deti universidade de aveiro
departamento de eletrónica, telecomunicações e informática

HW1: Mid-term assignment report

Ana Alexandra Antunes [876543], v2022-04-07

1 Introduction

1.1 Overview of the work

This report presents the midterm individual project required for TQS, covering both the software product features and the adopted quality assurance strategy.

This project is a web application that provides details on COVID incidence data for a certain country, city, region and date. These details include the number of deaths, confirmed cases, recovery, etc.

1.2 Current limitations

This application is integrated with a third-party API to retrieve data so it's highly dependent on it. So, if for example, the API goes down, this application can no longer return data.

Another limitation is the covid metrics for a certain region or city, because it doesn't provide information for every region or at most every city and it doesn't display this availability to the user. So the user doesn't know which city/region this application has information about.

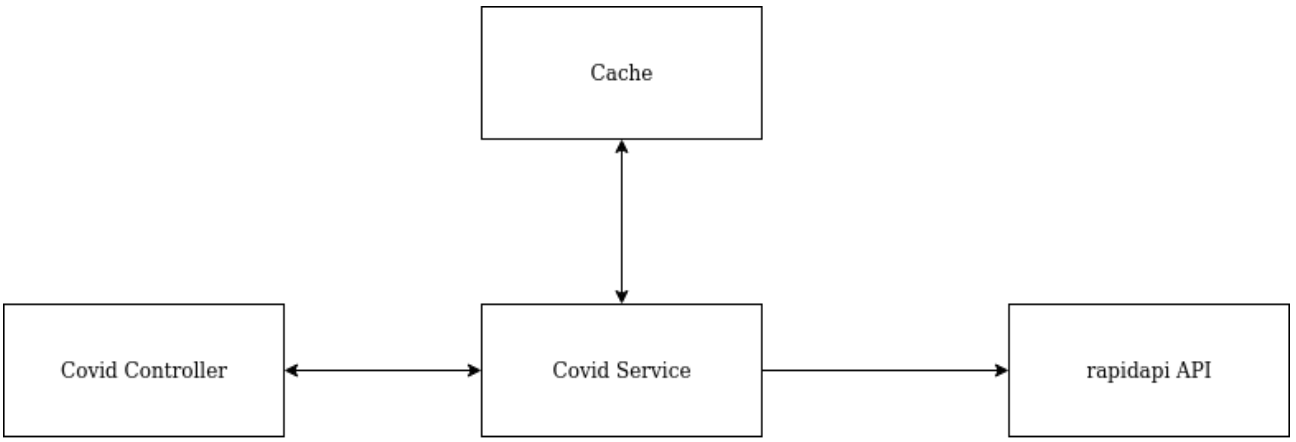
2 Product specification

2.1 Functional scope and supported interactions

The target Audience for this application is people who want information about covid incidents in a certain country. Anyone who wants to know the number of confirmed cases, deaths and recoveries from covid for a certain region, city or country in a certain date can use this application to accomplish that objective.

2.2 System architecture

The architecture has three main parts: The user interface, which is done using Thymeleaf, the backend application, done using Spring Boot and the external API, rapidapi.



2.3 API for developers

A developer can obtain covid metrics from a country, region and city.

To document the api endpoints and schemas, i used the swagger library and this information is available in the page http://localhost:8080/swagger-ui/index.html

3 Quality assurance

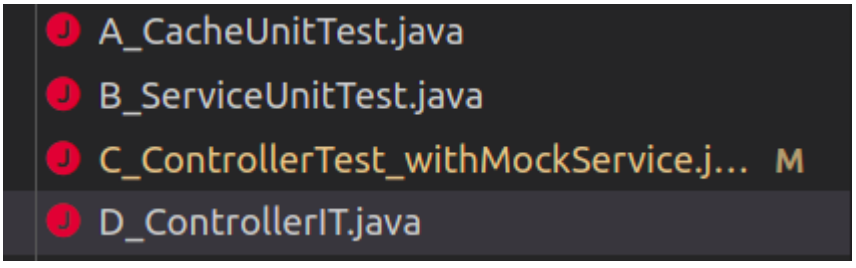
3.1 Overall strategy for testing

The strategy for test development intended to be implemented was Time driven development, however it was not always possible due to lack of experience, knowledge and time. For the functional testing to adopt a Behavior-driven approach, cucumber was used since it is the most familiar tool to use features with scenarios.

