HW1: Mid-term assignment

I. Oliveira, v2020-03-27

Objectives

Develop a **multi-layer web application**, in Spring Boot, supplied with automated tests (at multiple levels).
This is an **individual** assignment.

Required elements

Project scope

Your application should provide details on **air quality** for a certain region/city. The air quality can be characterized with different metrics, such as particles in suspension (PM10,...) or gases (CO, Ozone,...). It may be the current, past, or future (forecast) values, depending on the data sources you will use.
Variations on the theme are acceptable, as long as related to public health surveillance (e.g.: pollen concentration, meteorology hazard alerts, etc.)

The project must include **different components:**

1. A **web page**, minimalist, which allows users to enter or select a location and access its air quality values (it can be more than one page but keep it simple).
2. **Integration with external source(s)**. Actual air quality data should be fetched from a remote service. That is, your backend services will act as a client for a third-party API[1], which you will access to obtain the values.

   The solution should also implement internally a local [2] (in-memory) *cache,* with the latest results (from the external source) copied locally; if the same data is requested, you should use the *cached* values. The *cache* should define a *time-to-live* policy and produce some basic statistics about its operation (count of requests, *hits/misses),* which you do not need to save persistently [you may use a data structure, like a HashMap, or in-memory database].
3. Your own **API (REST)** that can be invoked by external clients. Your API should allow one to programmatically interrogate (your) backend for useful air quality data (e.g.: by location, by days,...). Your API should also provide basic statistics on the use of the (internal) *cache*.

Technologies stack

The solution should be based on Spring Boot for the services/backend. The web layer can be done with any HTML/JavaScript framework, or using a templating system that integrates with Spring Boot (e.g.: thymeleaf).

---

[1] There are several API available in the Internet to get air quality data. Make your own survey and assessment; then choose one that you find easy to use to integrate in your project.
[2] This structure should be implemented (albeit in a basic way); it is not intended to reuse existing libraries (e.g.: JCache, Spring Boot Cache Manager).

Tests to implement

The project should include the automation of different types of tests:

A) Unit tests (suggestion: tests on *cache* management; tests on "validators", "converters" or "utils" in general, if applicable).

B) Service level tests, with dependency isolation using *mocks* (suggestion: Unit test with isolation from external service).

C) Integration tests on your API. Suggested technology: Spring Boot MockMvc and/or REST-Assured.

D) Functional testing (on the web interface). Suggested technology: Selenium WebDriver.

Quality metrics

The project should include quality metrics (as available, for e.g., from Sonar Qube). To do this, you should also implement:

E) Integration of analysis with Sonar Qube. Note: If the Git project is public, it can be analyzed in SonarCloud.

Extra points (optional)

For extra points:

F) The project uses more than one external source (remote API) and switches between one and the other, either by configuration, or upon the (in)availability of the sources.

G) The project uses a Continuous Integration framework, with the automation of testing and static code analysis.

Submission

The submission consists of a brief report and a Maven-based code project.

1. The Technical Report should **explain the strategy** that was adopted (your options as a developer) and offer **evidence** of the results obtained (e.g.: which testes per testing level, screenshots of the representative steps, (small) code snippets of the key parts, screenshots with the test results, etc.). The results of the Sonar Qube dashboard should be included (and discussed).

   There is a template for the report.

2. The Git repository with the solution. The repository location (URL) should appear at the beginning of the report.

   Besides the code, be sure to include in the repository, a short video with a demonstration of your solution (or a link to the video, instead, if the video is a large file).