HW1: Mid-term assignment report

Miguel Rocha Ferreira [98599], v2022-04-30

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 consisted of creating a platform to check Covid-19 Statistics and History Data (cases, tests and deaths) for different countries around the globe. This platform is based on a REST-API service integrated with a Web application and tests to verify the integrity of the system. The project also contains a cache that prevents too many accesses to the external API, storing the most recent values for a limited time.

1.2 Current limitations

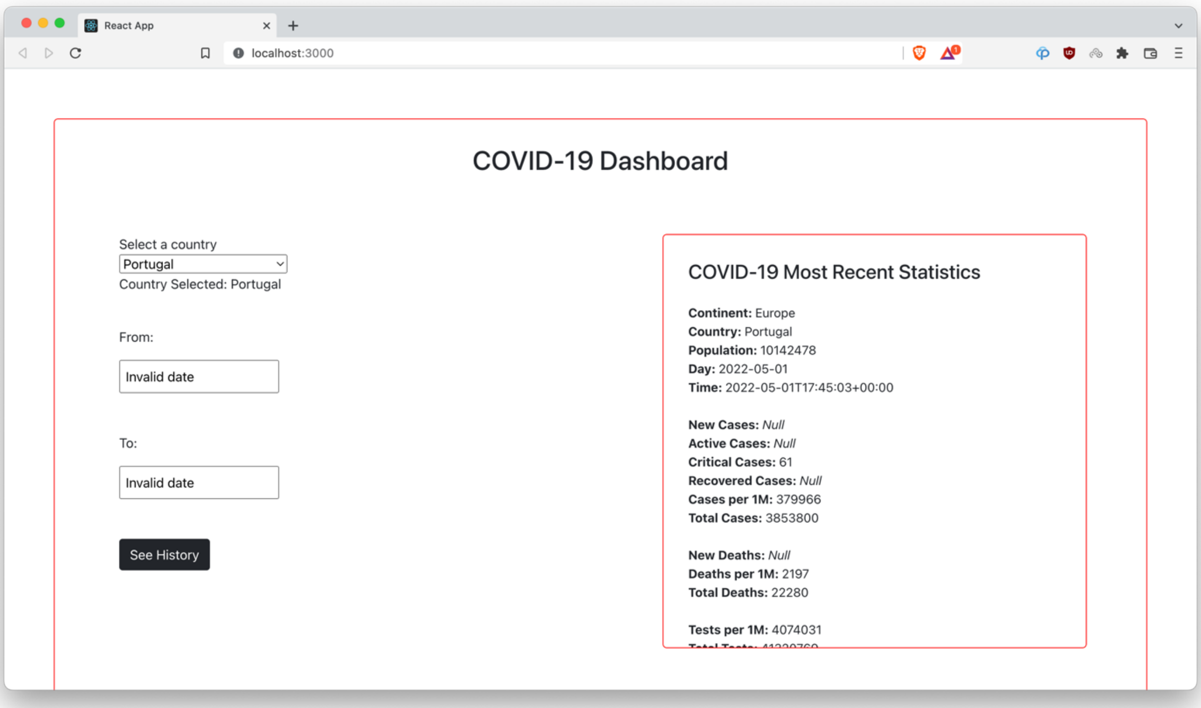
Given the fact that the pandemic is becoming a less recurrent subject, some countries have stopped to track or publish their Covid-19 data. With that, in most countries, there are many fields with *null* values.
One other limitation is the use of only one external API, making the product very subject to failing if the external API fails.

A possible feature to implement in the feature would be gathering of statistics by continent, for example, doing an arithmetic mean of that continent's countries' data.
Another issue is the amount of code smells that exist in the project. When analyzing the project with *SonarQube* (will be shown later in this report) there were many code smells presented but due to the lack of time I wasn't able to fix them.

2 Product specification

2.1 Functional scope and supported interactions

The application can be used for everyone who wishes to get COVID-19 statistics for a specific country.
It consists of a Single-Page Web App that contains an input of the country we want to search, and two date inputs, to define the range of the data we want.
To run it, we just need to run 'docker-compose build' and then 'docker-compose up'.
If we only want to check the most recent statistics of a specific country, we only need to choose a country from the list. It's not necessary to fill the dates.
If we want statistics for a specific day, we need to set the country to the country we want and the *From* date and the *To* date to the day we want to get statistics for.
If we want statistics for a range of days, it's the same as above, but we specify the beginning date and the end date on the *From* and *To* input fields, respectively.



COVID-19 Dashboard

Select a country
Portugal
Country Selected: Portugal

From:
Invalid date

To:
Invalid date

COVID-19 Most Recent Statistics

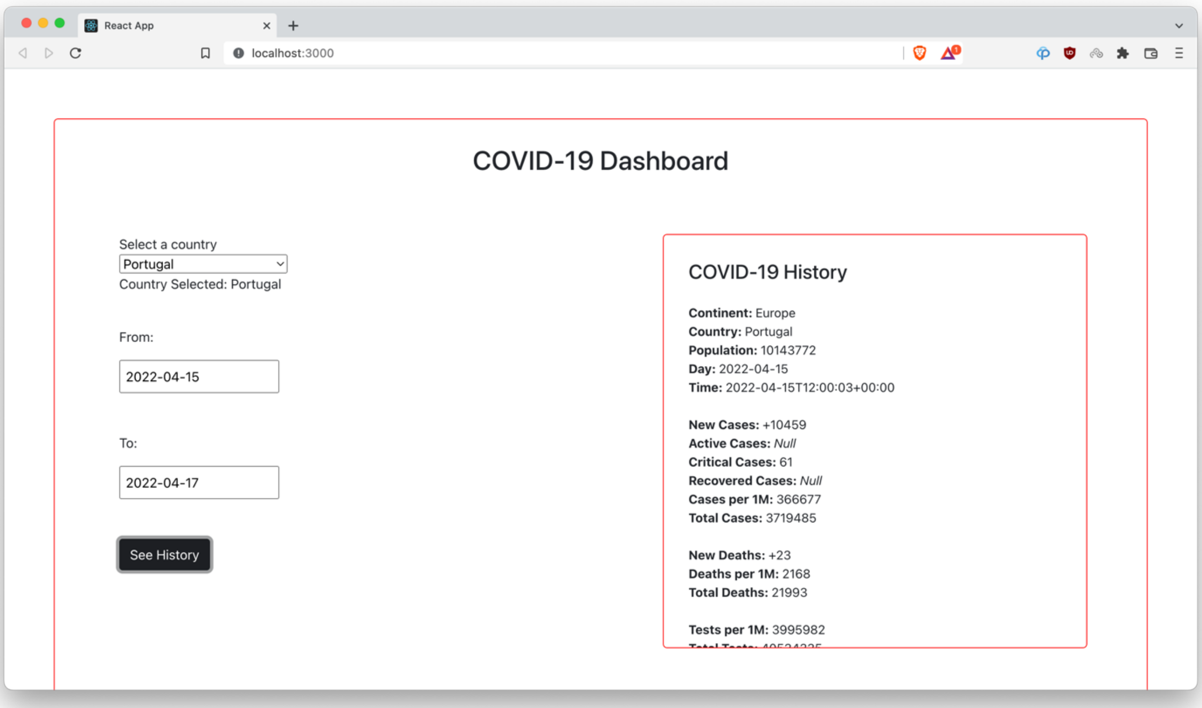
Continent: Europe
Country: Portugal
Population: 10142478
Day: 2022-05-01
Time: 2022-05-01T17:45:03+00:00

New Cases: *Null*
Active Cases: *Null*
Critical Cases: 61
Recovered Cases: *Null*
Cases per 1M: 379966
Total Cases: 3853800

New Deaths: *Null*
Deaths per 1M: 2197
Total Deaths: 22280

Tests per 1M: 4074031

Figure 1: App showing statistics of Portugal



COVID-19 Dashboard

Select a country
Portugal
Country Selected: Portugal

From:
2022-04-15

To:
2022-04-17

See History

COVID-19 History

Continent: Europe
Country: Portugal
Population: 10143772
Day: 2022-04-15
Time: 2022-04-15T12:00:03+00:00

New Cases: +10459
Active Cases: *Null*
Critical Cases: 61
Recovered Cases: *Null*
Cases per 1M: 366677
Total Cases: 3719485

New Deaths: +23
Deaths per 1M: 2168
Total Deaths: 21993

Tests per 1M: 3995982

Figure 2: App showing history results of Portugal

2.2 System architecture

The JavaScript framework chosen for the WebApp was React. For the backend, it was used the Spring Framework module Spring Boot.

