TEST PLAN

BANCO DE SÉRIES

Designed By:
Diego Henrique
08/10/2020

TABLE OF CONTENTS

1.0 INTRODUCTION

This document presents the test activities planning for the application **Banco de Séries** which has the goal to be the guide for the test activities, revision, verification and project validation. Regarding these, correctives and preventives actions can be taken once the results get highly divergent from the planned.

2.0 SCOPE

This plan is applied for integration tests, system tests, and user acceptance tests to be conducted during each release version.

Different test strategies, such as performance, stress, load and security tests will not be covered in this plan.

3.0 TESTING STRATEGY

The testing strategies were developed to guarantee the completion of the activities described in the scope (section 2).

Aiming to complete all activities in a short period of time the testing team will work in 2 Sprint of 5 days each. The testing guarantees the needed coverage making possible a new release of the application in the first Sprint. And in the last Sprint, the team will be focused on retesting bug fixes, regression tests, and user acceptance tests.

In the following subsections the plan contains the testing levels and types with their objectives and acceptance criteria.

3.1 Alpha Testing (Unit Testing)

Unit tests are designed for verifying the behavior of a single function, method, or class through a mock of objects being created to test sections of code that are not yet part of a complete application. For the application these tests seek static coverage in a level of code implementation before the module is released.

Objective: Guarantee the small part of the software is working as designed, returning the expected results and handling errors according to the various possible inputs.

Techniques:

- Static evaluation through white box testing. Exercising the coding statements, assignments and flow.
- The unit testing will be written using Junit.

Exit Criteria: All functions have the minimum tests to guarantee if it is working correctly. The team will seek at least 90% of coverage of code testing.

3.2 System Testing

The system testing will be focused in the user cases described in the requirements and it will validate the data processing and assertion returned from basic functionalities.

Objectives: Guarantee the correct navigation, and data processing and recovery.

Techniques:

- Execution of flow and functions using invalid and valid inputs, checking if the return is according to the expected based on the requirements.
- Validate error and warning messages displayed.

Exit Criteria: All planned tests executed and all reported defects are fixed and retested.

3.3 Integration Testing

The integration tests will validate the correct functionality of module integration. For the website the integration tests will validate the integrity of the data returned from the database and its processing to the user.

Objectives: Guarantee the database access and processing are working as designed and no data corruption or lost is seen after module integration. Also, it will validate the time processing in operation of data retrieve.

Techniques:

- Data persistency will be validated after executing multiple flows between frontend and backend.
- Execution of each database access method call, with valid and invalid inputs.
- Inspect the retrieve of data ensuring no corruption.
- API testing will be used through Postman.

Exit Criteria: All planned tests executed and all reported defects are fixed and retested.

3.4 User Interface Testing

This type of test verifies if the user interaction with software is according to the requirements and also, if the user is able to execute all the functionalities and navigate through them. Here, the usability and accessibility will be evaluated as well.

Objectives: Guarantee the navigation through the application reflects the requirements and the business demands for all users. Graphics elements (windows, menus, elements size, position, focus and state) are implemented as designed.

Techniques:

- Create tests for each window aiming to verify the navigation and the elements states.
- Exercite the input of data to validate the software behaviour/output based on business rules.
- Validate links, login execution, input fields, and functions.
- Use of tools to verify the accessibility (https://achecker.ca/checker/index.php)

Exit Criteria: Validate successfully the data consistency in the interface according to an high standard accepted by the market (benchmark) and stakeholders.

3.5 Exploratory Testing

The exploratory tests validate the alternative ways not covered by systematic tests increasing the quality of the software and the testing process efficiency.

Objectives: Increase the coverage of the testing process exercising the high risks part of the software.

Techniques:

- Create a map of high risk functionalities based on business rules x low probability of failure functionalities based on the technical team to be tested.
- These tests have to be executed for the most experienced tester in the application.

Exit Criteria: All planned area to be tested are covered and all reported defects are fixed and retested.

3.6 User Acceptance Testing

These tests guarantee the software confiability based in a client perspective or in an system administrator, and also the coverage of all requirements.

Objectives: Guarantee the software was implemented according to the requirements and is able to be executed by an end user or administrator user.

Techniques:

- Validations related to the business rules assuring all items required are present and working.
- User management testing

Exit Criteria: All functionalities described in the software documentation are implemented and working as required.

4.0 CONTROL PROCEDURES

Problem Reporting

When an incident is encountered during the testing process a Change Request must be documented in a clear and concise way and assigned for a developer responsible for the specific feature.

The Change Request is a document to request a modification to the software based on an inconsistency. In this document is necessary the identification of the issue and its level of criticality with evidence (screenshot, video and logs) and all the steps to make possible the developer reproduce it in his environment. A template is available in the Appendix.

APPENDIX

Change Request Template

TITLE: The title contains a short and concise description of the inconsistency.
User informations (username and password) are visible in the URL when clicking on login button

AUTHOR AND DATE: *Diego Henrique, 10/14/2020*

PRIORITY: *LOW/MEDIUM/HIGH/URGENT*
This information will define the urgency to the request be accepted by a developer based on the business impact.

SEVERITY: *FEATURE/TEXT/TRIVIAL/MINOR/MAJOR/CRASH/BLOCK*
The severity indicates what is the impact the inconsistency can cause in the user perspective.

REPRODUCIBILITY: *ALWAYS/SOMETIMES/RANDOM/UNABLE TO REPRODUCE*
This field presents the frequency that inconsistency is reproducible.

SETUP:
In this section the required setup to reproduce the inconsistency must be listed *and also the environment and application version.*

- *Have a valid user to login in the application.*
- *Have the version 1.03 available.*

STEPS TO REPRODUCE:
Here, all the steps to reproduce the inconsistency must be listed.

1. *Launch the application.*
2. *Click in the login button (top right) to be redirected to the login page.*
3. *Insert username and password and click in the login button and observe the URL.*

EXPECTED RESULT:
The expected result based on the requirement must be described here.

3. Only the application address must be seen in the URL.

ACTUAL RESULT:
Describe what is seen in the current inconsistency. Most of the time the text will be the same as the title.

3. User information (username and password) are visible in the URL.

ADDITIONAL INFORMATION:

- This inconsistency is not seen when log in using administrator credentials.
- This inconsistency is not reproducible in a mobile environment.

ATTACHMENTS:
Here, share the screenshots, videos and logs about the inconsistency to support the Change Request.

- *UserInfosInURLLogs.txt*
- *image.png*
- *video.mp4*