HW1: Mid-term assignment report

Theo Magno Bellinato Filgueiras Menezes [96145], v2022-05-0302

1 Introduction 1
1.1 Overview of the work 1
1.2 Current limitations 1

2 Product specification 2
2.1 Functional scope and supported interactions 2
2.2 System architecture 2
2.3 API for developers 2

3 Quality assurance 2
3.1 Overall strategy for testing 2
3.2 Unit and integration testing 2
3.3 Functional testing 4
3.4 Code quality analysis 5

4 References & resources 5

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.

I have developed an application for COVID tracking and named it COVID Tracker. It allows people to check statistics of COVID cases, deaths, and tests for a lot of countries. The application uses information from an open API, which is referenced in the end of this report.

1.2 Current limitations

As it uses an external API, the main limitation of this application is the dependency of the API to work properly. The best way to avoid this problem is turning it independent from any external products. Another way around is using different APIs, if one of them is no longer working, we use another one and so on. Although, I have not developed the system to work with others APIs, so this limitation will remain.

2 Product specification

2.1 Functional scope and supported interactions

This application will be mainly used by people who are trying to check recent COVID statistics (current or last day). These people may be doctors, researchers or just people worried about COVID.

They will search for a country with the search or select one from the list and after that they will have cases, deaths and tests statistics from the period mentioned above.

2.2 System architecture

This application was developed with SpringBoot in the backend. For the frontend I have used Thymeleaf to use HTML templates, inside the SpringBoot application, with jQuery framework for JavaScript.

2.3 API for developers

- COVID tracking data endpoints:

  GET: ***/api/country*** get all countries under tracking.

  GET: ***/api/country/{name}*** search for countries by name (it returns countries that contains *string* **name**.

  GET: ***/api/statistics/{name}*** get last day's statistics from a country.

- Cache usage statistics:

  GET: ***/api/cache*** get cache usage statistics (number of requests, hits and misses).

3 Quality assurance

3.1 Overall strategy for testing

My test development strategy was TDD (test driven development). For the tests I have used JUnit for the unit tests, Mockito for service level tests (to *mock* dependencies), SpringBoot MockMvc for integration tests and Cucumber with Selenium Web Driver for functional tests.

3.2 Unit and integration testing

The unit tests were used to check the cache and its statistics. Integration tests were used to test the API and the logger.

On the file UnitTests.java the were done three unit tests, one to check expiration time of the information in cache, another one to test the statistics (number of requests, hits and misses) and the last one to check if the information is in cache before the expiration time.

- **check-expiration-test():**

```
@Test
void check_expiration_test() throws InterruptedException {
    ArrayList<String> brazil = new ArrayList<>();
    brazil.add("Test");
    brazil.add(String.valueOf(Instant.now().getEpochSecond()));
    cache.put( name: "Brazil", brazil);
    TimeUnit.SECONDS.sleep( timeout: 30);
    assertEquals( expected: "", cache.checkCache( country: "Brazil"));
}
```

- **check-cache-stats-test():**

```
@Test
void check_cache_stats_test() throws InterruptedException {
    ArrayList<String> brazil = new ArrayList<>();
    brazil.add("Test");
    brazil.add(String.valueOf(Instant.now().getEpochSecond()));
    cache.put( name: "Brazil", brazil);
    for (int i=0; i<3; i++) {
        TimeUnit.SECONDS.sleep( timeout: 10);
        cache.checkCache( country: "Brazil");
    }
    assertEquals( expected: 3, cache.getCache_requests());
    assertEquals( expected: 2, cache.getCache_hits());
    assertEquals( expected: 1, cache.getCache_miss());
}
```

- **check-cache-test():**

```
@Test
void check_cache_test() {
    ArrayList<String> brazil = new ArrayList<>();
    brazil.add("Test");
    brazil.add(String.valueOf(Instant.now().getEpochSecond()));
    cache.put( name: "Brazil", brazil);
    assertEquals( expected: "Test", cache.checkCache( country: "Brazil"));
}
```

For the integration tests I have done only one for the API integration, with the following methods:

- **when_getAllCountries():**

```
@Test
void when_getAllCountries() throws Exception {
    mvc.perform(get( urlTemplate: "/api/country").contentType(MediaType.APPLICATION_JSON)).andExpect(status().isOk())
            .andExpect(jsonPath( expression: "$", hasSize(233)))
            .andExpect(jsonPath( expression: "$[0]", is( value: "Afghanistan")));
}
```

- **when_getBra():**

```java
@Test
void when_getBra() throws Exception {
    mvc.perform(get( urlTemplate: "/api/country/Bra").contentType(MediaType.APPLICATION_JSON)).andExpect(status().isOk())
            .andExpect(jsonPath( expression: "$", hasSize(2)))
            .andExpect(jsonPath( expression: "$[0]", is( value: "Brazil")));
}
```

- **when_getBrazilStats():**

```java
@Test
void when_getBrazilStats() throws Exception {
    mvc.perform(get( urlTemplate: "/api/country/Bra").contentType(MediaType.TEXT_PLAIN)).andExpect(status().isOk())
            .andExpect(jsonPath( expression: "$[0]", is( value: "Brazil")));
}
```

3.3 Functional testing

The functional test case I have used was the following one:

- Go to the WebApp home page;
- Search for “Bra”;
- Click the first country to appear on the list (Brazil);
- Check information (Country name and continent).