3.2 Unit and integration testing

I used unit testing for the controller class, for the cache, and for the service. The controller class was used to test the return of the data. The cache was used to save data, the time to live policy, delete data and save hitMiss statistics. The service was used to test the request with the third party API and the information from the cache.

I also used an integration test in the controller class to test the functionality of the overall application.



3.3 Functional testing

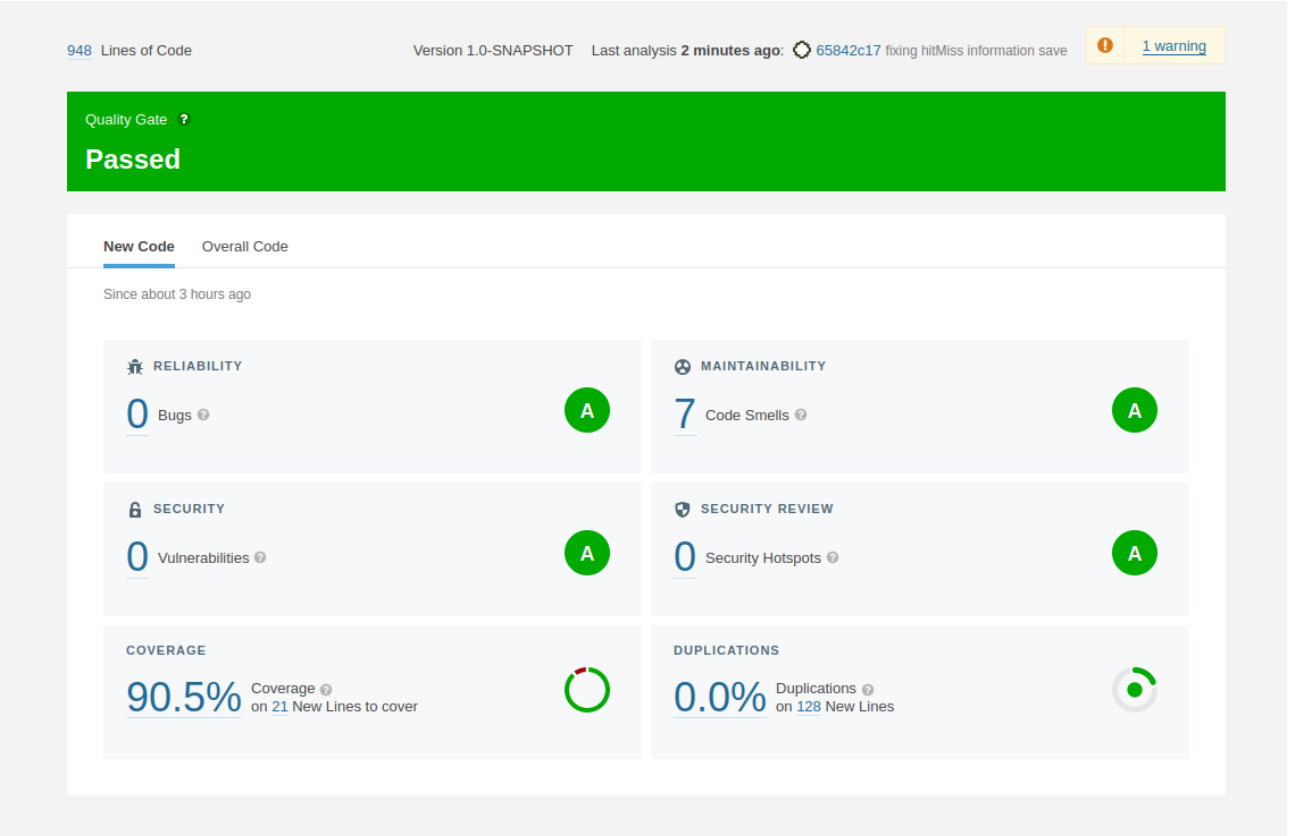
To create user-facing test cases was a made one feature for the hit/misses statistics and the covid metrics with three scenarios (I tried to separate it into two distinguishing features, however I faced an error beyond my abilities so this was my solution). The first one to check the statistic (all at 0), then make a request and check the response and finally check again the statistics to see if they have changed.

```
Feature: Get the COVID incidence data
 Scenario: No requests were made
    When I ask for the hitMiss statistics
    Then the total value is 0, the hit  value is 0 and the miss value is 0
  Scenario: Search Data by City Name, Country Code, Region name and date
      When I search for covid data from  the city 'Adams', country code 'USA', date '2022-04-28' and Region name 'US'
      Then data should appear, like for example, the province 'Washington'
  Scenario: A request was made
    When I ask for the hitMiss statistics
    Then the total value is 1, the hit  value is 0 and the miss value is 1
```

