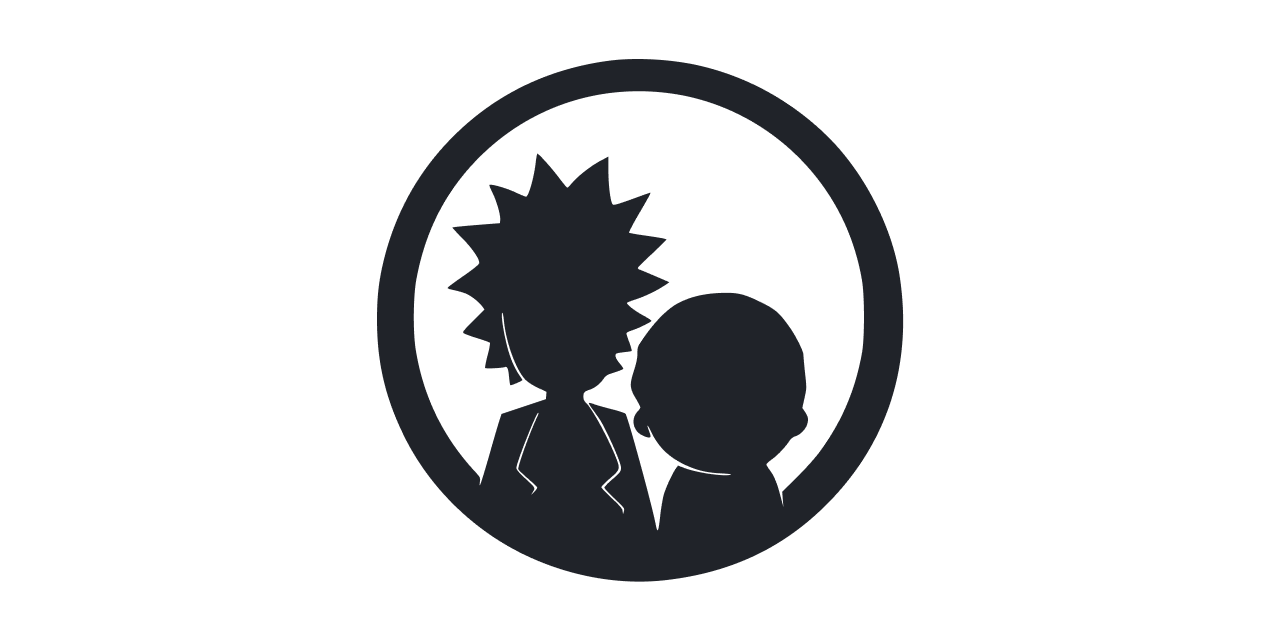
****

## **Test Plan "Rick and Morty"**

**Lorenzo Quintana Marcos**

### **1. Introduction**

#### **1.1 Purpose**

The purpose of this document is to detail the test plan for the "Rick and Morty" project. This plan describes the strategies and resources needed to conduct system testing. It focuses on ensuring that all developed functionalities meet the defined acceptance criteria and that the system is robust and free of defects.

#### **1.2 Scope**

This test plan covers the functionality tests of the Rick and Morty APIs. The tests include character search, character detail display, validation of tolerance to incomplete names, filtering characters by status, and obtaining episode details by ID.

#### **1.3 Objectives**

* Validate that character search by name works correctly.
* Verify that the character detail display shows accurate information.
* Ensure that the system offers appropriate suggestions when there are no exact matches.
* Check that the character filter by status works correctly.
* Verify that obtaining episode details by ID shows accurate information.

### **2. Testing Tools**

* **Cucumber:** For writing and executing tests in Gherkin language.
* **JUnit:** For executing unit and integration tests.
* **Rest Assured:** For testing RESTful APIs.
* **Maven:** For dependency management and test execution.
* **Postman:** For manual validation of APIs and creating test collections.
* **Cucumber Report:** For generating detailed test reports.

### **3. Acceptance Criteria and Test Cases**

#### **3.1 User Story 1: Character Search**

**Acceptance Criteria:**

* The user can search for characters by full or partial name.
* The system displays the details of the found characters.
* If there are no exact matches, the system offers suggestions.

**Test Cases:**

* **Search by Full Name:**
  + Given the user enters "Rick Sanchez" in the search field,
  + When the user presses the search button,
  + Then the system displays the details for "Rick Sanchez".
* **Search by Partial Name:**
  + Given the user enters "Rick" in the search field,
  + When the user presses the search button,
  + Then the system displays a list of characters that include "Rick" in their name.
* **Search with No Exact Match:**
  + Given the user enters "Rack Sánchez" in the search field,
  + When the user presses the search button,
  + Then the system offers suggestions close to "Rick Sanchez".

#### **3.2 User Story 2: Character Detail Display**

**Acceptance Criteria:**

* The user can obtain detailed information about a character by their ID.
* The information must include name, status, species, gender, origin, location, image, and episodes.

**Test Cases:**

* **Get Details by ID:**
  + Given the user has the ID "1",
  + When the user requests details for the character,
  + Then the system displays the details of the character with ID "1", including name, status, species, gender, origin, location, image, and episodes.

#### **3.3 User Story 3: Filter Characters by Status**

**Acceptance Criteria:**

* The user can filter characters by status (alive, dead, unknown).
* The system displays a list of characters corresponding to the requested status.

**Test Cases:**

* **Filter by Alive Status:**
  + Given the user filters characters by status "alive",
  + When the user presses the status search button,
  + Then the system displays a list of characters with status "alive".
* **Filter by Dead Status:**
  + Given the user filters characters by status "dead",
  + When the user presses the status search button,
  + Then the system displays a list of characters with status "dead".
* **Filter by Unknown Status:**
  + Given the user filters characters by status "unknown",
  + When the user presses the status search button,
  + Then the system displays a list of characters with status "unknown".

#### **3.4 User Story 4: Get Episode Information by ID**

**Acceptance Criteria:**

* The user can obtain detailed information about an episode by its ID.
* The information must include the episode name, air date, and characters appearing in the episode.

**Test Cases:**

* **Get Episode Details by ID:**
  + Given the user knows the episode ID "1",
  + When the user requests details for the episode,
  + Then the system displays the details of the episode with ID "1", including name, air date, and characters.
* **Get Episode Details by ID:**
  + Given the user knows the episode ID "2",
  + When the user requests details for the episode,
  + Then the system displays the details of the episode with ID "2", including name, air date, and characters.
* **Get Episode Details by ID:**
  + Given the user knows the episode ID "10",
  + When the user requests details for the episode,
  + Then the system displays the details of the episode with ID "10", including name, air date, and characters.

### 

### **4. Defect Management**

Defects found during testing will be recorded in a bug tracking system (e.g., Jira). Each defect will be categorized by severity and priority, and assigned to the corresponding developer for resolution.

### **5. Conclussion**

This test plan provides a detailed guide to ensure the quality and functionality of the "Rick and Morty" system. The cooperation of the entire development and QA team is required to achieve the project's objectives. With a systematic and rigorous approach to testing, it is expected that the system will deliver a high level of quality and meet the needs of the end users.