver.find_element(By.NAME,
    'password')
login_button = driver.find_element(By.ID, '
    btnSub')
```

3) Best Practices: To maximize the effectiveness of Selenium test scripts, adherence to best practices is essential. Strategies for handling dynamic elements, synchronization, and error handling contribute to the creation of robust and maintainable test suites.

D. Selenium and Spring Boot Integration

1) Testing Spring Boot Applications: Integrating Selenium with **Spring Boot for testing the "Pets Hotel" application** involves understanding the synergies between the two technologies. This **badge** explores how Selenium seamlessly integrates into the Spring Boot environment.

E. Conclusion for Selenium Section

In conclusion, Selenium stands as a pillar in the realm of automated testing, offering a versatile and powerful framework for ensuring the quality and reliability of web applications. Its integration into the testing of the "Pets Hotel" Spring Boot application demonstrates its adaptability and effectiveness in real-world scenarios. As we delve deeper into the results and discussions, the role of Selenium becomes even more apparent in shaping the success of automated testing processes.

IV. Spring Boot Application "*Pets Hotel*"

The *Pets Hotel* **Spring Boot Application** is a web-based software solution designed to facilitate the efficient management of an Animal Hotel. Its primary purpose is to streamline the operations and services provided by an animal hotel, ensuring an efficient experience for both hotel administrators and customers.

A. Application Features

The main **features** of this application include:

- Profile Selection: The system was two distinct user profiles, *Hotel Administrator* and *Customer*, that offers differentiated access and control over the application's features.
- Customer Management: Hotel staff can register, view, update, and delete customer information, simplifying customer relationship management.
- Animal Management: The application enables the registration of animals staying at the hotel, including details such as name, specie, age, and race. Each animal is associated with a specific customer, making tracking and management more efficient.
- Room Management: Hotel administrators can oversee the available rooms within the establishment. Information such as room number, price, and room type is easily accessible and manageable through the application.
- Reservations and Accommodation: *Customer* can create and manage (delete) its own reservations. This feature includes associating customers, animals, and rooms, recording check-in and check-out dates, and selecting room types, offering comprehensive control and organization of reservations.
- Statistics and Reports: In addition to its core functionalities, the application provides tools for generating statistics and reports on the hotel's operations. This includes the number of stays per species from all time or in a date interval.

B. Techonologies Used

The **technologies** used to develop this application include

- Jakarta EE (formerly known as Java EE) [3]: is used for implementing enterprise-level functionality, including the development of RESTful web services. It ensures that the application adheres to Java EE standards for robust and secure operation.
- Spring Boot [4]: the application is built using the Spring Boot framework, which simplifies the development of Java-based applications. It provides features for building web applications, handling data access, and managing the application's lifecycle.
- MySQL [5]: is employed as the relational database management system (RDBMS) for data storage. It stores customer details, animal information, reservation records, room data, and other essential application data.

C. Data Persistence

The application utilizes a well-defined database schema and data models to organize and store information efficiently. Here is an overview of **Relational Schema**:

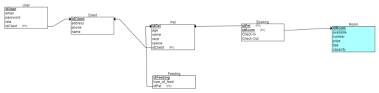

Fig. 1. Relational Schema of Pets Hotel Database

D. Interactive Showcase: App Demo & Code Repository

In this section, we provide an in-depth exploration of the Pets Hotel Web Application. Gain insights into the functionality and structure of our project by utilizing the following resources.

Demo Video:
Watch App Demo
Experience a comprehensive overview of the Pets Hotel Web App through our detailed demo video. Witness the application in action, highlighting key features and functionalities.

Code Repository:
GitHub Repository
Delve into the underlying architecture and source code of the Pets Hotel Web App by visiting our GitHub repository. This resource provides a transparent view of the project's codebase, offering valuable insights into the development process.

V. Test scenarios and their implementation with Selenium

As stated above, the tests were done with Selenium and Python. These tests aimed to verify the operation of the application. We divided the **Test Scenarios**, that were delivered by the application features (discriminated in IV-A), in categories and each one have the respective **Test Scenarios Set**.

- Customer Tests (CT)
- Pets Tests (PT)
- Rooms Tests (RT)
- Booking Tests (BT)

We will cover these tests more thoroughly in the sections below.

A. Customer Tests - CT

Starting with user tests, these were also divided into:

- CT-1: Register User (Customer) Test
- CT-2: Adding Personal Data Test
- CT-3: Update Personal Data Test
- CT-4: Delete User Test

CT-1: The registration test consists of the user being on the login page, clicking on the registration button and writing their data such as email and password. After entering the data, the user clicks on the registration button and if he was redirected to home page is the confirmation that the test was successful, if not, it failed

CT-2: In the test of adding personal information, the user logs into the system and the main page of the system when logged in, has an area with the user's personal data such as name and address. The test consists in logging and filling these fields with this information and clicking in the add button, in case of success the page doesn't show the warning "No personal information added".

CT-3: In the test of updating personal information consists in logging and change these fields with the new information and clicking in the update button, in case of success the field updated shows the new update data.

CT-4: Finally, the delete test consist on enter on admin page looking for the customer button option, looking for the new user and after that, remove that user from database and all of his data.

B. Pets Tests - PT

The pet tests, these were also divided into:

- PT-1: Register Pet Test
- PT-2: Update Pet Test
- PT-3: Delete Pet Test

PT-1: The Register Pet test start with the previous user login in main page and search for the adding pet button on page. When click on the button, he can add her pet to the system. The pet has fields such as name, specie, age and breed and all of her need to be inserted manually by the user. When done, the user received a warning with the status of success or fail.

