HW1: Mid-term assignment

I. Oliveira, v2022-04-08

Objectives

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

Required elements

Project scope

Your (web) application should provide details on **COVID incidence data** for a certain country/territory. You should consider at least the number of (new) cases for a day or period, but you may add more data (e.g.: data aggregation). Variations on the theme are acceptable and welcome, if related to public health surveillance.

The project must include **different components** (Figure 1)**:**

1. A **web app**, minimalist, which allows users to enter or select a region/country and access its Covid metrics, e.g.: last 3 days.
2. **Integration with external source(s)**. Actual data should be retrieved from a remote service. Your backend services will function as a client for a third-party API[1], which you will access to obtain the required values.
3. **Implement a *cache***[2] to reduce the number of external accesses. Whenever a request is made to a remote API, you should cache the results; if the same (remote) data is requested (i.e., a duplicate request) then 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)*. All data related to the *cache* operations is not required to be persisted [you may use a memory data structure, like a HashMap].
4. Provide your **own REST API** to expose COVID monitoring data to be invoked by external clients. Your API should allow to programmatically interrogate (your) backend for useful covid tracking data (e.g.: filtering by region, by days,...).

   Your API should also provide an endpoint to get basic statistics on the use of the (internal) *cache*.
5. Use a logging strategy to produce useful evidence of the actions/events that happened while using your software, for later inspection/debugging. Be sure to apply a standard logging library.

[1] There are several API available (e.g.: https://rapidapi.com/collection/coronavirus-covid-19). Make your own survey and assessment, then choose one that you find easy to use to integrate in your project.
[2] This should be a simple cache data structure, **implemented by you**; avoid the use of existing caching libraries (e.g.: JCache, Spring Boot Cache Manager).

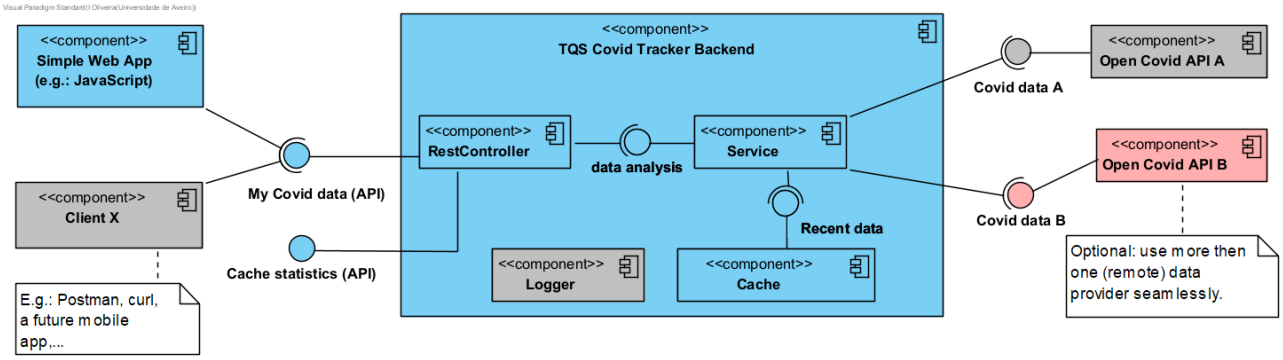

Figure 1: Reference architecture: you need to implemente the blue elements; gray componentes represent integrations; pink elements are optional.

Technologies stack

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

Tests to implement

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

A) Unit tests (suggestion: tests on the *cache* behavior; tests on "data validation", "converters" or "utils" in general, if applicable).

B) Service level tests, with dependency isolation using *mocks* (suggestion: test your services in isolation from external data provider).

C) Integration tests on your API (suggestion Spring Boot MockMvc and/or REST-Assured).

D) Functional testing (on the web interface). Be sure to adopt a Behavior-driven approach (features with scenarios).

Quality metrics

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

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

Extra points (optional)

For extra points:

F) The project works with more than one external source (remote API). It proactively switches between one or the other, either by configuration, or upon the (un)availability of the sources.

G) The project uses a Continuous Integration framework, with the automation of testing and static code analysis (e.g.: GitHub Actions, GitLab CI/CD)

Submission

The submission consists of a brief report and a Maven-based code project, including:

A brief **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). [A report template will be provided.]

1. The **code project**, committed to your TQS personal Git repository (folder /HW1).
   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).
2. **Oral presentation**. To be scheduled.