deti universidade de aveiro
departamento de eletrónica, telecomunicações e informática

HW1: Mid-term assignment report

José Pedro Marta Trigo [98597], v2022-05-02

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.

The assignment features many components such as an web app built with the React framework/library which shows several covid stats globally, by country, and by date; an API built with Java Spring Boot which integrates with external sources; and a caching system to reduce the number of external accesses.

1.2 Current limitations

Although two external API services were used, they don't proactively switch between one or the other, either by configuration, or upon the (un)availability of the sources. Instead, they were both used because they provided complementary information to each other.

2 Product specification

2.1 Functional scope and supported interactions

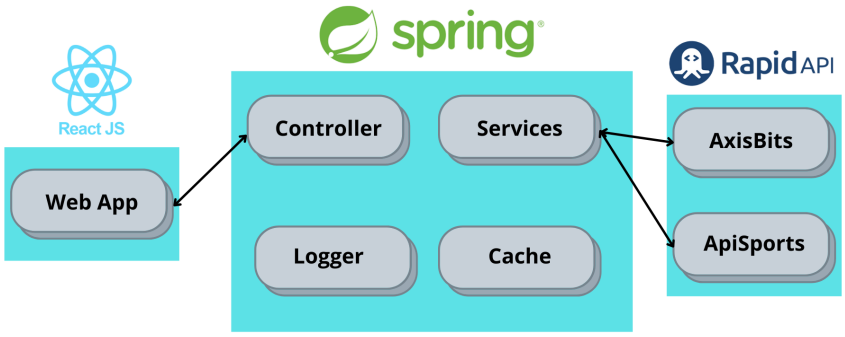
The main user of the application is someone who is interested about COVID-19 statistics.

He has access to global COVID-19 statistics filtered by date and also to individual country statistics, also supplied with the option to filter by date.

There is an extra page where the user can see web requests information (how many requests were cached and how many were externally accessed).

2.2 System architecture

- Frontend: React
- Backend: Java Spring Boot
- External APIs



2.3 API for developers

The api was tested and documented using Postman. Documentation available here.

GET Global stats

localhost:8080?date=2022-04-04

PARAMS

date	2022-04-04 (Optional)

```
curl --location --request GET 'localhost:8080
```

Example Response

Body Header (0)


```
{
  "data": {
    "date": "2022-04-04",
    "last_update": "2022-04-05 04:20:48",
    "confirmed": 493673448,
    "confirmed_diff": 2232805,
    "deaths": 6169905,
    "deaths_diff": 17018,
    "recovered": 0,
    "recovered_diff": 0,
```


View Mo

GET Countries list

localhost:8080/v1/countries_list

Example Request

```
curl --location --request GET 'localhost:8080
```

Example Response

Body Header (0)


```
{
  "get": "countries",
  "parameters": [],
  "errors": [],
  "results": 233,
  "response": [
    "Afghanistan",
    "Albania",
    "Algeria",
    "Andorra",
```


View Mo

GET Country Stats

localhost:8080/v1/country_stats?country=Portugal&date=2022-04-04

PARAMS

country	Portugal Required*
date	2022-04-04 (Optional)

Example Request

```
curl --location --request GET 'localhost:8080
```

Example Response

Body Header (0)


```
{
  "get": "history",
  "parameters": {
    "country": "Portugal",
    "day": "2022-04-04"
  },
  "errors": [],
  "results": 2,
  "response": [
    {
```


View Mo

GET Cache

localhost:8080/v1/cache

Example Request

```
curl --location --request GET 'localhost:8080
```

Example Response

Body Header (0)


```
{
  "hits": 4,
  "misses": 4,
  "requests": 8
}
```

3 Quality assurance

3.1 Overall strategy for testing

Although Test Driven Development was tried, it was not always possible because of the limited hindsight of a Junior developer.

3.2 Unit and integration testing

Regarding unit and integration testing, most of the code was covered by the tests, especially the cache and controller components, reaching a total coverage of more than 90%.

There was a rigorous attempt to try and think about every possible scenario, breaking point and exception to test, even going as far as using random number generators to add a degree of variability to the tests.

```java
@Test
void testUpdateCacheStats() {
    int n;
    long hash;
    String data;

    // store n values in cache
    n = getRandomNumber( min: 2,  max: 10);
    for (int i = 0; i < n; i++) {
        data = "hello" + i;
        hash = cache.generateHash(data);
        cache.store(hash, data);
    }

    // Try to retrieve n values present in cache and n values not in cache
    for (int i = 0; i < n; i++) {
        cache.retrieve((long) i); // n misses
        cache.retrieve(cache.generateHash("hello" + i)); // n hits
    }

    assertEquals(n, cache.getMisses());
    assertEquals(n, cache.getHits());
    assertEquals(2 * n, cache.getRequests());

}
```

3.3 Functional testing

In regards to functional testing, the chosen tool to make these tests was the selenium framework.

