universidade de aveiro
deti departamento de eletrónica, telecomunicações e informática

HW1: Mid-term assignment report

Camila Franco de Sá Fonseca [97880], v2022-04-24

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 application allows users to obtain data on Covid incidence both referring to individual countries and worldwide through a web application and a REST API.

1.2 Current limitations

- Only worldwide data has backup API support;
- Only individual countries are filterable on;
- All cache data is lost when restarting the API;
- If backup api were to be used, it wouldn’t be compatible with current filtering methods - its contents can be displayed (Any API can be hooked up to the existing system in fact) but not filtered on.

2 Product specification

2.1 Functional scope and supported interactions

Other web applications may use the API as a data source. Interested users may use the web application to access the data in a more user-friendly manner. The API admin might use the web application to easily check the API statistics (Cache hits/misses, total request number)

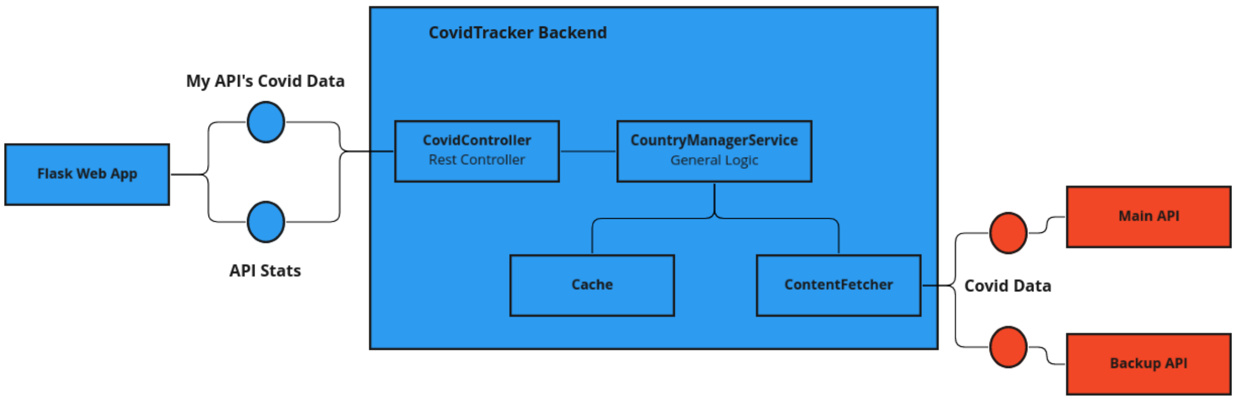
2.2 System architecture

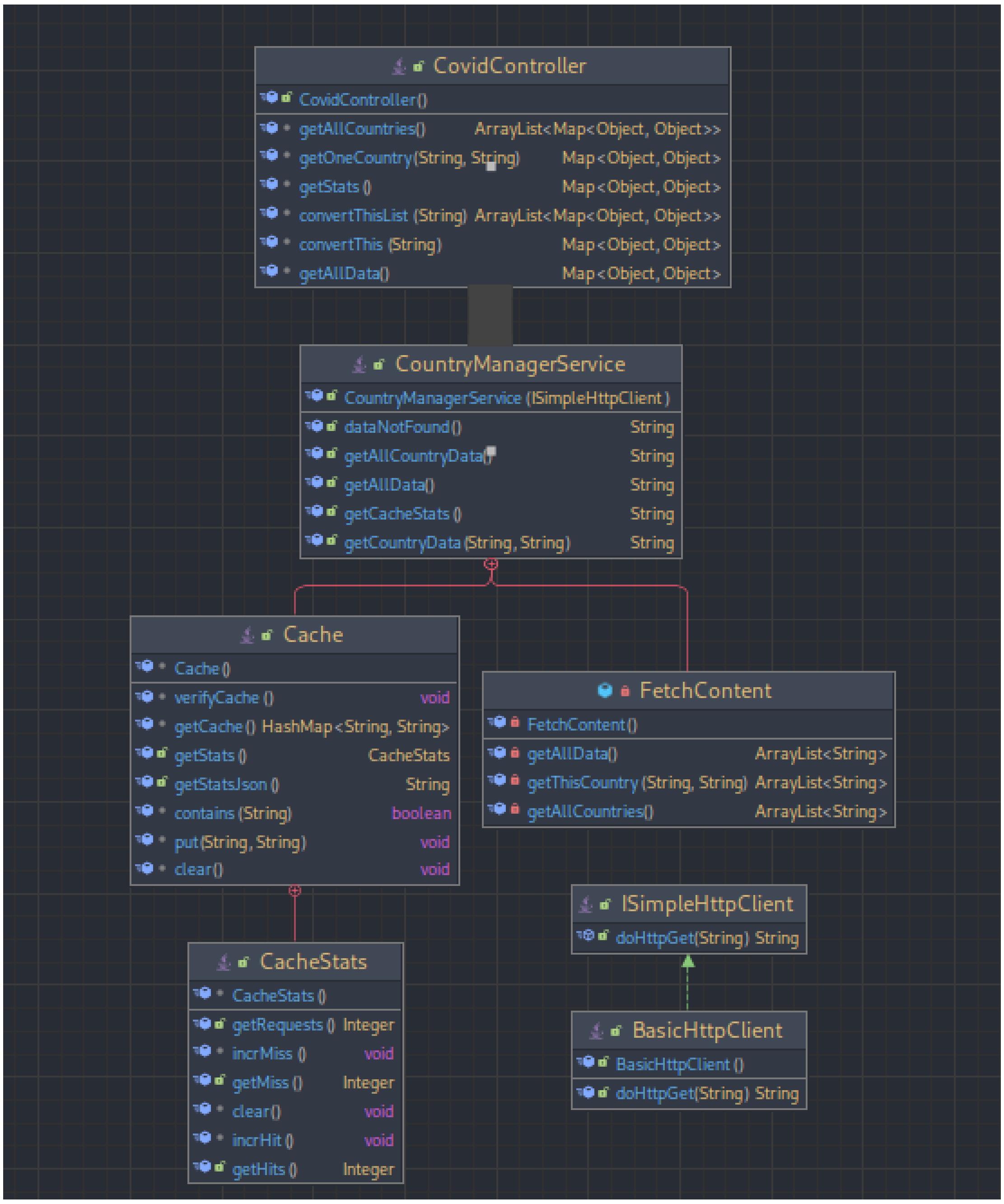
Frontend - Flask (Python)