3.4 Code quality analysis

The tool used to code quality analysis was sonarcloud. A lesson learned by this tool is that while it's fine to use the expression "e.printStackTrace()" when the application is on production for debugging purpose, it is unwise when the application is on the run, instead it is suggested to use a logger.

```
try {
    response = client.newCall(request).execute();
} catch (IOException e) {
    logger.error( format: "context",e.getMessage());;
}
```



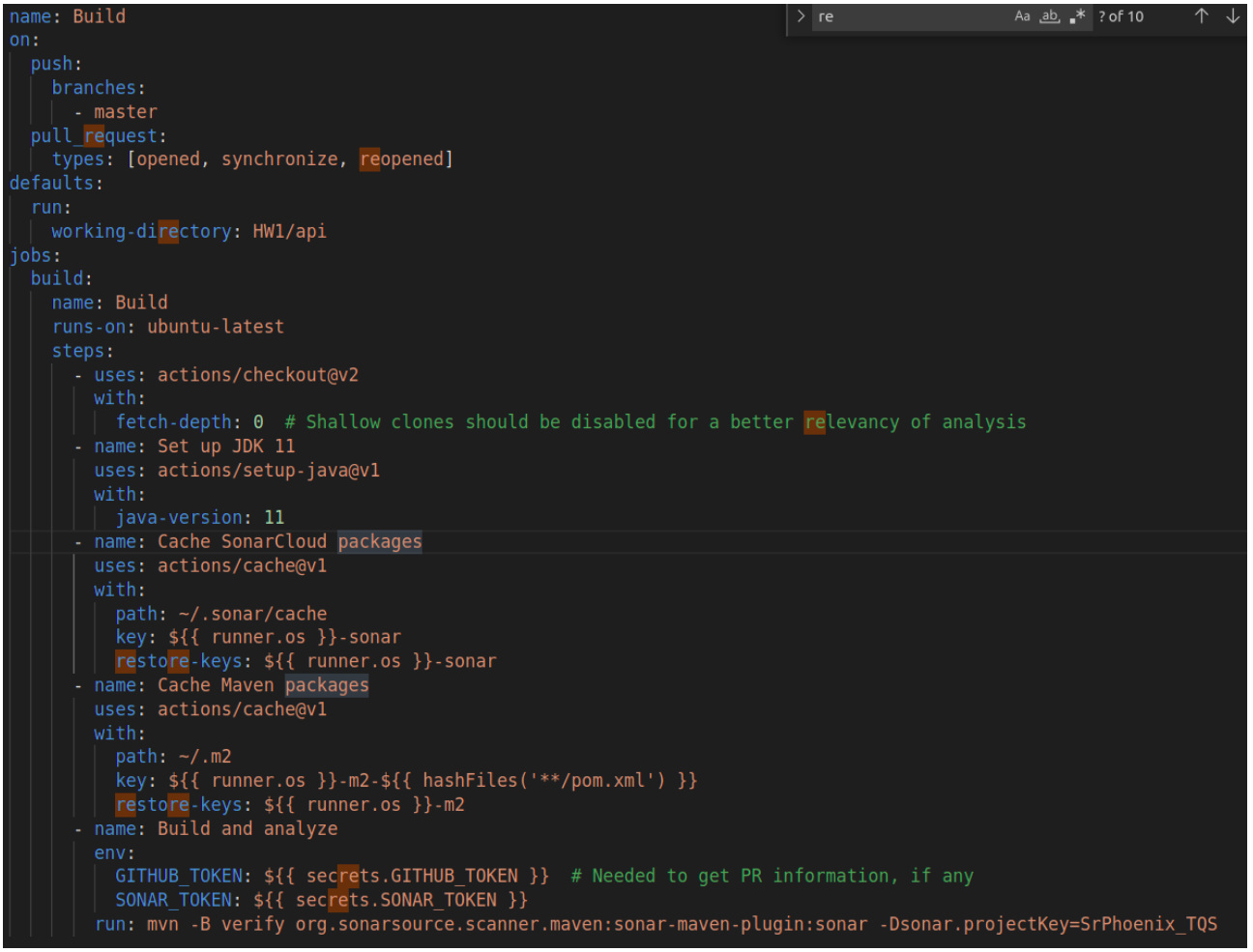
In this test the application passed the quality gate with 0 bugs, 7 code smells, 0 vulnerabilities, 0 security hotspots, 0 duplications and 90.5% code coverage

