API Backend Automation - Reqres IN

Test Plan

Versión 1.1

Version History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 28/jul/24 | 1.1 | Redesign of the document in English | Sebastian Muñoz |
| 07/may/24 | 1.0 | Document Creation | Sebastian Muñoz |
|  |  |  |  |
|  |  |  |  |

Table of Contents

**Content**

[1. Introduction 4](#_Toc166069336)

[1.1 Document objective 4](#_Toc166069337)

[1.2 Functionality Summary 4](#_Toc166069338)

[1.3 Scope 4](#_Toc166069339)

[2. Testing Requeriments 5](#_Toc166069340)

[3. Testing Strategy 5](#_Toc166069341)

[3.1 Tools 6](#_Toc166069342)

[4. Project Milestones 6](#_Toc166069343)

# Introduction

## Document Objective

Identify existing project information and software components to be tested.

Ensure that the developed software meets client requirements and expectations, as well as established standards.

## Functionality Summary

Automated testing implementation is required for the API components of Mieles S.A.S.'s backend. Four key system functionalities will be automated, including user search, registration, update, modification, and deletion, as well as handling unknown resources in the ReqRes API.

Ensuring proper API request and response functionality, as well as accurate parameterization based on provided documentation, is necessary.

## Scope

**Within Scope:**

Backend testing of the reqres.in API service will be automated based on the described test request. Test cases will be automated to perform expected validations according to the available API documentation.

Validations will be conducted on the reqres.in API service, based on the provided documentation included in the package. Validations will cover four API functionalities related to resource and user management, including registration, update, modification, deletion, as well as general and individual resource and user searches.

Proper validation and verification of service inputs and responses will be performed, including mandatory fields, critical paths, and alternative flows of the service. Validation will be conducted based on the provided documentation, specifications, and schemas found within the API's Swagger using appropriate protocols and documented alternative flows for each functionality.

Risks:

\*Instability in the test environment

\*Changes in documentation

\*Changes in the API without prior notice

\*Lack of clear documentation or specifications

\*Service availability

**Out of Scope:**

\* Any other functionality not described within the scope

\*Integration testing and scopes beyond those described in the document

\*Different products mentioned within the scope

\*Validation of modules, functionalities, or systems not explicitly described or implied in the provided documentation

\*Maintenance of current policies, guidelines, and standards applicable to the application

# Testing Requirements

The following list identifies items, use cases, and requirements identified for testing:

\*API Documentation: Obtain access to the API documentation via the link https://reqres.in detailing available endpoints, accepted parameters, supported HTTP methods, expected response codes, and structure.

\*Automation Tools: Ensure readiness of automation tools used for testing, including Postman and a GIT repository for managing and documenting test-related activities.

\*Identification of Endpoints to Test: Identify endpoints for testing, with the main ones defined as the four essentials: using GET method /api/users to list users, using POST method /api/users for user creation, using PUT method /api/users for user update, and using DELETE method /api/users for user deletion.

\*Test Data: Define a set of representative test data covering different use cases, including valid and invalid input data for each endpoint, as well as boundary data to verify the robustness of the API.

# Testing Strategy

For the options defined in the scope of testing, test scenarios are identified and generated to allow execution that compares the system's functionality with the specifications in the documentation supporting the project path, validating the API's behavior against expected outcomes.

Testing execution will aim to demonstrate that the development aligns with client requirements in the following manner:

1. Requirements Analysis and API Documentation: Review the provided API documentation to understand available endpoints, supported HTTP methods, required parameters, and expected responses. Key use cases that can be covered by automation testing will be identified.
2. Identification of Test Scenarios: Define a list of test scenarios based on identified use cases and API functional requirements. Initial validations are performed using Postman against the API to validate and verify identified scenarios, providing a basis for designing test scripts.
3. Development of Test Scripts: Develop test scripts using Maven as a dependency manager, utilizing Karate as the framework for test automation. This includes data preparation and execution of API requests to validate received responses.
4. Project Management and Documentation: Utilize a GIT repository to upload and maintain an updated record of all work conducted, facilitating collaboration and tracking of activities.
5. Bug Handling and Reporting: In case of bugs, detailed reports will be generated including information on executed tests, obtained results, and identified issues. This enables the development team to efficiently address and correct any identified problems.
6. Retest: Once bugs are resolved, scripts will be re-executed and updated reports generated to track closure of resolved incidents or reopening with updated feedback.
7. Testing Closure: Upon resolution of incidents, corresponding documentation and scripts from the testing process will be hosted in a GIT repository, and the completed work will be delivered.

## Tools

The following tools were used in the project:

|  |  |  |
| --- | --- | --- |
|  | Tool | Version |
| Dependency Manager | Apache-Maven | 3.9.6 |
| Version Control System | Git | 2.45.0 |
| Source Code Repository | Github | ~ |
| Documentation and Test Tracking | Microsoft Excel | 18.0 |
| Testing Framework | Karate-junit5 | 1.5.0.RC4 |
| Pre-validation Tool | Postman | 10.24.16 |

## 

# Project Milestones

|  |  |  |  |
| --- | --- | --- | --- |
| **Milestone Task** | **Effort (in hour)** | **Start Date** | **Finish Date** |
| Test Plan | 2 | 5/07/2024 | 5/07/2024 |
| Test Case design | 4 | 5/07/2024 | 5/07/2024 |
| Script development | 8 | 5/08/2024 | 5/17/2024 |
| Execution and reporting | 4 | 5/17/2024 | 5/17/2024 |
| Documentation and closure | 2 | 6/28/2024 | 6/28/2024 |