| | |
|---|---|
| **Tester** | Diogo Antequera (ISTQB-Certified QA Tester) |
| **Environment** | Swagger Petstore – Public REST API https://petstore.swagger.io/ |
| **Test Type** | API Testing / Functional API Validation / Positive & Negative Testing |
| **Objective** | Validate the core functionality, behavior and reliability of the Petstore REST API using Postman. Ensure that all endpoints behave according to specifications, status codes are correct, data integrity is maintained, and error handling is consistent across CRUD operations. |
| **Scope of Testing** | • Testing CRUD operations for Pet endpoints |
| | • Status code validation (2xx, 4xx, 5xx) |
| | • Functional validation of API responses |
| | • Payload structure & JSON schema validation |
| | • Header validation |
| | • Response time checks |
| | • Error handling (invalid data, missing fields, wrong routes) |
| | • Multiple test data scenarios |
| | • Test automation using Postman scripts (Tests tab) |
| **Out of Scope** | • Performance testing (JMeter / Load tests) |
| | • Security testing (Auth, tokens, OWASP) |
| | • UI testing |
| | • Database-level validation |
| | 1. A new Pet must be created successfully if all required fields are valid. |
| | 2. The API must return the correct Pet object when queried by Pet ID. |

| | |
|---|---|
| **Business Rules to Validate** | 3. Updating a Pet with valid fields must overwrite previous data. |
| | 4. Deleting a Pet must return a valid confirmation code. |
| | 5. Searching for a non-existing Pet must return 404. |
| | 6. Invalid payload formats must be rejected with proper error messages. |
| | 7. Required fields must not accept null or empty values. |
| | 8. API should respond within acceptable time (< 1200 ms). |
| **Tools** | • Postman (API Testing) |
| | • Postman Scripts (JavaScript tests) |
| | • JSON Schema Validation |
| | • Google Sheets for documentation |
| | • GitHub for portfolio storage |

| ID | Feature | Description | Endpoint | Method | Request Body | Expected Result | Actual Result | Status | Severity | Priority | Notes |
|---|---|---|---|---|---|---|---|---|---|---|---|
| TC_API_01 | Create Pet | Verify that a pet is created successfully with valid data. | POST /pet | POST | {"id": 12345, "name": "Rex", "status": "available"} | Status 200; response body contains same ID/name/status. | Pet created; response returned correct fields. | Pass | High | High | Pet creation workflow behaves consistently. Recommended to also test with boundary values for name and additional fields (tags, category). |
| TC_API_02 | Create Pet | Verify API rejects creation without required fields. | POST /pet | POST | {"name": "", "status": "available"} | Status 400; error message returned. | API returned 400 with error. | Pass | High | Medium | API correctly rejects missing required fields. Additional checks can include invalid data types (e.g., number for name). |
| TC_API_03 | Retrieve Pet | Verify that an existing pet can be retrieved by ID | GET /pet/12345 | GET | N/A | Status 200; correct pet details returned. | Pet found; details correct. | Pass | High | High | Retrieval stable for existing IDs. Consider testing retrieval after update and after deletion for consistency. |
| TC_API_04 | Retrieve Pet | Verify that requesting a non-existent pet returns 404. | GET /pet/999999 | GET | N/A | Status 404; “Pet not found” message. | API returned 404. | Pass | Medium | Medium | Correct 404 handling. Good to test with negative numbers and alphanumeric IDs to confirm robustness. |
| TC_API_05 | Update Pet | Verify pet info can be updated successfully. | PUT /pet | PUT | {"id": 12345, "name": "RexUpdated", "status": "sold"} | Status 200; updated fields returned correctly. | Pet updated with new values. | Pass | High | High | Update overwrites existing data as expected. Can test partial updates when the API supports PATCH. |
| TC_API_06 | Update Pet | Verify that updating a pet with non-existing ID fails | PUT /pet | PUT | {"id": 555555, "name": "Ghost", "status": "available"} | Status 404; error message. | API returned 404. | Pass | Medium | Low | Proper error returned. Additional payload variations (null fields, invalid formats) could be validated. |
| TC_API_07 | Delete Pet | Verify a pet is deleted successfully with valid ID. | DELETE /pet/12345 | DELETE | N/A | Status 200; confirmation message. | Pet deleted successfully. | Pass | High | High | Delete operation works as expected. Recommended to check retrieval after deletion to confirm entity removal. |
| TC_API_08 | Delete Pet | Verify deleting a non-existing pet returns 404. | DELETE /pet/999999 | DELETE | N/A | Status 404; error message. | API returned 404. | Pass | Medium | Low | API returns consistent 404 behavior. Could test edge IDs (0, -1, extremely large numbers). |
| TC_API_09 | Create Pet | Verify API rejects malformed JSON. | POST /pet | POST | {"id": 12345, "name": "Broken", } | Status 400; parse error. | API returned 400. | Pass | Medium | Medium | API returns correct parse error. Additional malformed cases (missing brackets, invalid quotes) may be included. |
| TC_API_10 | List Pets | Verify pets can be listed by status. | GET /pet/findByStatus?status=available | GET | N/A | Status 200; array of pets with “available” status. | Returned pets with correct status. | Pass | Medium | Medium | “available” status returned correctly. Consider testing multiple statuses simultaneously: status=available,pending. |
| TC_API_11 | List Pets | Verify that invalid status returns empty array or 400. | GET /pet/findByStatus?status=invalidtext | GET | N/A | Status 400 or empty list. | API returned empty list. | Pass | Low | Low | API correctly handles invalid query parameters. Could include long random strings or SQL-injection-like patterns. |
| TC_API_12 | Performance | Verify API response time is within acceptable limits. | GET /pet/12345 | GET | N/A | Response time < 1200 ms. | Response time = 430 ms. | Pass | Medium | Medium | Response time acceptable (<1200ms). Results may vary based on server load; ideal to run multiple samples. |
| TC_API_13 | Headers | Ensure API returns JSON content. | GET /pet/12345 | GET | N/A | Header Content-Type: application/json. | Correct header returned. | Pass | Medium | Medium | API consistently returns application/json. Additional header checks could include caching headers or encoding. |
| TC_API_14 | HTTP Methods | Verify calling GET endpoint with wrong method returns 405. | POST /pet/12345 | POST | N/A | Status 405; “Method not allowed”. | API returned 405. | Pass | Low | Low | Correct 405 behavior. Good to test GET, PUT, DELETE on invalid endpoints to ensure uniform error handling. |
| TC_API_15 | Schema Validation | Ensure new pet response matches the official JSON schema. | POST /pet | POST | {"id": 54321, "name": "Milo", "status": "available"} | Status 200; Schema validation passes. | Schema valid, keys match expected structure | Pass | High | High | Schema matches expected structure. Always recommended to validate optional fields and nested objects when available. |