Quality Assurance and Testing

Table of Contents:

Overview

We aim to make “Quality Assurance” an essential part of our software development and working hard towards this goal. Unit Tests, Integration Tests, Acceptance Tests and Performance tests will help us to achieve it. Alongside these tests, we already integrated SonarQube, an open source quality management platform, dedicated to continuously analyze and measure source code quality. We aim to improve SonarQube rules and incorporate the suggestions and reduce the technical debt reported.

The goal of this document is to describe which actions are currently being carried out to improve the testing and quality assurance (QA) of OpenCGA, this will allow to improve the software quality (reliability, efficiency, security, maintainability, ...). This document will try to clearly describe:

- Which test scenarios we will cover and which we will not
- Code Quality Assurance (QA) goals to improve the source code
- Roles and responsibilities of the team members
- Environment for running the tests
- Tools to be used
- Roadmap and schedule

Test Scenarios

OpenCGA is complex application with several components and layers, e.g. database, Java APIs, RESTful web services, client libs, command lines (CLI), … Different levels of testing are needed, some of them have already been implemented and others need to be done.

Scope

We have developed a good number of different test sets and continuously adding more to make sure our software adheres best quality. Different test sets achieve different level of testing as explained in next sections.

Unit Tests

These tests aim to test individual functionalities and give us confidence that developed code is right, usually isolated from the rest of the system, e.g. loading some variants, create a study in catalog, … These tests show that the implemented code is right, which is very important but not enough. OpenCGA currently has more than 600 JUnit tests and more are added every week. As you know the goal for RC3 is to have a test coverage of 80%. No further actions are need here I think.

Integration Tests

These aim to test some components -or even the whole system- to study how they work together. Currently we have some top level tests for this but a lot of work is needed here

Acceptance Tests

Acceptance tests will make sure we are delivering the “Right Code”. Ideally these tests should cover all the possible scenarios both positive and negative and help us to adjust API for best experience. These tests are created using requirements documents, stands as agreement between client and development team and should stay updated . Any changes in requirement should be reflected into acceptance tests. OpenCGA exposes its functionality through RESTful web services (and gRPC in some few cases) and it is accessible using different user interfaces such as client libs (Java, Python, …) and command lines (CLI s). Also some web apps are being developed.

This architecture has many benefits, one of them is that all the user interfaces query the RESTful web services, so by testing the REST web services:

- We are testing the whole system: catalog, storage databases, security, …
- You are ensuring to the client libs and CLIs that the system is working correctly, and this alleviates the tests needed for client libs and CLIs.

Another benefit is that quite straightforward to implement stress tests by creating a high load to the REST servers.

Performance/Stress Tests

These tests will check performance of OpenCGA core APIs (Permissions) under different stress levels. These tests are not intended to verify each and every possible scenario in OpenCGA (that's part of Acceptance Tests) rather behaviour of systems with different load, scenarios (empty db, different configurations settings etc). By testing core API, we will get valuable information about performance of our system and will help us to tweak our Architecture and Design for better performance. Jmeter is the suggested tool.

Tools

Some of the tools we use for testing include

- JUnit
- Apache JMeter
- FitNesse
- Jenkins
- SonarQube