There were considered three scenarios:

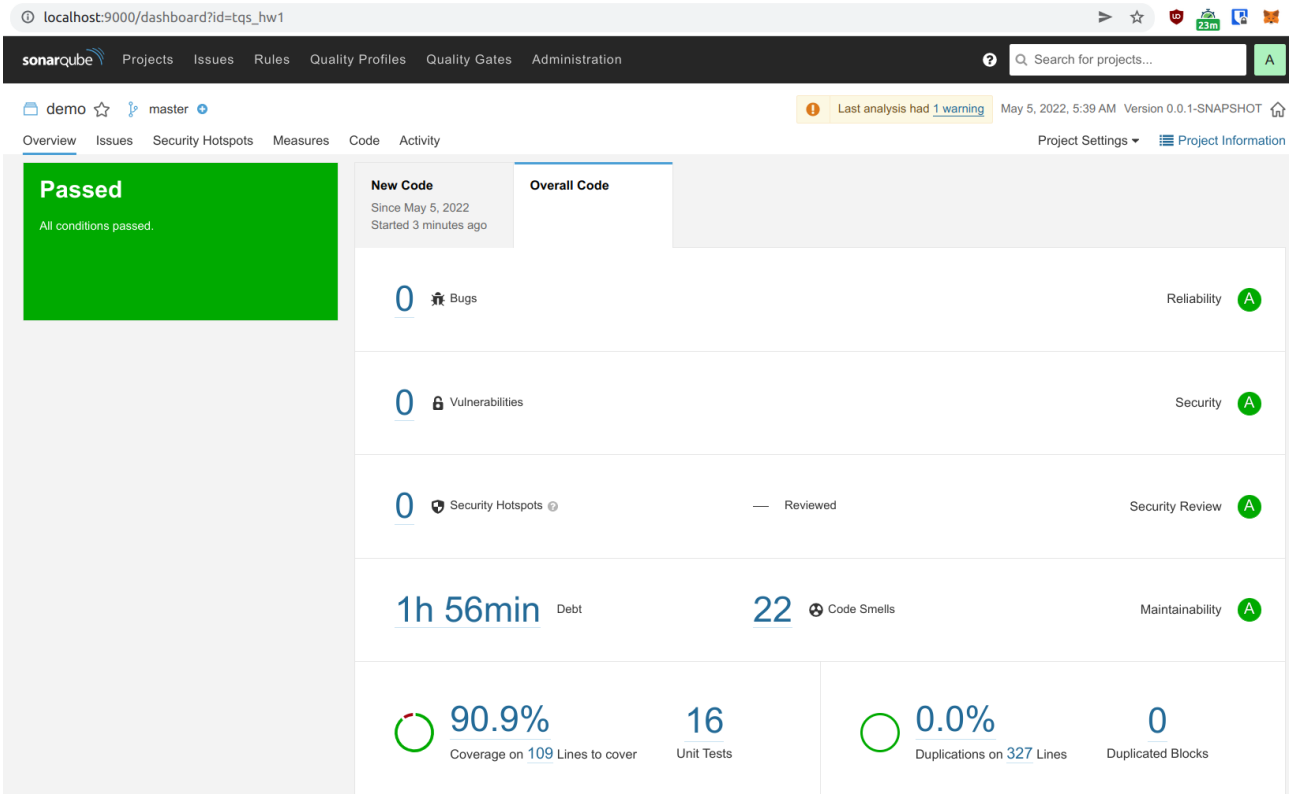
```
Feature: Example
    Scenario: Search for world data in a given day
        When I load the webpage 'http://localhost:3000/'
        And I choose a date on the calendar
        And I click the search button
        Then data field with id 'last-updated-text' is loaded from the API

    Scenario: Search for Regional data for a given Country and a given day
        When I load the webpage 'http://localhost:3000/'
        And I click tab with id 'side-bar-opt-1' in the sidebar
        And I choose a date on the calendar
        And I choose 'Poland' from the select element
        And I click the search button
        Then data field with id 'country-population-val' is loaded from the API

    Scenario: See Cache stats
        When I load the webpage 'http://localhost:3000/'
        And I click tab with id 'side-bar-opt-2' in the sidebar
        Then data field with id 'cache-requests-val' is loaded from the API
```

3.4 Code quality analysis

To do a static code analysis, SonarQube was used. The results were positive, achieving greater than 90% code coverage.



4 References & resources

Project resources

Resource:	URL/location:
Git repository	https://github.com/zepedrotrigo/tqs_98597/tree/main/hw1
Video demo	https://www.youtube.com/watch?v=c3hFhRNWMOk
QA dashboard (online)	(Run manually and locally, screenshot in report and also in video demo)

Reference materials

- https://docs.sonarqube.org/latest/setup/get-started-2-minutes/
- https://rapidapi.com/api-sports/api/covid-193/
- https://rapidapi.com/axisbits-axisbits-default/api/covid-19-statistics/
- https://spring.io/guides/tutorials/rest/