PT-2: Update Information pet is the same, the user login in the site search for the pet in end of page where he have a list of all your pets and click on button Update, where he can change the data. After that he just need to click on Update button and will receive the warning with the status of success or fail.

PT-3: The delete test is similar to update the only difference is in delete test the button the user need to click is the delete so it's particularly the same steps. In the end, the user receive the warning of success or fail.

C. Rooms Tests - RT

The room tests, these were also divided into:

- RT-1: Add New Room Test
- RT-2: Change type of Room Test
- RT-3: Remove Room Test

In tests with rooms, the system administrator need to make login in system. So the tests start with the admin logging in the system from the main page like default user.

RT-1: The Add New Room test we use the system administrator account to add a room, no linkage to any other property is necessary, and we input the mandatory attributes of a room number, type and price. When done, the user received a warning with the status of success or fail.

RT-2: Change type of Room test requires a the system administrator account, in this test, the first item of the list of rooms is selected, then the same inputs for created a room entry are needed to feed the test script, room number, type

and price. When done, the user received a warning with the status of success or fail.

RT-3: The remove room test requires a the system administrator account, in this test, the first item of the list of rooms is selected, then the remove button is pressed. When done, the user received a warning with the status of success or fail.

D. Booking Tests - BT

The booking tests, these were also divided into:

- BT-1: User Bookings Test
- BT-2: Admins Bookings Test

Despite being divided, the steps are similar so I will treat them as if they were the same. So, we can dived the tests in:

- Add Booking
- Remove Booking
- **Negative Test Case**: Booking a Check-In Date that follows the Check-Out date

BT-1: To test the add booking we need logging in the main page, as user or admin, after that we search for button to go to booking page, in the booking page we have three options, we choose the add option where he have the date from enter and leave, the name of pet and the type of room. With that selected, we click on the add button and the system will return one warning with status failed or successfully. Test succeeded if the new lodging is already discriminated on booking page, if not, fails.

BT-2: To remove the booking we just need to login go to the booking page search from the pet's name and in same row where we can find the name we have one button to delete the booking. Test succeeded if lodging in question disappear from booking page if continues there, fails.

BT-3: The negative test consists in try booking in an impossible date, like the enter date after the out date. In that case, the tests return fails, which mean the system don't let this happen, which is a good sign.

VI. Results

Now with everything explained above, how the tests are actually performed and the results seen, we showcase the results in the following table I.

The unit testing phase encompassed a total of 13 individual tests, where the system's performance was rigorously evaluated. Out of these tests, 12 concluded successfully, demonstrating the system's robustness in meeting the defined criteria. However, one test resulted in a failure, signifying a singular area of concern that warrants further attention and resolution.

To further gauge the system's reliability and performance, we calculate the Probability of Failure on Demand (POFOD). This metric specifically assesses the likelihood of system failure when its operation is required.

Calculating POFOD:

Given:

- Number of Failed Tests: 1
- Total Number of Tests: 13

TABLE I
Tests and Results

Tests:	**Expected Result:**	**Obtained Result:**	**Status:**
Register Client	Passed	Passed	Success
Insert Personal Data	Failed	Failed	Failed
Update Personal Data	Passed	Passed	Success
Remove Client	Passed	Passed	Success
Add Pet	Passed	Passed	Success
Update Pet	Passed	Passed	Success
Remove Pet	Passed	Passed	Success
Add Booking	Passed	Passed	Success
Remove Booking	Passed	Passed	Success
Add Booking Negative case	Failed	Failed	Success
Add Room	Passed	Passed	Success
Update Room	Passed	Passed	Success
Remove Room	Passed	Passed	Success

Formula for POFOD:

$$POFOD = \frac{\text{Number of Failures}}{\text{Total Number of Demands}}$$

In this context:

$$POFOD = \frac{1}{13} = 0.0769$$

This implies a POFOD of approximately 7.69%, signifying the likelihood of system failure when the system is demanded for operation.

The POFOD calculation serves as a crucial measure in understanding the system's potential vulnerability during its operational demands, highlighting the importance of addressing the identified failure to enhance overall system reliability.

VII. Discussions and Conclusions

This project proved the importance of automated test, and how its implementation since the very first stages of development can increase the reliability and quality of an application.

We came to understand that due to increasing complexity it becomes testing software without the help of testing tool becomes much less reliable, and such things as integration testing is near impossible, therefor, the agility, scalability, and scope of this method of testing is much superior.

In conclusion, tools like selenium help provide tools to increase productivity and the scope of testing, and that automated testing, when applied well and since the beginning of the development process, can facilitate and increase code quality and reliability.

Acknowledgment

We want to acknowledge the contributions of our professor, Nuno Pombo P.H.D. for this class, for supporting us, clarifying our questions, and making our path clear when comes to this project, as well for providing us with high quality material and theoretical papers to help with our research.

REFERENCES

[1] Selenium [Online]. Available: https://www.selenium.dev/
[2] Selenium WebDriver. Available: https://www.selenium.dev/documentation/webdriver/
[3] Java EE [Online] Available: https://www.oracle.com/pt/java/technologies/java-ee-glance.html
[4] Spring Boot [Online] Available: https://spring.io/projects/spring-boot
[5] MySQL [Online] Available: https://www.mysql.com/