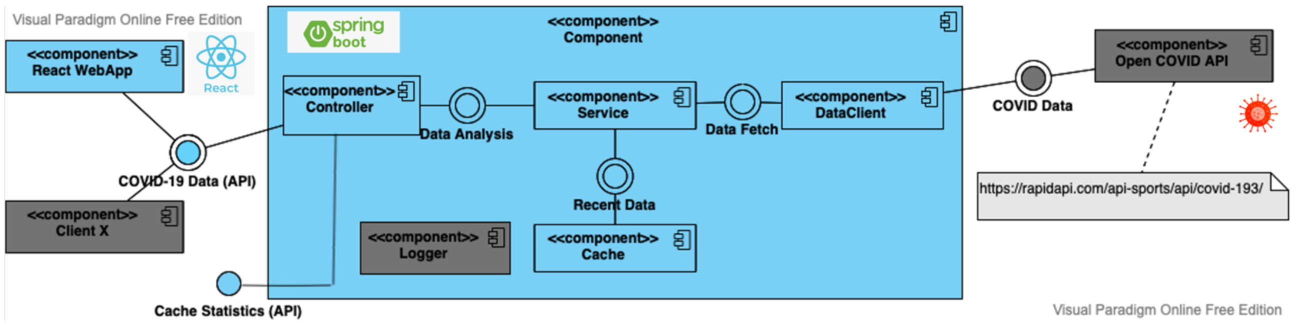
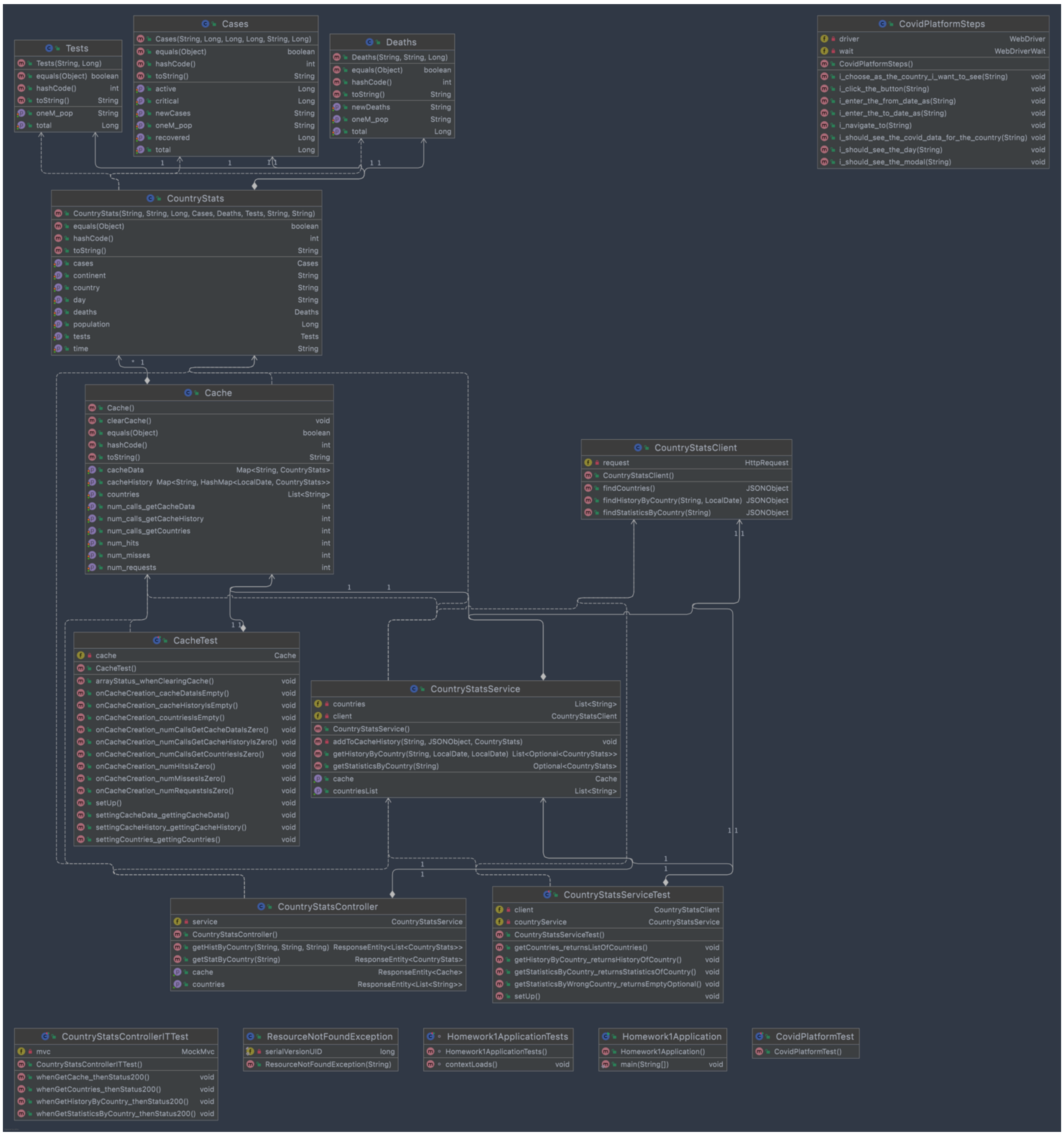