- Multiple (2) sources can be switched between, when the first is down.

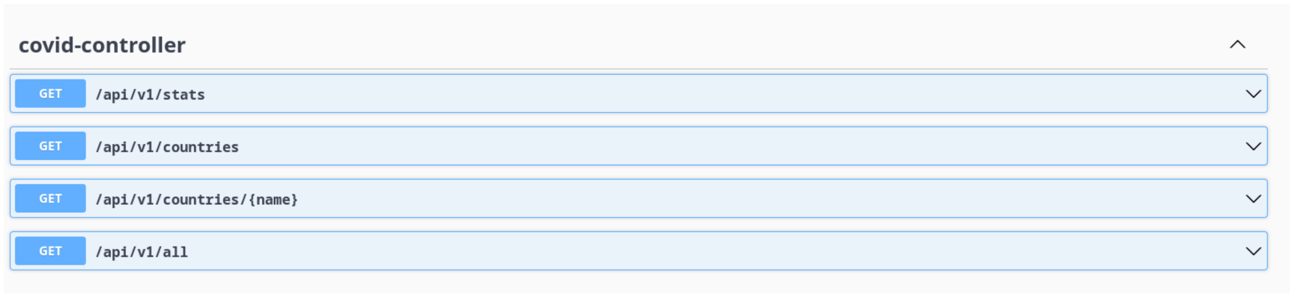
Backend - Spring Boot (Java)

- Tests using Selenium, Junit and Cucumber.




CovidController
CovidController()
getAllCountries() ArrayList<Map<Object, Object>>
getOneCountry(String, String) Map<Object, Object>
getStats() Map<Object, Object>
convertThisList(String) ArrayList<Map<Object, Object>>
convertThis(String) Map<Object, Object>
getAllData() Map<Object, Object>
CountryManagerService
CountryManagerService(ISimpleHttpClient)
dataNotFound() String
getAllCountryData() String
getAllData() String
getCacheStats() String
getCountryData(String, String) String
Cache
Cache()
verifyCache() void
getCache() HashMap<String, String>
getStats() CacheStats
getStatsJson() String
contains(String) boolean
put(String, String) void
clear() void
FetchContent
FetchContent()
getAllData() ArrayList<String>
getThisCountry(String, String) ArrayList<String>
getAllCountries() ArrayList<String>
ISimpleHttpClient
doHttpGet(String) String
CacheStats
CacheStats()
getRequests() Integer
incrMiss() void
getMiss() Integer
clear() void
incrHit() void
getHits() Integer
BasicHttpClient
BasicHttpClient()
doHttpGet(String) String

