1 - Technical Test: Quality Assurance Analyst

Test Plan

Candidate: Renato Alfonsetti Silva
Date: 06/11/2025
Project: Crawling API – Axur Challenge

1. Goal

Validate the functionality of the API that searches for terms provided by the user on web pages and returns the URLs where the term was found, according to specified requirements.

2. Test Scope

- **Functionalities to be tested:**
    - Start a new search via POST /crawl with a valid term.
    - Query search results via GET /crawl/{id}.
    - Validation of term length (minimum 4, maximum 32 characters).
    - Case insensitive search.
    - Validation of search ID format and generation.
    - Browsing and crawling within the same base URL (absolute and relative links).
    - Support for multiple simultaneous searches.
    - Return of partial results for ongoing searches.

- **Functionalities out of scope:**
    - Application performance under high load (unless explicitly requested).
    - Advanced security testing (in addition to basic validations).
    - Graphical interface (none).

3. Types of Planned Tests

Test Type	Description
Functional Testing	Validate all API functionalities.
Limit Tests	Validate term length limits (4 to 32 characters).
Integration Testing	Validate communication between endpoints and backend systems.
Concurrency Test	Validate multiple simultaneous searches.
API Usability Testing	Check messages, status codes and JSON formats.
Security Test	Validate inputs and avoid injection (basic).
State Regulation Test	Check active and done status of searches.

4. Casos de Teste

4.1 POST/crawl - Start search

CT-ID	Description	Prohibited	Expected Result	Observation
CT01	Valid term (e.g. "security")	{"keyword":"security"}	200 OK, JSON with 8 char alphanumeric "id"	ID should be generated automatically
CT02	Term with less than 4 characters	{"keyword":"abc"}	400 Bad Request or valid error with message	Validate minimum constraint
CT03	Term with more than 32 characters	Term with 33+ chars	400 Bad Request or valid error with message	Validate maximum constraint
CT04	Empty or missing term	{} ou {"keyword":""}	400 Bad Request	Mandatory field
CT05	Term with upper and lower case	{"keyword":"SeCuRiTy"}	200 OK, start search	Case insensitive should work

4.2 GET/crawl/{id} - Check result

CT-ID	Description	Prohibited	Expected Result	Observation
CT06	Valid ID, active search	GET /crawl/{id}	200 OK, JSON with "status":"active" and partial URLs	Partial results may appear
CT07	Valid ID, search complete	GET /crawl/{id}	200 OK, JSON with "status":"done" and final URLs	Status done indicates end of search
CT08	Invalid/non-existent ID	GET /crawl/invalidid	404 Not Found or appropriate error	ID Validation
CT09	Check URL returns	URLs must have the same base URL	Correct URLs as per crawling	Validate crawling based on URL

4.3 Behavior and rules tests

CT-ID	Description	Prohibited	Expected Result	Observation
CT10	Check ID generation	Valid POST	ID with 8 alphanumeric characters	Usar regex [a-zA-Z0-9]{8}
CT11	Search term with case variations	security, Security, SECURITY	Equivalent results	Case insensitive
CT12	Check crawling on base	Test external	External links should	Crawling

	URL only	links	NOT be visited	restricted to base URL
CT13	Simultaneous execution of searches	2+ simultaneous POSTs	Each search independent, status maintained	Test concurrency
CT14	Return of partial results	GET query in active search	Partial URLs available on return	"Active" status and partial return

5. Acceptance Criteria

- The API must respond with the correct HTTP codes (200, 400, 404).
- The search only accepts terms from 4 to 32 characters.
- The search ID must be a unique, 8-character alphanumeric.
- The returned URLs must respect the base URL and follow the links correctly.
- It must support multiple simultaneous searches without conflict.
- Search statuses must reflect the actual state (active or done).
- Partial return of results for ongoing searches must be possible.
- The system must not crash or lose data while running.

6. Test Execution Strategy and Tools

6.1 Manual and Automated Execution

- **Manual Testing:**
  For initial validation of endpoints, using tools such as Postman or Insomnia to send HTTP requests, analyze responses and perform exploratory testing.
- **Automated Testing:**
  Create automated test suites using frameworks such as:
    - REST Assured (Java) or pytest + requests (Python) for API testing.
    - JMeter for load/concurrency testing if necessary.

6.2 Monitoring and Logs

- Use application logs to validate internal behavior, ID generation, and search status.
- Monitor environment variables to ensure that the base URL is correct during testing.

6.3 Test Environment

- Set up an isolated environment with the correct base URL environment variable.
- Ensure the application is running on port 4567 for testing.

7. Risks and Considerations

- Performance may vary depending on the size of the site being crawled.
- The application depends on the network status and the target site for real tests.
- Possible unhandled errors in search (e.g. dead links, redirects).

8. Conclusion

This plan covers the complete validation of the functional requirements and business rules of the described API, ensuring that the product meets the expectations of quality, robustness and usability. The execution of tests, both manual and automated, allows for early identification of errors and ensures the delivery of reliable software.

2 - Teste técnico: Quality Assurance Analyst

- Automação

Made with Cypress with JavaScript.