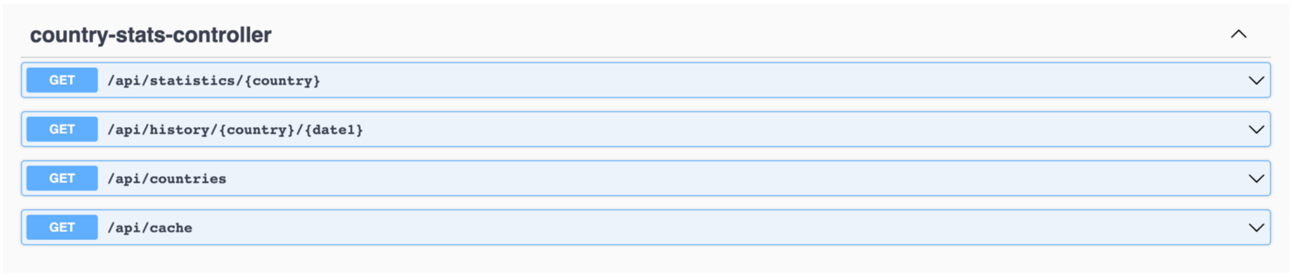
Figure 3: System Architecture


Tests
Tests(String, Long)
equals(Object) boolean
hashCode() int
toString() String
oneM_pop String
total Long
Cases
Cases(String, Long, Long, Long, String, Long)
equals(Object) boolean
hashCode() int
toString() String
active Long
critical Long
newCases String
oneM_pop String
recovered Long
total Long
Deaths
Deaths(String, String, Long)
equals(Object) boolean
hashCode() int
toString() String
newDeaths String
oneM_pop String
total Long
CovidPlatformSteps
driver WebDriver
wait WebDriverWait
CovidPlatformSteps()
i_choose_as_the_country_i_want_to_see(String) void
i_click_the_button(String) void
i_enter_the_from_date_as(String) void
i_enter_the_to_date_as(String) void
i_navigate_to(String) void
i_should_see_the_covid_data_for_the_country(String) void
i_should_see_the_day(String) void
i_should_see_the_modal(String) void
CountryStats
CountryStats(String, String, Long, Cases, Deaths, Tests, String, String)
equals(Object) boolean
hashCode() int
toString() String
cases Cases
continent String
country String
day String
deaths Deaths
population Long
tests Tests
time String
Cache
Cache()
clearCache() void
equals(Object) boolean
hashCode() int
toString() String
cacheData Map<String, CountryStats>
cacheHistory Map<String, HashMap<LocalDate, CountryStats>>
countries List<String>
num_calls_getCacheData int
num_calls_getCacheHistory int
num_calls_getCountries int
num_hits int
num_misses int
num_requests int
CountryStatsClient
request HttpRequest
CountryStatsClient()
findCountries() JSONObject
findHistoryByCountry(String, LocalDate) JSONObject
findStatisticsByCountry(String) JSONObject
CacheTest
cache Cache
CacheTest()
arrayStatus_whenClearingCache() void
onCacheCreation_cacheDataIsEmpty() void
onCacheCreation_cacheHistoryIsEmpty() void
onCacheCreation_countriesIsEmpty() void
onCacheCreation_numCallsGetCacheDataIsZero() void
onCacheCreation_numCallsGetCacheHistoryIsZero() void
onCacheCreation_numCallsGetCountriesIsZero() void
onCacheCreation_numHitsIsZero() void
onCacheCreation_numMissesIsZero() void
onCacheCreation_numRequestsIsZero() void
setUp() void
settingCacheData_gettingCacheData() void
settingCacheHistory_gettingCacheHistory() void
settingCountries_gettingCountries() void
CountryStatsService
countries List<String>
client CountryStatsClient
CountryStatsService()
addToCacheHistory(String, JSONObject, CountryStats) void
getHistoryByCountry(String, LocalDate, LocalDate) List<Optional<CountryStats>>
getStatisticsByCountry(String) Optional<CountryStats>
cache Cache
countriesList List<String>
CountryStatsController
service CountryStatsService
CountryStatsController()
getHistByCountry(String, String, String) ResponseEntity<List<CountryStats>>
getStatByCountry(String) ResponseEntity<CountryStats>
cache ResponseEntity<Cache>
countries ResponseEntity<List<String>>
CountryStatsServiceTest
client CountryStatsClient
countryService CountryStatsService
CountryStatsServiceTest()
getCountries_returnsListOfCountries() void
getHistoryByCountry_returnsHistoryOfCountry() void
getStatisticsByCountry_returnsStatisticsOfCountry() void
getStatisticsByWrongCountry_returnsEmptyOptional() void
setUp() void
CountryStatsControllerITTest
mvc MockMvc
CountryStatsControllerITTest()
whenGetCache_thenStatus200() void
whenGetCountries_thenStatus200() void
whenGetHistoryByCountry_thenStatus200() void
whenGetStatisticsByCountry_thenStatus200() void
ResourceNotFoundException
serialVersionUID long
ResourceNotFoundException(String)
Homework1ApplicationTests
Homework1ApplicationTests()
contextLoads() void
Homework1Application
Homework1Application()
main(String[]) void
CovidPlatformTest
CovidPlatformTest()
Figure 4: Project UML Class Diagram

Because the diagram ended up being too big and the image is not very clear, I'll be leaving the image named as 'Project Class Diagram' on the Images folder, located at the root of the project.

2.3 API for developers

To generate the API documentation, it was used Swagger UI API.
To access the API documentation, we can access the following link: http://localhost:8080/swagger-ui/index.html. The Springboot App needs to be running to access the documentation.