2.3 API for developers



covid-controller

GET `/api/v1/stats`

GET `/api/v1/countries`

GET `/api/v1/countries/{name}`

GET `/api/v1/all`

stats - Statistics relative to the Cache

countries - Data for all countries (by country)

all - Worldwide data (aggregated)

countries/<name> - Data for a specific country

3 Quality assurance

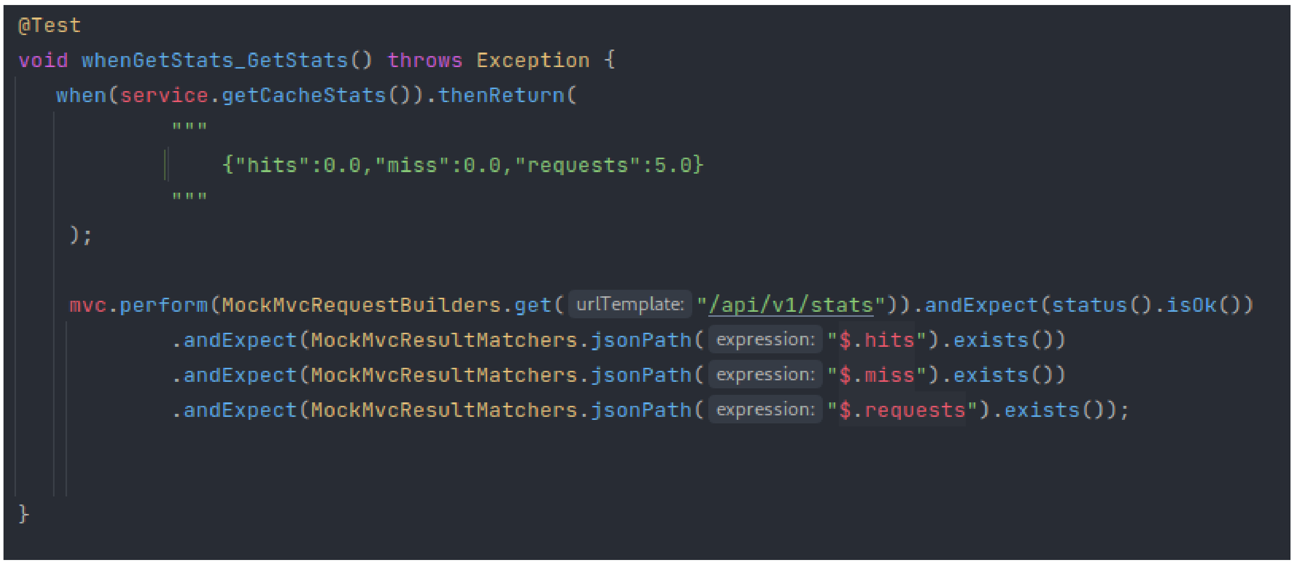
3.1 Overall strategy for testing

TDD was mostly used, many unit and integration tests were written before anything was implemented. However, changes to the implementation plans happened during which forced rewritting a large part of these tests. Cucumber/Selenium-based tests were obviously written after everything else was done.
Junit and Mockito were used to write the unit and integration tests.

3.2 Unit and integration testing

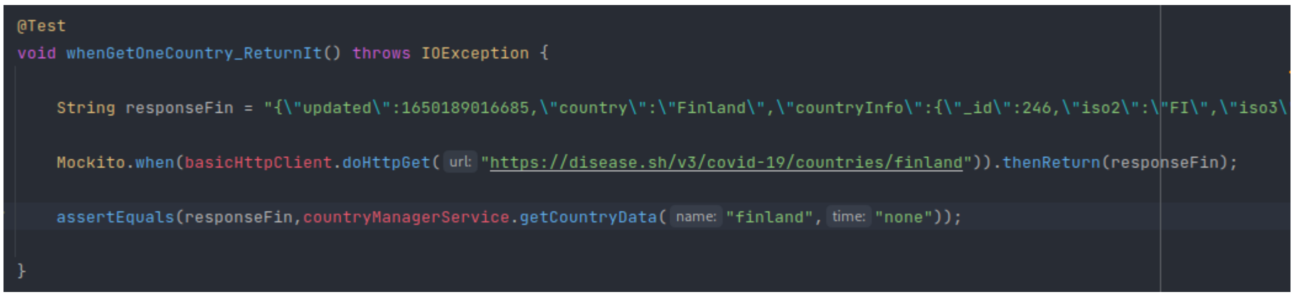
(Some examples were included)

Unit Tests were used to test the Controller and the Service (Which included tests on the cache).
Controller - simple tests to verify the correct functioning of a couple of endpoints, mocking the Service with a MockMVC.