The code smells are all in the codeservice class where 5 of them are “String contains no format specifiers” (logger) and the other two are the use of the exception classes.

```
49          }
50
51  vice…   private ResponseDataArray request(String date, String region, String country, String city) throws URISyntaxExcepti
52  vice…       StringBuilder url = new StringBuilder("https://covid-19-statistics.p.rapidapi.com/reports?");
53              if (!date.equals(""))
54                  url.append("date="+date);
55  vice…       if (!region.equals(""))
56  vice…           url.append("&region_name="+region);
57  vice…       if (!country.equals(""))
58  vice…           url.append("&iso="+country);
59  vice…       if (!city.equals(""))
60  vice…           url.append("&city_name="+city);
61  vice…       logger.info("url: ", url.toString());
62  vice…
63  vice…       Request request = new Request.Builder()
64                  .url(url.toString())
65                  .get()
66                  .addHeader("X-RapidAPI-Host", "covid-19-statistics.p.rapidapi.com")
67                  .addHeader("X-RapidAPI-Key", "54a04285f3msh775e05c8199e3d1p10dee3jsn18132b791694")
68                  .build();
69  vice…           Response response = null;
70                  try {
71                      response = client.newCall(request).execute();
72                  } catch (IOException e) {
73  vice…               logger.error("context",e.getMessage());
74  vice…           }
75  vice…       logger.info("Response is Successful: " , response.isSuccessful());
76  vice…       String responseStr = response.body().string();
77  vice…       logger.info("Response body: ", responseStr);
78  vice…       ResponseDataArray result = convertToResponseData(responseStr);
79  vice…
80  vice…       return result;
81  vice…
82  vice…   }
83
84  vice…   public ResponseDataArray convertToResponseData(String response) {
85              logger.info("converting to POJO");
86  vice…       logger.info("JSONObject: " , response);
87  vice…
```

3.5 Continuous integration pipeline [optional]

I used sonarCloud to implement a continuous integration pipeline so every time a pull request or push is made it automatically tests the application.



```yaml
name: Build
on:
  push:
    branches:
      - master
  pull_request:
    types: [opened, synchronize, reopened]
defaults:
  run:
    working-directory: HW1/api
jobs:
  build:
    name: Build
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
        with:
          fetch-depth: 0  # Shallow clones should be disabled for a better relevancy of analysis
      - name: Set up JDK 11
        uses: actions/setup-java@v1
        with:
          java-version: 11
      - name: Cache SonarCloud packages
        uses: actions/cache@v1
        with:
          path: ~/.sonar/cache
          key: ${{ runner.os }}-sonar
          restore-keys: ${{ runner.os }}-sonar
      - name: Cache Maven packages
        uses: actions/cache@v1
        with:
          path: ~/.m2
          key: ${{ runner.os }}-m2-${{ hashFiles('**/pom.xml') }}
          restore-keys: ${{ runner.os }}-m2
      - name: Build and analyze
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}  # Needed to get PR information, if any
          SONAR_TOKEN: ${{ secrets.SONAR_TOKEN }}
        run: mvn -B verify org.sonarsource.scanner.maven:sonar-maven-plugin:sonar -Dsonar.projectKey=SrPhoenix_TQS
```

4 References & resources

Project resources

Resource:	URL/location:
Git repository	https://github.com/SrPhoenix/TQS/tree/master/HW1
Video demo	available in the repository
QA dashboard (online)	https://sonarcloud.io/project/configuration?id=SrPhoenix_TQS
CI pipeline	https://github.com/SrPhoenix/TQS/blob/master/.github/workflows/build.yml

Reference materials

https://rapidapi.com/axisbits-axisbits-default/api/covid-19-statistics/
swagger - https://www.baeldung.com/spring-rest-openapi-documentation