country-stats-controller

GET /api/statistics/{country}

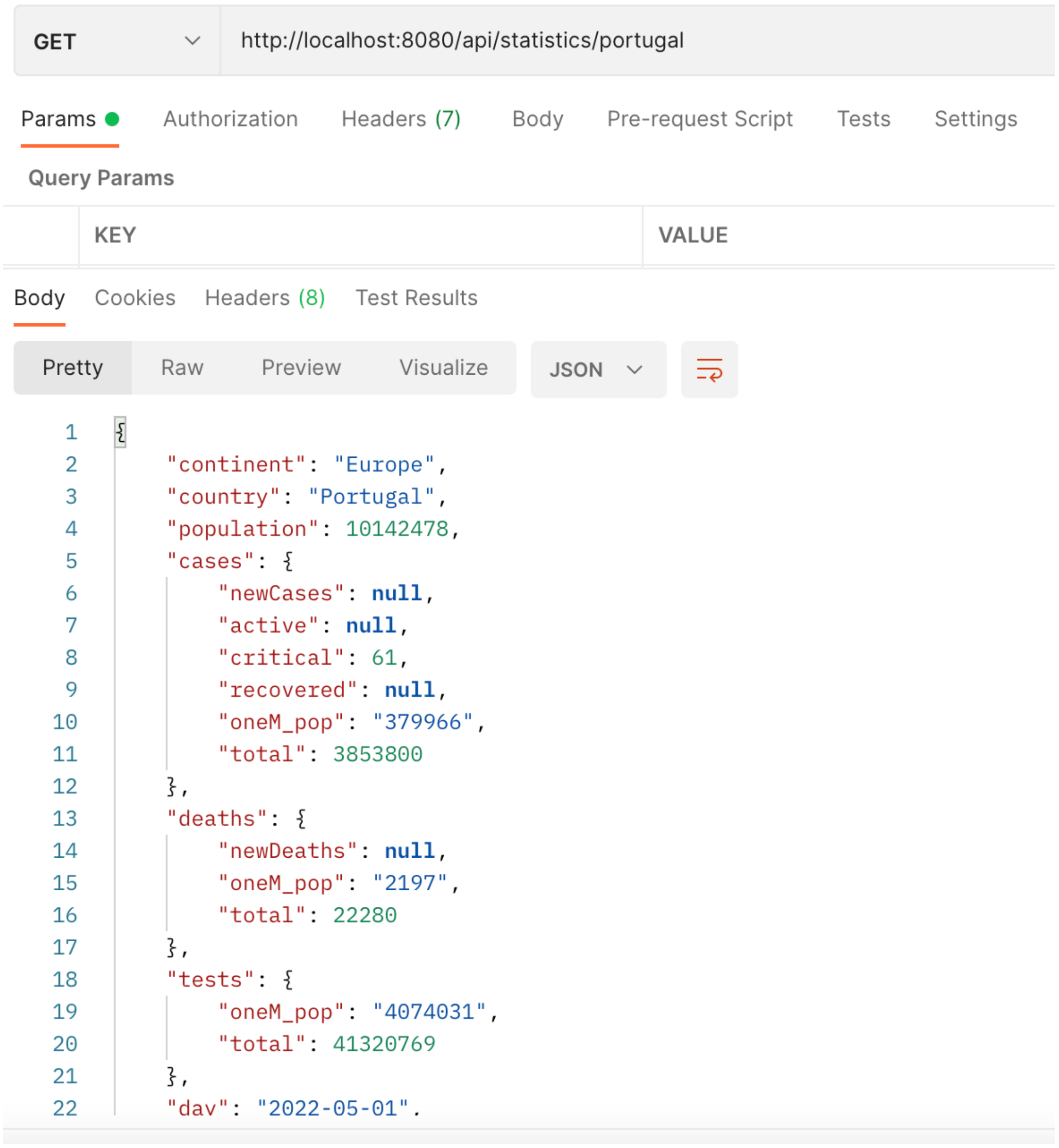
GET /api/history/{country}/{date1}

GET /api/countries

GET /api/cache

Figure 5: API Endpoints generated by SwaggerUI

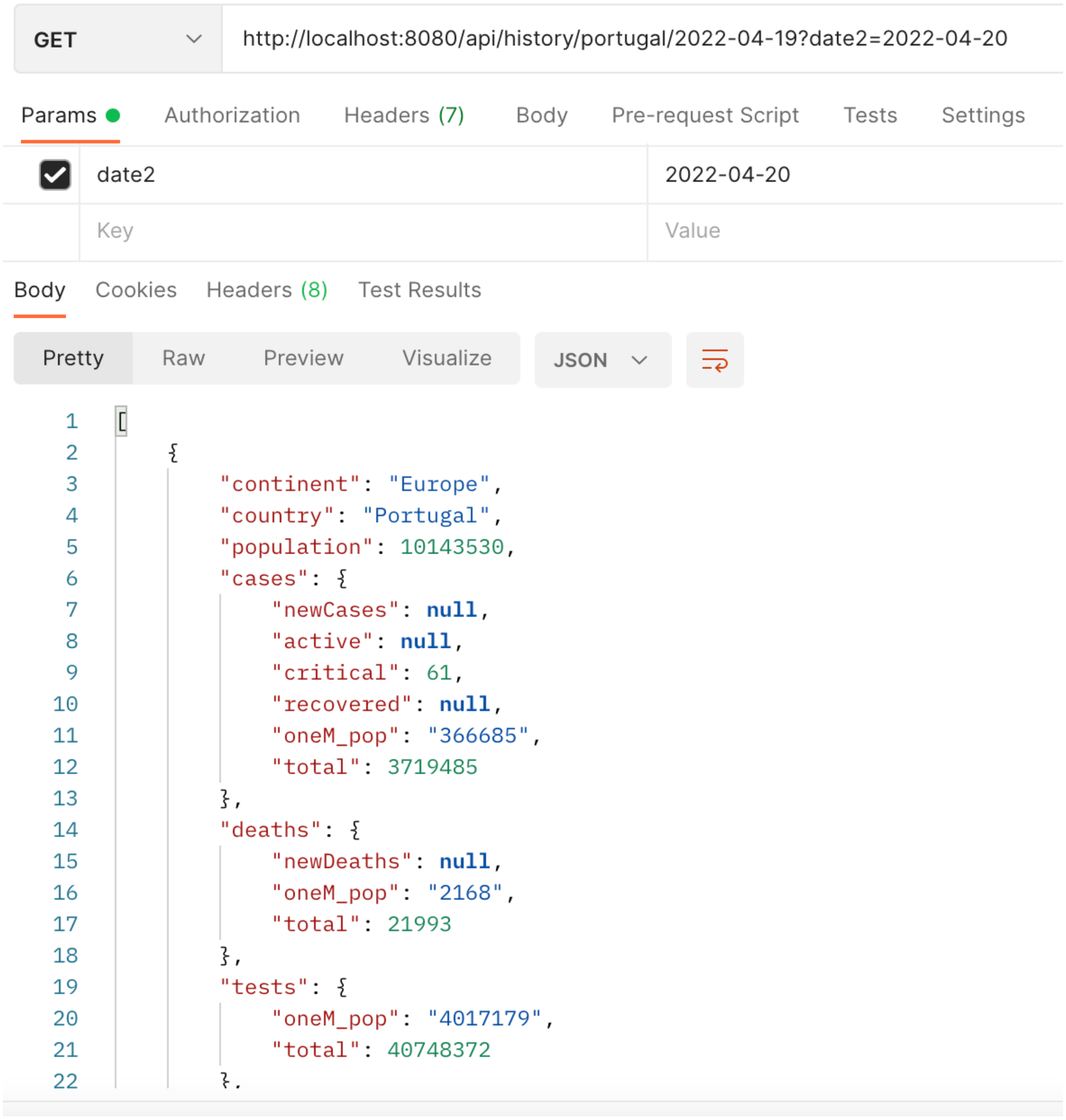
- **GET /api/statistics/{country} –** Gives us the statistics of the given country
  - Example of output:



```
{
    "continent": "Europe",
    "country": "Portugal",
    "population": 10142478,
    "cases": {
        "newCases": null,
        "active": null,
        "critical": 61,
        "recovered": null,
        "oneM_pop": "379966",
        "total": 3853800
    },
    "deaths": {
        "newDeaths": null,
        "oneM_pop": "2197",
        "total": 22280
    },
    "tests": {
        "oneM_pop": "4074031",
        "total": 41320769
    },
    "dav": "2022-05-01".
```

Figure 6: Statistics of Portugal

- **GET /api/history/{country}/date1 –** Gives us the history of the given country and the given date.
    - Example of output:



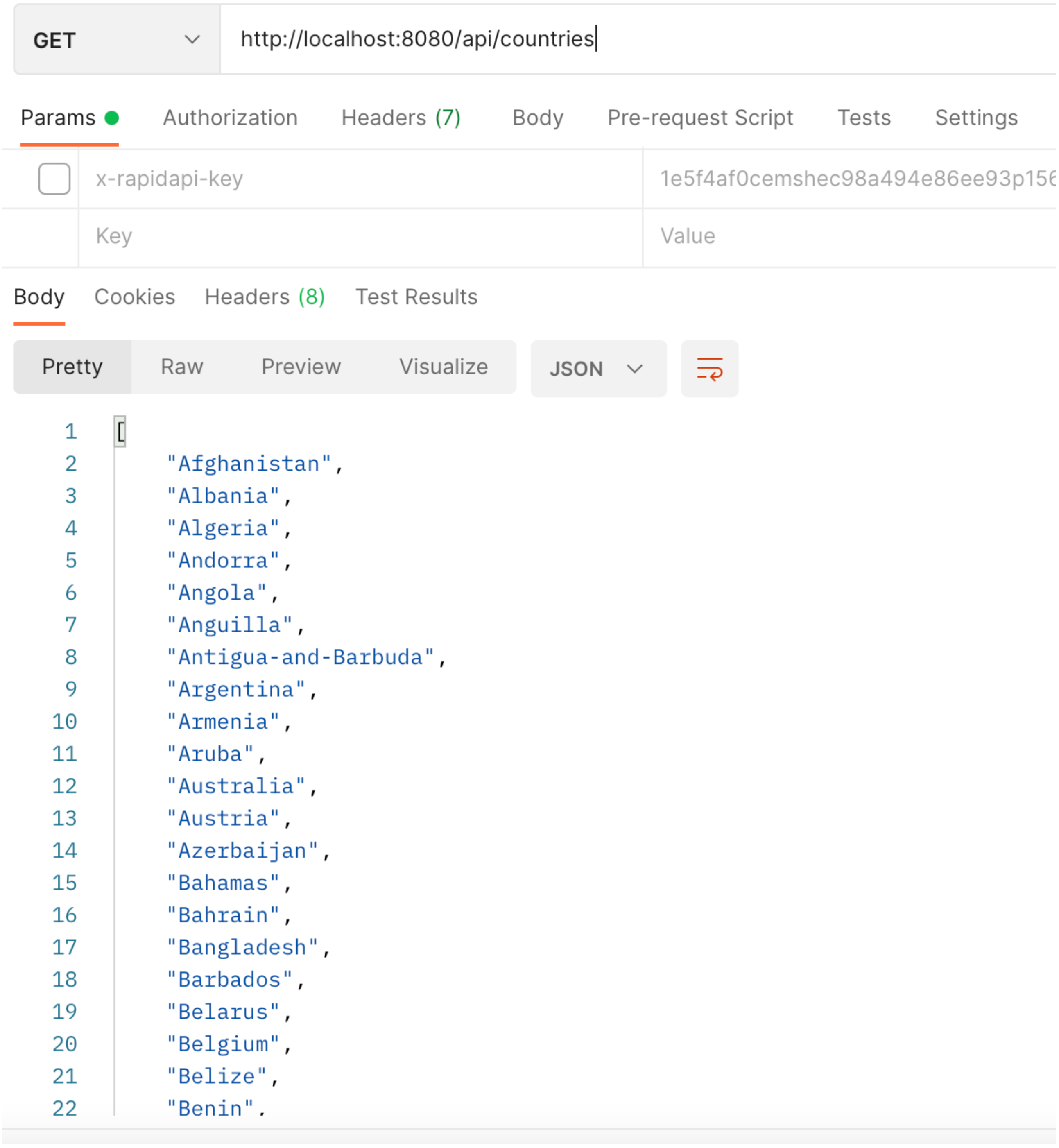
GET http://localhost:8080/api/history/portugal/2022-04-19?date2=2022-04-20

Key	Value
date2	2022-04-20

```
[
    {
        "continent": "Europe",
        "country": "Portugal",
        "population": 10143530,
        "cases": {
            "newCases": null,
            "active": null,
            "critical": 61,
            "recovered": null,
            "oneM_pop": "366685",
            "total": 3719485
        },
        "deaths": {
            "newDeaths": null,
            "oneM_pop": "2168",
            "total": 21993
        },
        "tests": {
            "oneM_pop": "4017179",
            "total": 40748372
        },
```

Figure 7: History Statistics of Portugal

- **GET /api/countries –** Gives us the list of countries
  - Example of output:



GET http://localhost:8080/api/countries

Params ● | Authorization | Headers (7) | Body | Pre-request Script | Tests | Settings