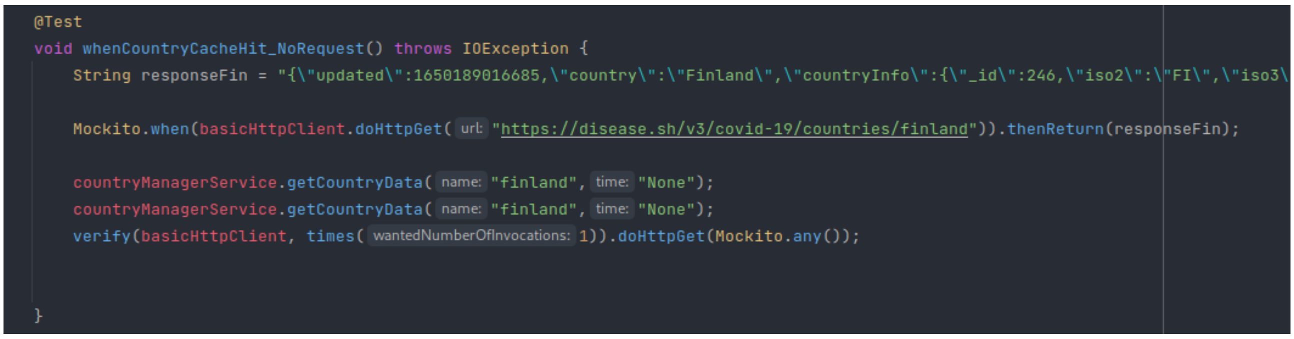


```java
@Test
void whenGetStats_GetStats() throws Exception {
   when(service.getCacheStats()).thenReturn(
            """
               {"hits":0.0,"miss":0.0,"requests":5.0}
            """
    );

    mvc.perform(MockMvcRequestBuilders.get( urlTemplate: "/api/v1/stats")).andExpect(status().isOk())
            .andExpect(MockMvcResultMatchers.jsonPath( expression: "$.hits").exists())
            .andExpect(MockMvcResultMatchers.jsonPath( expression: "$.miss").exists())
            .andExpect(MockMvcResultMatchers.jsonPath( expression: "$.requests").exists());


}
```

Service - simple tests to verify the correct functions were being called on the Fetcher, mocking the http client using mockito. Also verified that error messages were being returned correctly, and that when a cache hit was obtained, no calls to the http client were made.