This test was done with Cucumber and Selenium WebDriver. With Cucumber I have written the following feature:

```
Feature: Check a country's COVID cases

  Scenario: Brazil
    Given go to the WebApp home page
    When I search for "Bra"
    When click on "Brazil"
    Then country's name is "Brazil" and continent is "South-America"
```

With this feature these tests were made:

```java
package tqs5.hw1;

import ...

public class WebAppSteps {
    private final WebDriver driver = new FirefoxDriver();
    @Given("go to the WebApp home page")
    public void go_to_the_web_app_home_page() {
        driver.get("http://localhost:8080");
    }
    @When("I search for {string}")
    public void search_for(String query) throws InterruptedException {
        WebElement element = driver.findElement(By.id("country"));
        element.sendKeys(query);
        element.submit();
        TimeUnit.SECONDS.sleep( timeout: 5);
    }
    @When("click on {string}")
    public void click_on_brazil(String country) { driver.findElement(By.id(country)).click(); }
    @Then("country's name is {string} and continent is {string}")
    public void country_s_name_is_brazil(String country, String continent) throws InterruptedException {
        TimeUnit.SECONDS.sleep( timeout: 5);
        assertEquals(driver.findElement(By.id("country_name")).getText(), country);
        assertEquals(driver.findElement(By.id("country_continent")).getText(), actual: "Continent: "+continent);
    }
    @After()
    public void closeBrowser() { driver.quit(); }
}
```

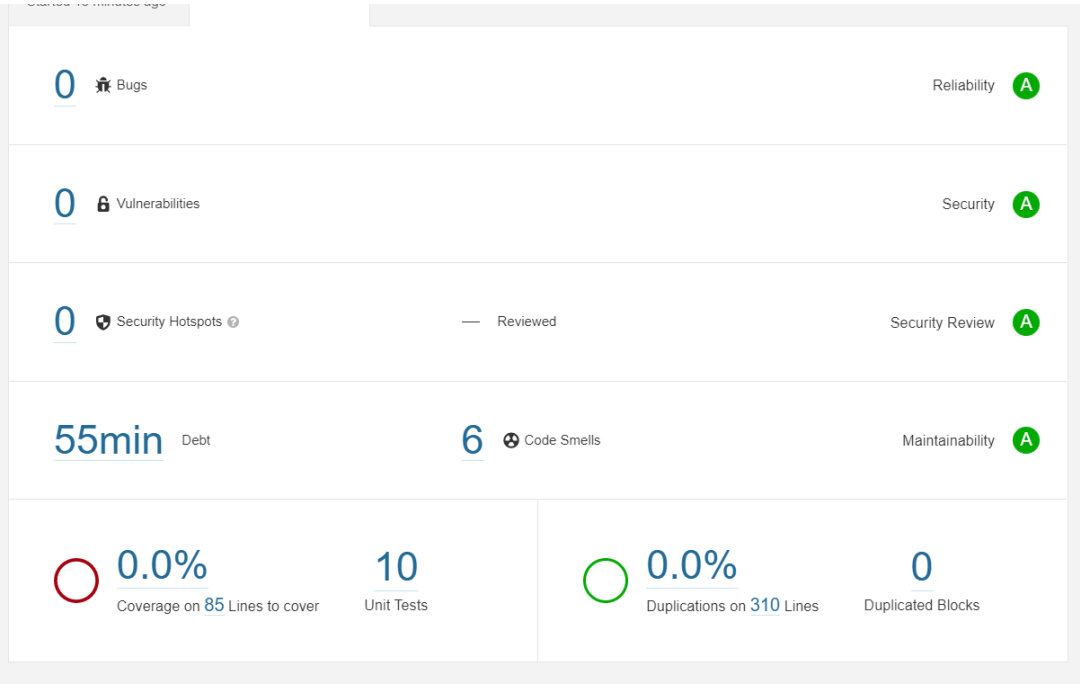
3.4 Code quality analysis

The main problem I have found with SonarQube was the security hotspot of printStackTrace() in a try-catch block. After that I have changed when I used it on code to avoid this problem.

In terms of code smell, this tool reported me the reuse of constants that could be passed through a variable. Besides that, SonarQube also reported that test classes should not have a public class declaration, just class, which I did not know, and that a good variable names pattern is just with alphanumeric digits when I used '_' to separate words.

Finally, following the instructions from this tool I have reached a code analysis with only 1 Code Smells, which are RunCucumberTest class not having any test and package naming, which I tried to change but the application was trying to test with both new and old package name's and were failing for the old one. I would not consider the first as a code smell because this file exists how Cucumber will work.

After changing all problems reported by SonarQube I have reached the following results:



I just do not know why the coverage was 0% because I have tested all method from all files, so this number should be much higher than that.

4 References & resources

Project resources

Resource:	URL/location:
Git repository	https://github.com/TheoMagno/TQS-96145
Video demo	https://github.com/TheoMagno/TQS-96145/blob/main/HW1/ApplicationDemo.mkv

Reference materials

Used API: https://rapidapi.com/api-sports/api/covid-193/