Key	Value
x-rapidapi-key	1e5f4af0cemshec98a494e86ee93p156

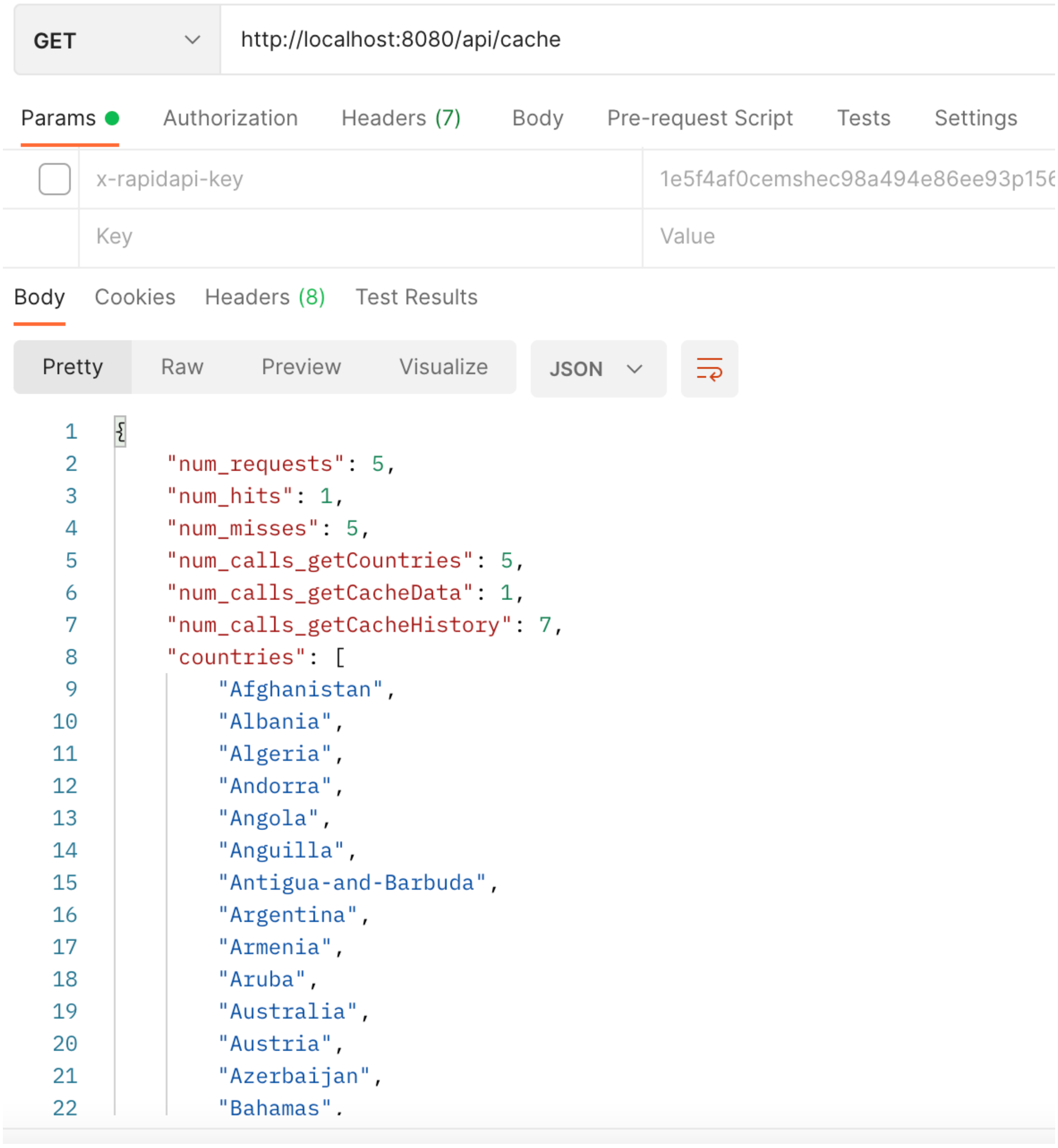
Body | Cookies | Headers (8) | Test Results

Pretty | Raw | Preview | Visualize | JSON

```
[
    "Afghanistan",
    "Albania",
    "Algeria",
    "Andorra",
    "Angola",
    "Anguilla",
    "Antigua-and-Barbuda",
    "Argentina",
    "Armenia",
    "Aruba",
    "Australia",
    "Austria",
    "Azerbaijan",
    "Bahamas",
    "Bahrain",
    "Bangladesh",
    "Barbados",
    "Belarus",
    "Belgium",
    "Belize",
    "Benin".
```

Figure 8: List of Countries

- **GET /api/cache –** Gives us the cache statistics and data structures
    - Example of output:



GET http://localhost:8080/api/cache

Params ● Authorization Headers (7) Body Pre-request Script Tests Settings

Key	Value
x-rapidapi-key	1e5f4af0cemshec98a494e86ee93p156

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "num_requests": 5,
    "num_hits": 1,
    "num_misses": 5,
    "num_calls_getCountries": 5,
    "num_calls_getCacheData": 1,
    "num_calls_getCacheHistory": 7,
    "countries": [
        "Afghanistan",
        "Albania",
        "Algeria",
        "Andorra",
        "Angola",
        "Anguilla",
        "Antigua-and-Barbuda",
        "Argentina",
        "Armenia",
        "Aruba",
        "Australia",
        "Austria",
        "Azerbaijan",
        "Bahamas",
```

Figure 9: Cache Statistics

3 Quality assurance

3.1 Overall strategy for testing

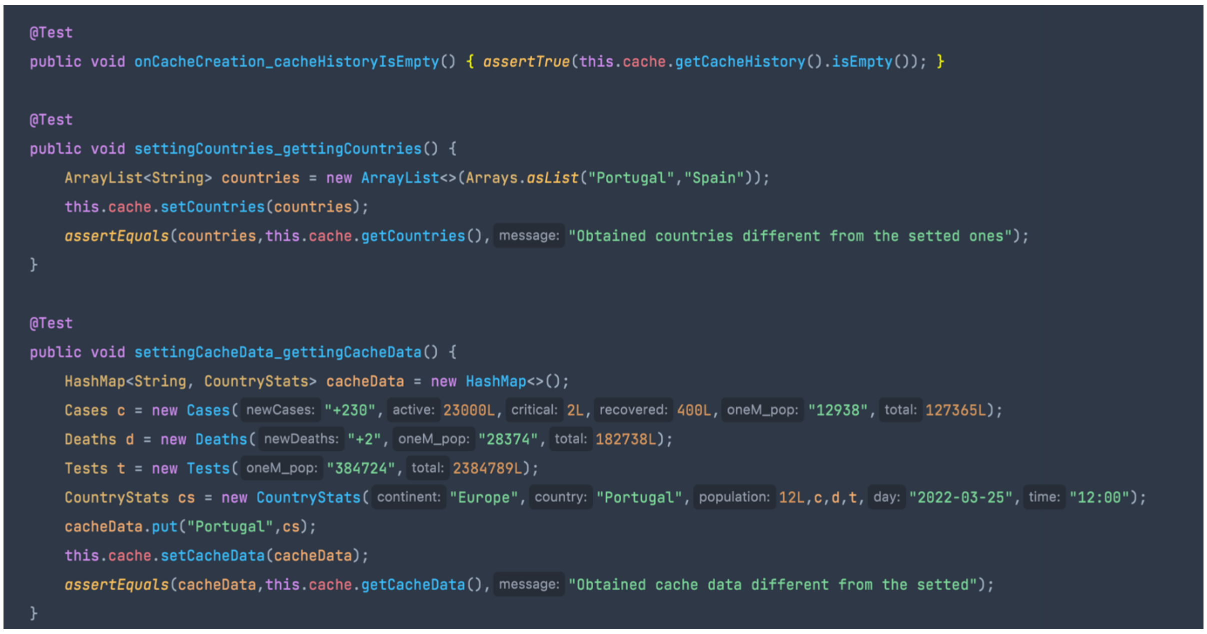
The development strategy I followed was starting with the app development (backend and frontend) and then change it according to the tests created.
This approach was adopted because I didn’t have the architecture totally defined so I decided to build incrementally and then test the features.
I also used BDD using Selenium and Cucumber.

3.2 Unit and integration testing

I used unit tests to test the Cache behavior, testing each method. I also used unit tests to test the service implementation, mocking the data received from the API. The controller was tested with integration tests. It was done using Spring MVC test support (MockMvc).

- Unit Tests
    - onCacheCreation_numRequestsIsZero()
    - onCacheCreation_numHitsIsZero()
    - onCacheCreation_numMissesIsZero()
    - onCacheCreation_numCallsGetCountriesIsZero()
    - onCacheCreation_numCallsGetCacheDataIsZero()
    - onCacheCreation_numCallsGetCacheHistoryIsZero()
    - onCacheCreation_countriesIsEmpty()
    - onCacheCreation_cacheDataIsEmpty()
    - onCacheCreation_cacheHistoryIsEmpty()
    - settingCountries_gettingCountries()
    - settingCacheData_gettingCacheData()
    - settingCacheHistory_gettingCacheHistory()
    - arrayStatus_whenClearingCache()

Example of Unit Tests:

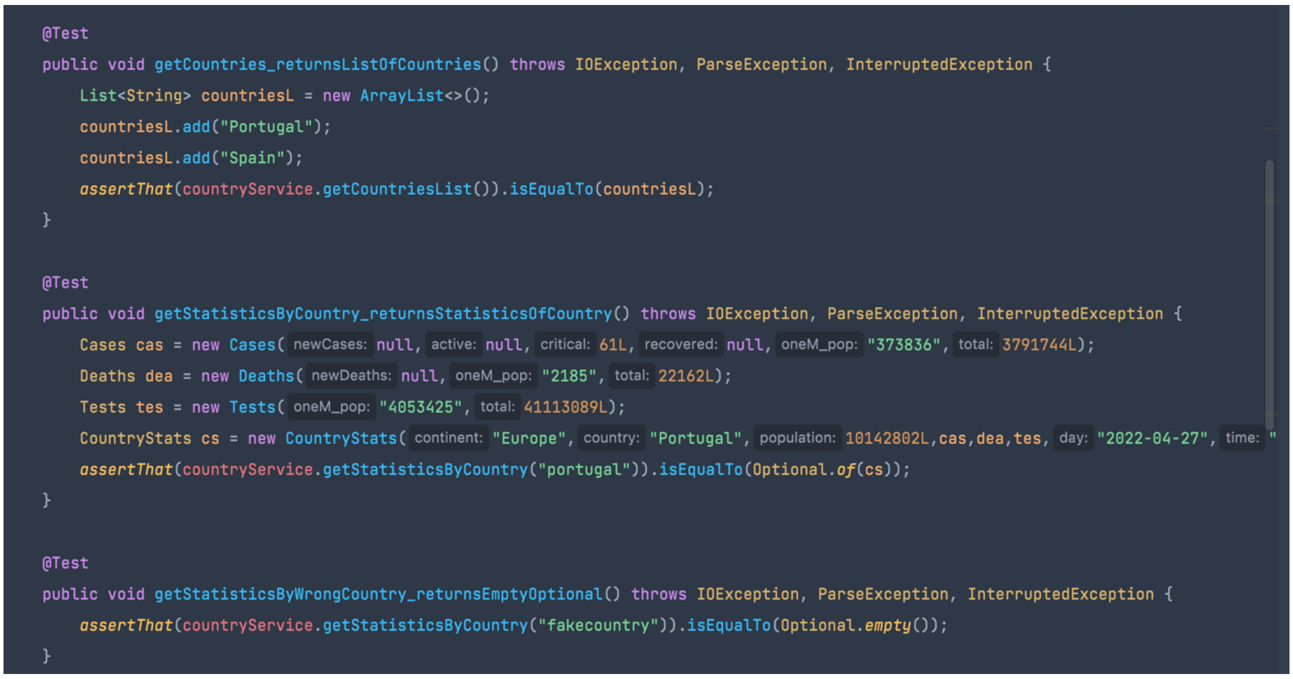


```java
@Test
public void onCacheCreation_cacheHistoryIsEmpty() { assertTrue(this.cache.getCacheHistory().isEmpty()); }

@Test
public void settingCountries_gettingCountries() {
    ArrayList<String> countries = new ArrayList<>(Arrays.asList("Portugal","Spain"));
    this.cache.setCountries(countries);
    assertEquals(countries,this.cache.getCountries(), message: "Obtained countries different from the setted ones");
}

@Test
public void settingCacheData_gettingCacheData() {
    HashMap<String, CountryStats> cacheData = new HashMap<>();
    Cases c = new Cases( newCases: "+230", active: 23000L, critical: 2L, recovered: 400L, oneM_pop: "12938", total: 127365L);
    Deaths d = new Deaths( newDeaths: "+2", oneM_pop: "28374", total: 182738L);
    Tests t = new Tests( oneM_pop: "384724", total: 2384789L);
    CountryStats cs = new CountryStats( continent: "Europe", country: "Portugal", population: 12L,c,d,t, day: "2022-03-25", time: "12:00");
    cacheData.put("Portugal",cs);
    this.cache.setCacheData(cacheData);
    assertEquals(cacheData,this.cache.getCacheData(), message: "Obtained cache data different from the setted");
}
```

Figure 10: Example of Unit Tests

- Unit Tests (with Mocks)
    - getCountries_returnsListOfCountries()
    - getStatisticsByCountry_returnsStatisticsOfCountry()
    - getStatisticsByWrongCountry_returnsEmptyOptional()
    - getHistoryByCountry_returnsHistoryOfCountry()

Example of Unit Tests (with Mocks):

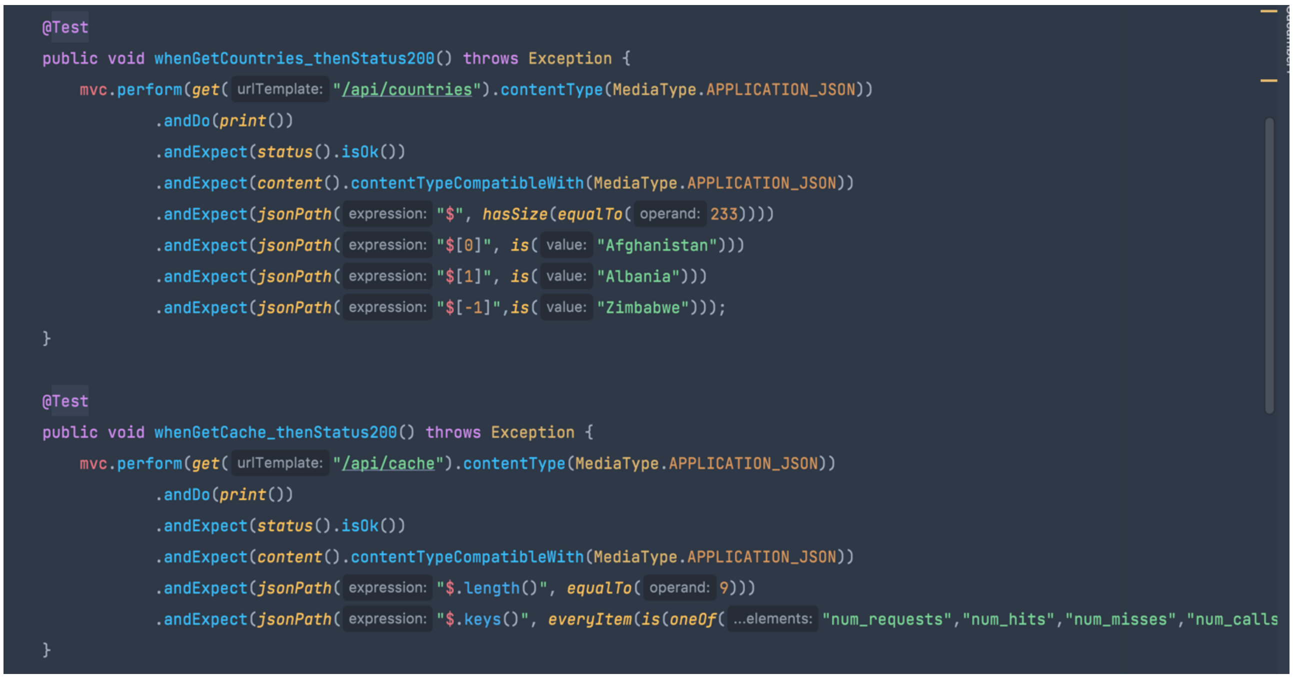