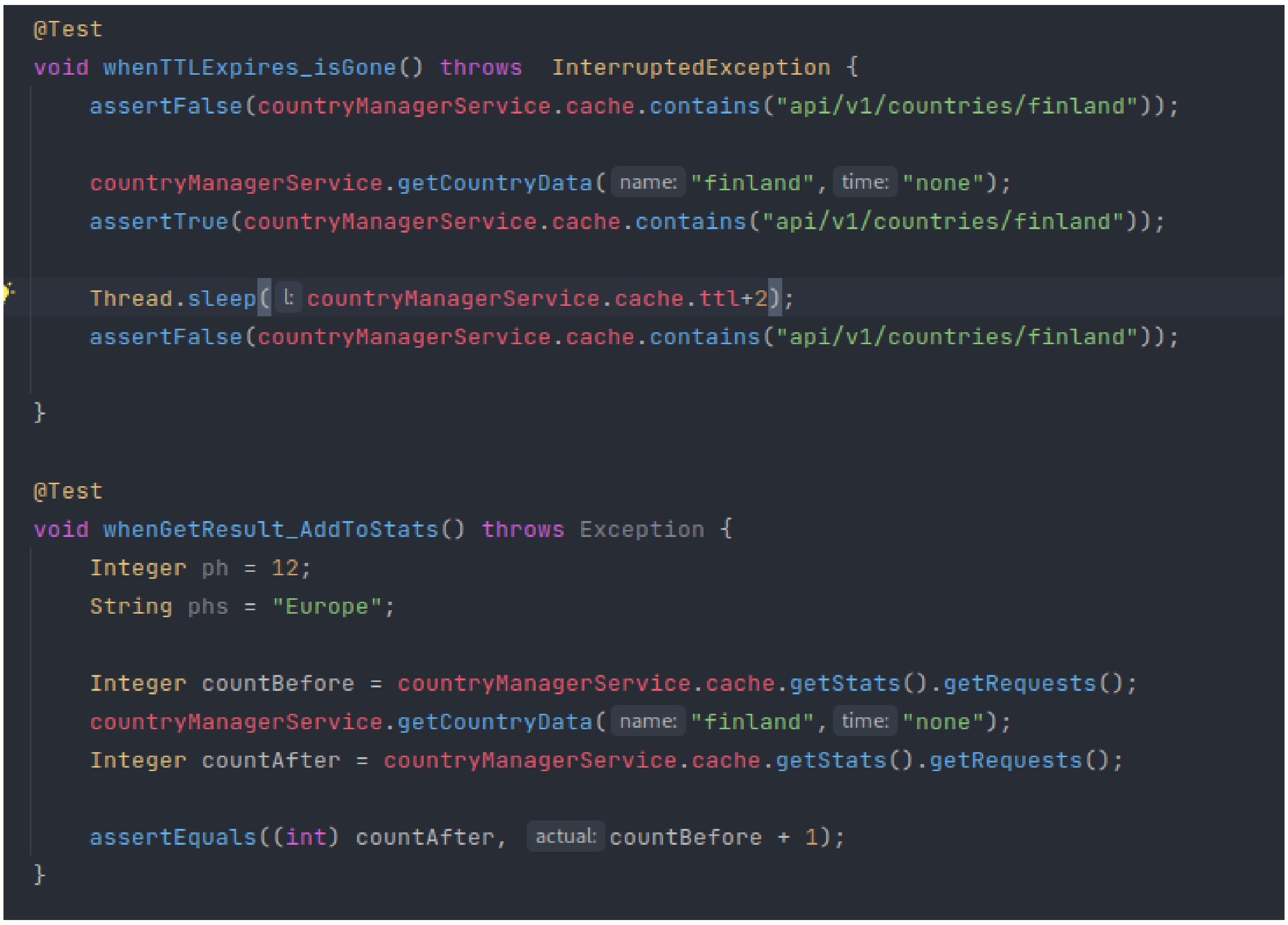


```java
@Test
void whenGetOneCountry_ReturnIt() throws IOException {

    String responseFin = "{\"updated\":1650189016685,\"country\":\"Finland\",\"countryInfo\":{\"_id\":246,\"iso2\":\"FI\",\"iso3\

    Mockito.when(basicHttpClient.doHttpGet( url: "https://disease.sh/v3/covid-19/countries/finland")).thenReturn(responseFin);

    assertEquals(responseFin,countryManagerService.getCountryData( name: "finland", time: "none"));

}
```



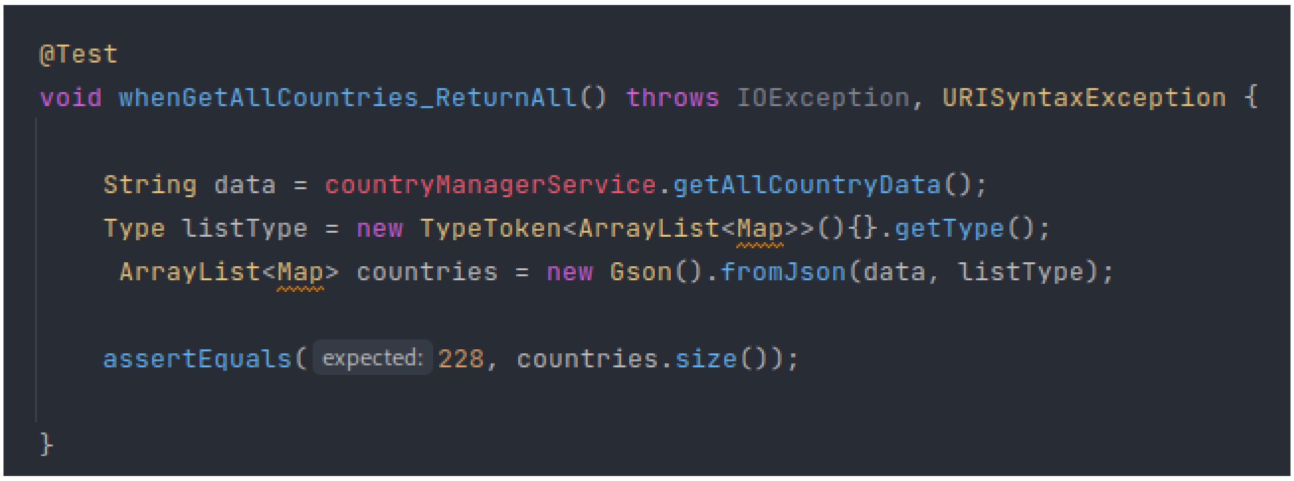
```java
@Test
void whenCountryCacheHit_NoRequest() throws IOException {
    String responseFin = "{\"updated\":1650189016685,\"country\":\"Finland\",\"countryInfo\":{\"_id\":246,\"iso2\":\"FI\",\"iso3\

    Mockito.when(basicHttpClient.doHttpGet( url: "https://disease.sh/v3/covid-19/countries/finland")).thenReturn(responseFin);

    countryManagerService.getCountryData( name: "finland", time: "None");
    countryManagerService.getCountryData( name: "finland", time: "None");
    verify(basicHttpClient, times( wantedNumberOfInvocations: 1)).doHttpGet(Mockito.any());


}
```

Cache - simple tests to verify the correct functioning of the cache, mocking the Service. Both the actual cache functioning and the stats recording were tested.



```java
@Test
void whenTTLExpires_isGone() throws  InterruptedException {
    assertFalse(countryManagerService.cache.contains("api/v1/countries/finland"));

    countryManagerService.getCountryData( name: "finland", time: "none");
    assertTrue(countryManagerService.cache.contains("api/v1/countries/finland"));

    Thread.sleep( l: countryManagerService.cache.ttl+2);
    assertFalse(countryManagerService.cache.contains("api/v1/countries/finland"));

}

@Test
void whenGetResult_AddToStats() throws Exception {
    Integer ph = 12;
    String phs = "Europe";

    Integer countBefore = countryManagerService.cache.getStats().getRequests();
    countryManagerService.getCountryData( name: "finland", time: "none");
    Integer countAfter = countryManagerService.cache.getStats().getRequests();

    assertEquals((int) countAfter,  actual: countBefore + 1);
}
```

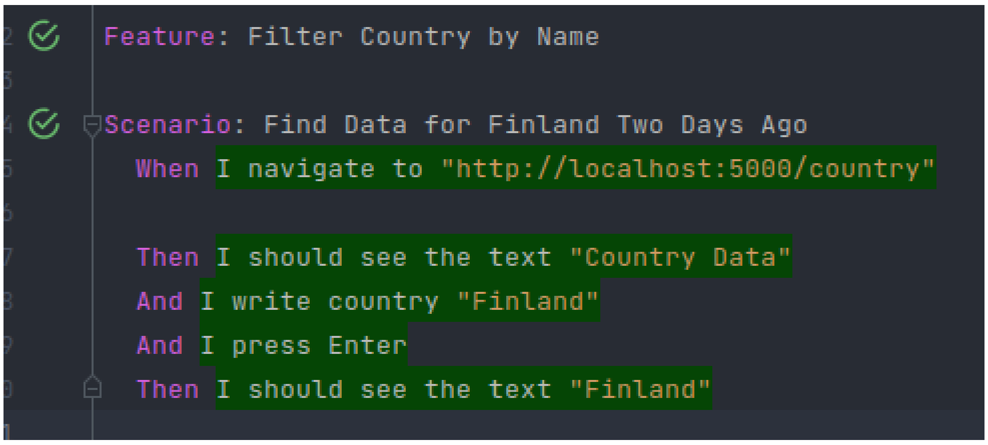
Integration Tests were used to test the service and ensure the results fetched from the actual API were correct and functioning, using simple Junit functions.



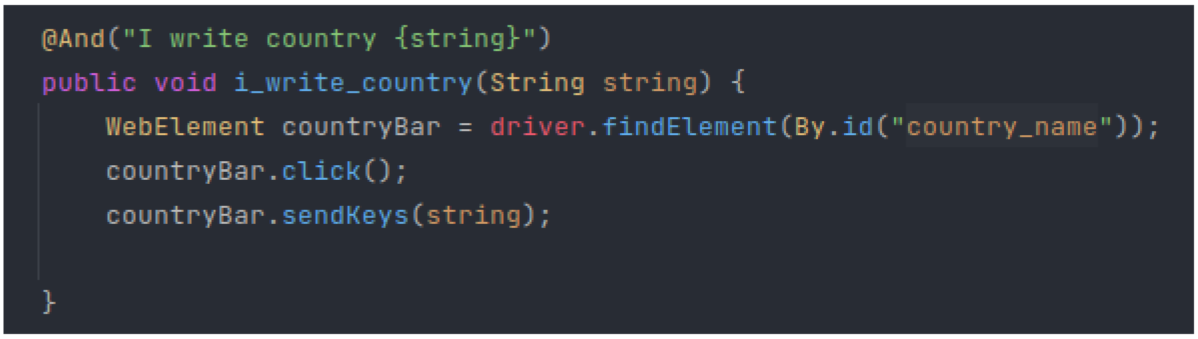
```java
@Test
void whenGetAllCountries_ReturnAll() throws IOException, URISyntaxException {

    String data = countryManagerService.getAllCountryData();
    Type listType = new TypeToken<ArrayList<Map>>(){}.getType();
     ArrayList<Map> countries = new Gson().fromJson(data, listType);

    assertEquals( expected: 228, countries.size());

}
```

3.3 Functional testing

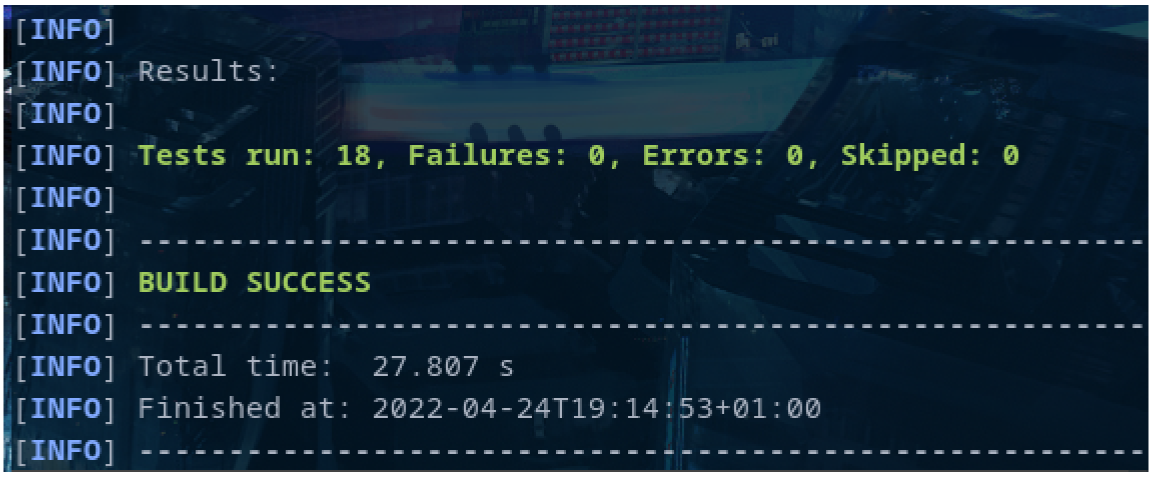
A simple use case was tested using Mockito and Cucumber (A feature implemented with Selenium's WebDriver tools and Junit). The feature considered the most common use case, to search for a Country to see its data.



```
Feature: Filter Country by Name

Scenario: Find Data for Finland Two Days Ago
  When I navigate to "http://localhost:5000/country"

  Then I should see the text "Country Data"
  And I write country "Finland"
  And I press Enter
  Then I should see the text "Finland"
```



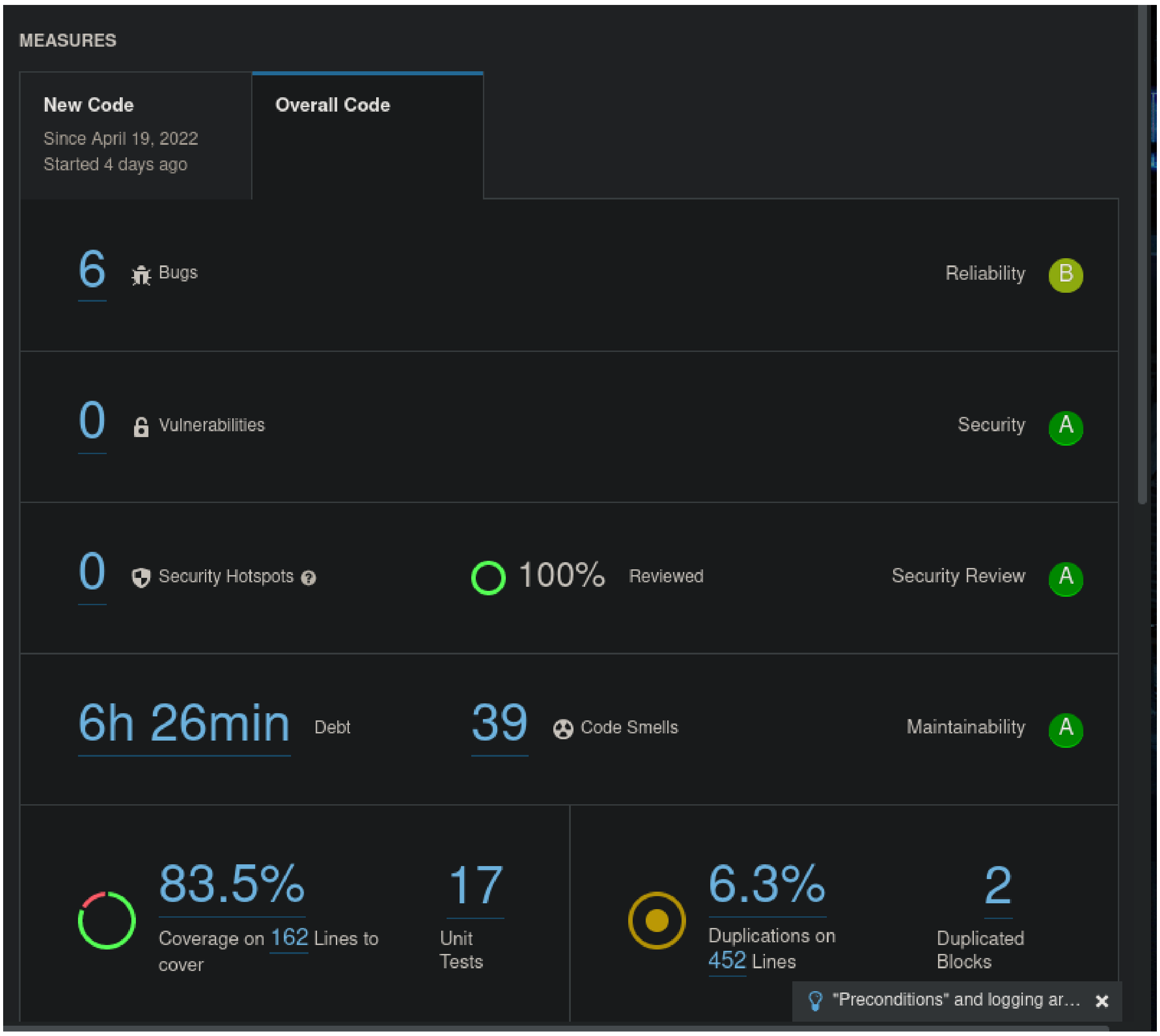
```
@And("I write country {string}")
public void i_write_country(String string) {
    WebElement countryBar = driver.findElement(By.id("country_name"));
    countryBar.click();
    countryBar.sendKeys(string);

}
```



```
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 18, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  27.807 s
[INFO] Finished at: 2022-04-24T19:14:53+01:00
[INFO] ------------------------------------------------------------------------
```

3.4 Code quality analysis

SonarQube was used for static code analysis.



MEASURES

New Code
Since April 19, 2022
Started 4 days ago

Overall Code

6 Bugs — Reliability B

0 Vulnerabilities — Security A

0 Security Hotspots — 100% Reviewed — Security Review A

6h 26min Debt — 39 Code Smells — Maintainability A

83.5% Coverage on 162 Lines to cover — 17 Unit Tests

6.3% Duplications on 452 Lines — 2 Duplicated Blocks

"Preconditions" and logging ar…

The bugs caught by sonarqube were minor and deemed safe, as the major problem with them were a obscure java notation. A couple bugs were fixed due to the analysis, that probably wouldn't have been caught otherwise.

Many code smells were addressed but some were found that were in direct conflict with the IDE's instructions (IntelliJ). The non addressed code smells were deemed non-critical. All of them wouldn't be addressed without the tool.

3.5 Continuous integration pipeline [optional]

Not applicable.

4 References & resources

Project resources

Resource:	**URL/location:**
Git repository	https://github.com/lnryatt/TQS_97880/tree/main/hw
Video demo	https://youtu.be/4tdZb72I9ss

Reference materials

Backend:
API Documentation - https://springdoc.org/#Introduction
Covid API - https://disease.sh/docs
REST Api - https://spring.io/guides/tutorials/rest/
Spring Docs -
https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/http/RequestEntity.html
Tests:
Many many many Stack Overflow pages to solve weird spring exceptions.
Setting up cucumber tests without the skeleton -
https://www.jetbrains.com/help/idea/running-cucumber-tests.html#cucumber-new-run-config
Cucumber docs - https://cucumber.io/docs/cucumber/api/#running-cucumber
JsonPath sandbox - https://jsonpath.com/

Frontend:
Flask docs - https://flask.palletsprojects.com/en/2.1.x/quickstart/#url-building
Handy flask tutorial -
https://www.digitalocean.com/community/tutorials/how-to-create-your-first-web-application-using-flask-and-python-3
Used bootstrap (very nice) - http://kristopolous.github.io/BOOTSTRA.386/demo.html