```java
@Test
public void getCountries_returnsListOfCountries() throws IOException, ParseException, InterruptedException {
    List<String> countriesL = new ArrayList<>();
    countriesL.add("Portugal");
    countriesL.add("Spain");
    assertThat(countryService.getCountriesList()).isEqualTo(countriesL);
}

@Test
public void getStatisticsByCountry_returnsStatisticsOfCountry() throws IOException, ParseException, InterruptedException {
    Cases cas = new Cases( newCases: null, active: null, critical: 61L, recovered: null, oneM_pop: "373836", total: 3791744L);
    Deaths dea = new Deaths( newDeaths: null, oneM_pop: "2185", total: 22162L);
    Tests tes = new Tests( oneM_pop: "4053425", total: 41113089L);
    CountryStats cs = new CountryStats( continent: "Europe", country: "Portugal", population: 10142802L,cas,dea,tes, day: "2022-04-27", time: "
    assertThat(countryService.getStatisticsByCountry("portugal")).isEqualTo(Optional.of(cs));
}

@Test
public void getStatisticsByWrongCountry_returnsEmptyOptional() throws IOException, ParseException, InterruptedException {
    assertThat(countryService.getStatisticsByCountry("fakecountry")).isEqualTo(Optional.empty());
}
```

Figure 11: Example of Unit Tests (with Mocks)

- Integration Tests
    - whenGetCountries_thenStatus200()
    - whenGetCache_thenStatus200()
    - whenGetStatisticsByCountry_thenStatus200()
    - whenGetHistoryByCountry_thenStatus200()

Example of Integration Tests:



```
@Test
public void whenGetCountries_thenStatus200() throws Exception {
    mvc.perform(get( urlTemplate: "/api/countries").contentType(MediaType.APPLICATION_JSON))
            .andDo(print())
            .andExpect(status().isOk())
            .andExpect(content().contentTypeCompatibleWith(MediaType.APPLICATION_JSON))
            .andExpect(jsonPath( expression: "$", hasSize(equalTo( operand: 233))))
            .andExpect(jsonPath( expression: "$[0]", is( value: "Afghanistan")))
            .andExpect(jsonPath( expression: "$[1]", is( value: "Albania")))
            .andExpect(jsonPath( expression: "$[-1]",is( value: "Zimbabwe")));
}

@Test
public void whenGetCache_thenStatus200() throws Exception {
    mvc.perform(get( urlTemplate: "/api/cache").contentType(MediaType.APPLICATION_JSON))
            .andDo(print())
            .andExpect(status().isOk())
            .andExpect(content().contentTypeCompatibleWith(MediaType.APPLICATION_JSON))
            .andExpect(jsonPath( expression: "$.length()", equalTo( operand: 9)))
            .andExpect(jsonPath( expression: "$.keys()", everyItem(is(oneOf( ...elements: "num_requests","num_hits","num_misses","num_calls
}
```

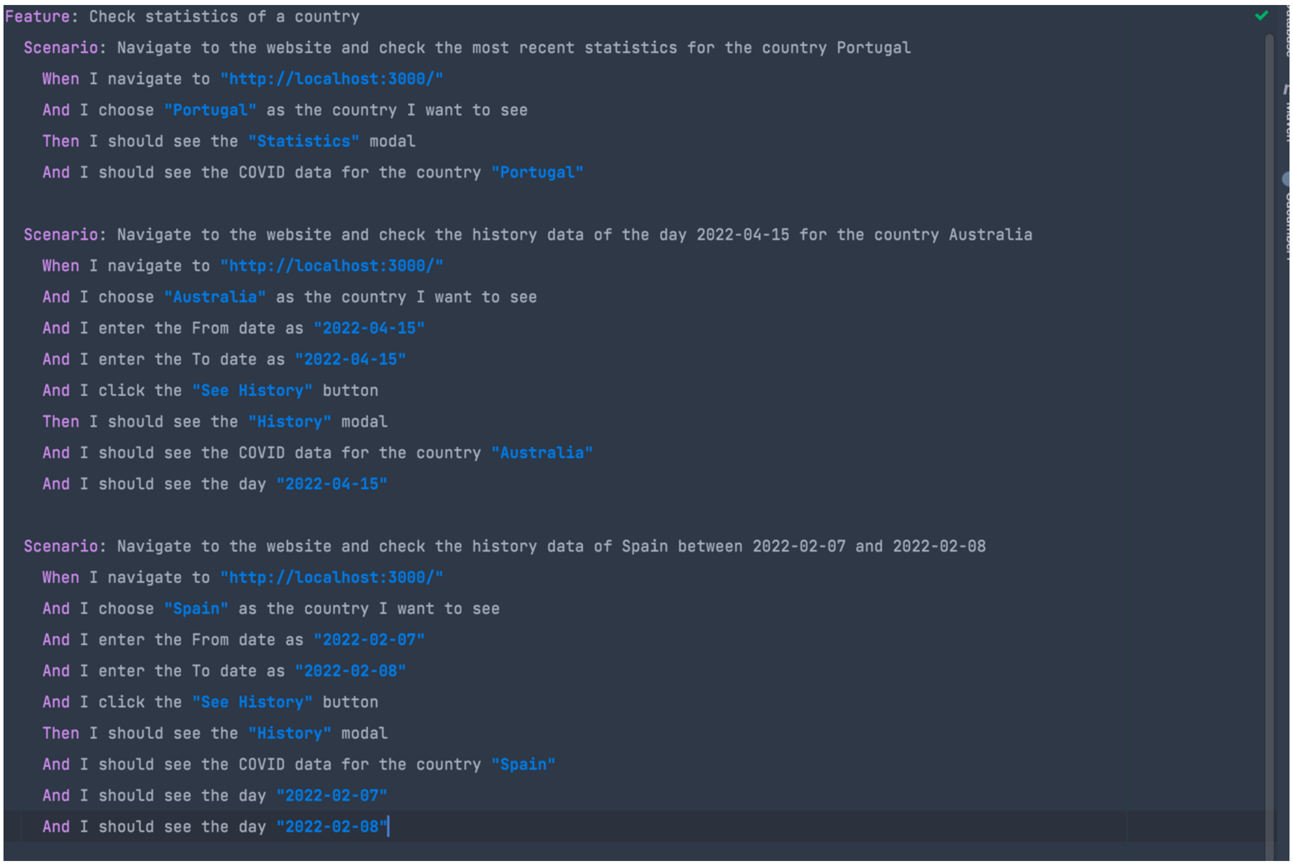
Figure 12: Example of Integration Tests

3.3 Functional testing

For the functional testing I tested 3 scenarios:

- Navigate to the website and check the most recent statistics for the country Portugal
- Navigate to the website and check the history data of the day 2022-04-15 for the country Australia
- Navigate to the website and check the history data of Spain between 2022-02-07 and 2022-02-08

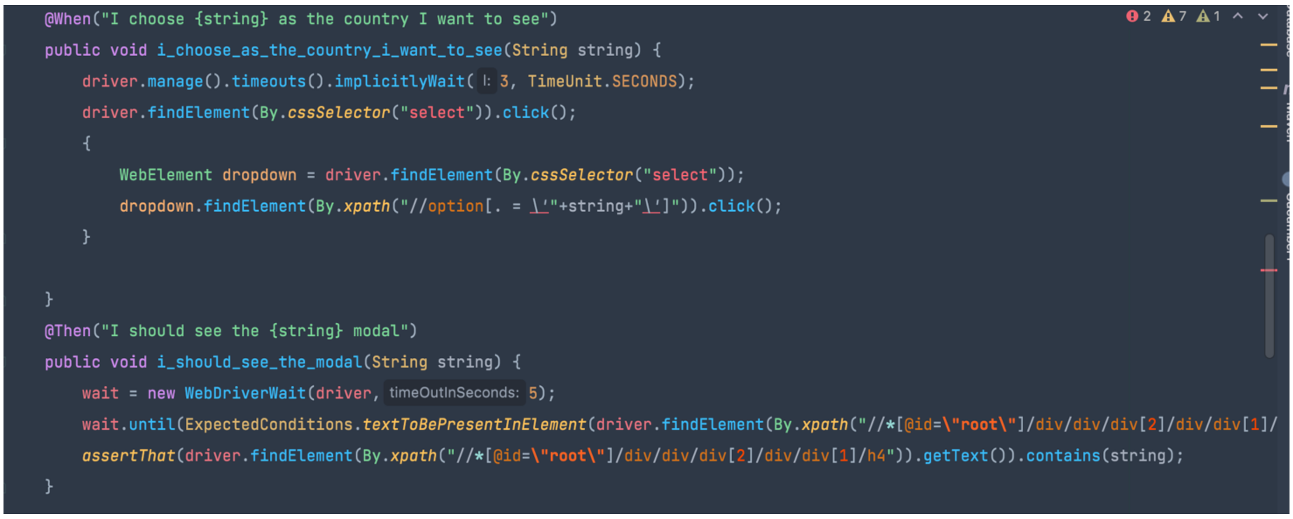
Here are the steps of each scenario:



```
Feature: Check statistics of a country
  Scenario: Navigate to the website and check the most recent statistics for the country Portugal
    When I navigate to "http://localhost:3000/"
    And I choose "Portugal" as the country I want to see
    Then I should see the "Statistics" modal
    And I should see the COVID data for the country "Portugal"

  Scenario: Navigate to the website and check the history data of the day 2022-04-15 for the country Australia
    When I navigate to "http://localhost:3000/"
    And I choose "Australia" as the country I want to see
    And I enter the From date as "2022-04-15"
    And I enter the To date as "2022-04-15"
    And I click the "See History" button
    Then I should see the "History" modal
    And I should see the COVID data for the country "Australia"
    And I should see the day "2022-04-15"

  Scenario: Navigate to the website and check the history data of Spain between 2022-02-07 and 2022-02-08
    When I navigate to "http://localhost:3000/"
    And I choose "Spain" as the country I want to see
    And I enter the From date as "2022-02-07"
    And I enter the To date as "2022-02-08"
    And I click the "See History" button
    Then I should see the "History" modal
    And I should see the COVID data for the country "Spain"
    And I should see the day "2022-02-07"
    And I should see the day "2022-02-08"
```

Figure 13: Functional Testing Scenarios

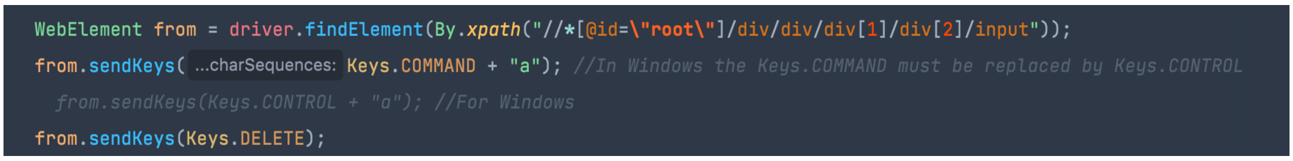
And the implementation of some of them:



```
    @When("I choose {string} as the country I want to see")
    public void i_choose_as_the_country_i_want_to_see(String string) {
        driver.manage().timeouts().implicitlyWait( l: 3, TimeUnit.SECONDS);
        driver.findElement(By.cssSelector("select")).click();
        {
            WebElement dropdown = driver.findElement(By.cssSelector("select"));
            dropdown.findElement(By.xpath("//option[. = \'"+string+"\']")).click();
        }

    }
    @Then("I should see the {string} modal")
    public void i_should_see_the_modal(String string) {
        wait = new WebDriverWait(driver, timeOutInSeconds: 5);
        wait.until(ExpectedConditions.textToBePresentInElement(driver.findElement(By.xpath("//*[@id=\"root\"]/div/div/div[2]/div/div[1]/
        assertThat(driver.findElement(By.xpath("//*[@id=\"root\"]/div/div/div[2]/div/div[1]/h4")).getText()).contains(string);
    }
```

Figure 14: Example of Functional Tests

NOTE: On the functional tests I used ‘Keys.COMMAND’ to use a shortcut on my test. I used the ‘Keys.COMMAND’ because I’m using macOS, but if the user is using Windows ‘Keys.CONTROL’ must be used instead. The line is commented bellow the one I use.

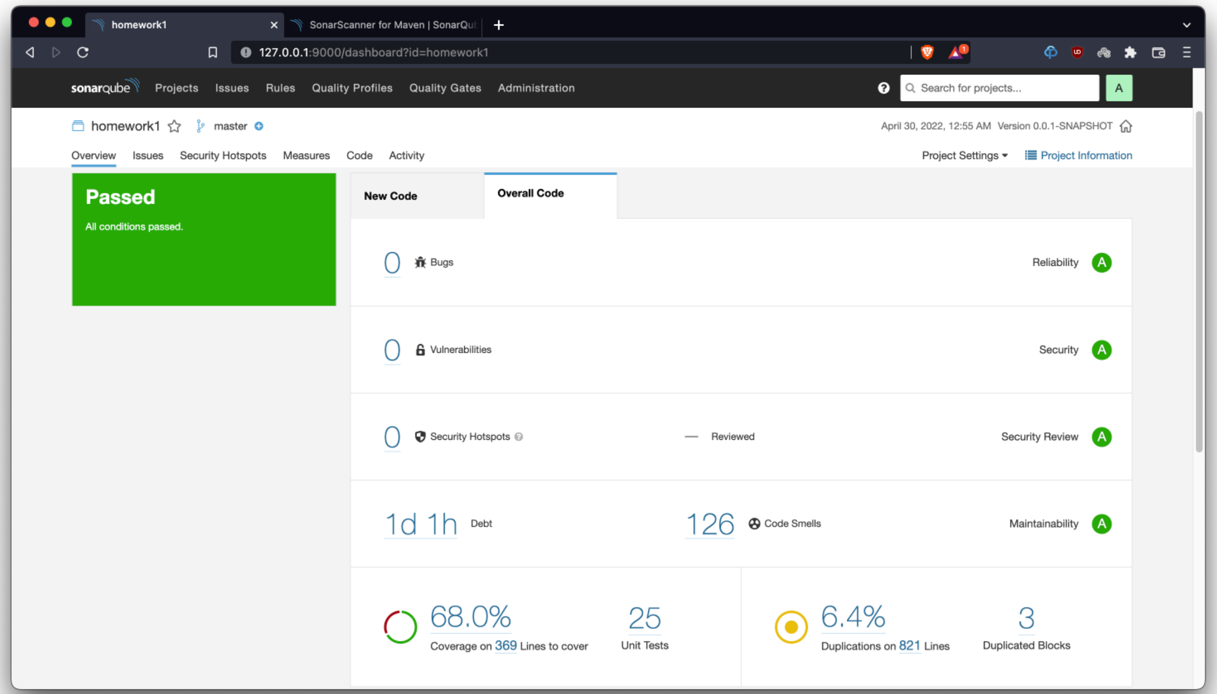


```
WebElement from = driver.findElement(By.xpath("//*[@id=\"root\"]/div/div/div[1]/div[2]/input"));
from.sendKeys( ...charSequences: Keys.COMMAND + "a"); //In Windows the Keys.COMMAND must be replaced by Keys.CONTROL
  from.sendKeys(Keys.CONTROL + "a"); //For Windows
from.sendKeys(Keys.DELETE);
```

Figure 15: Example of the shortcut

3.4 Code quality analysis

I used SonarQube to analyze my code.
The result was the following:



homework1 | SonarScanner for Maven | SonarQu

127.0.0.1:9000/dashboard?id=homework1

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

homework1 master — April 30, 2022, 12:55 AM Version 0.0.1-SNAPSHOT

Overview Issues Security Hotspots Measures Code Activity — Project Settings, Project Information

Passed
All conditions passed.

New Code | Overall Code

0 Bugs		Reliability A
0 Vulnerabilities		Security A
0 Security Hotspots	— Reviewed	Security Review A
1d 1h Debt	126 Code Smells	Maintainability A

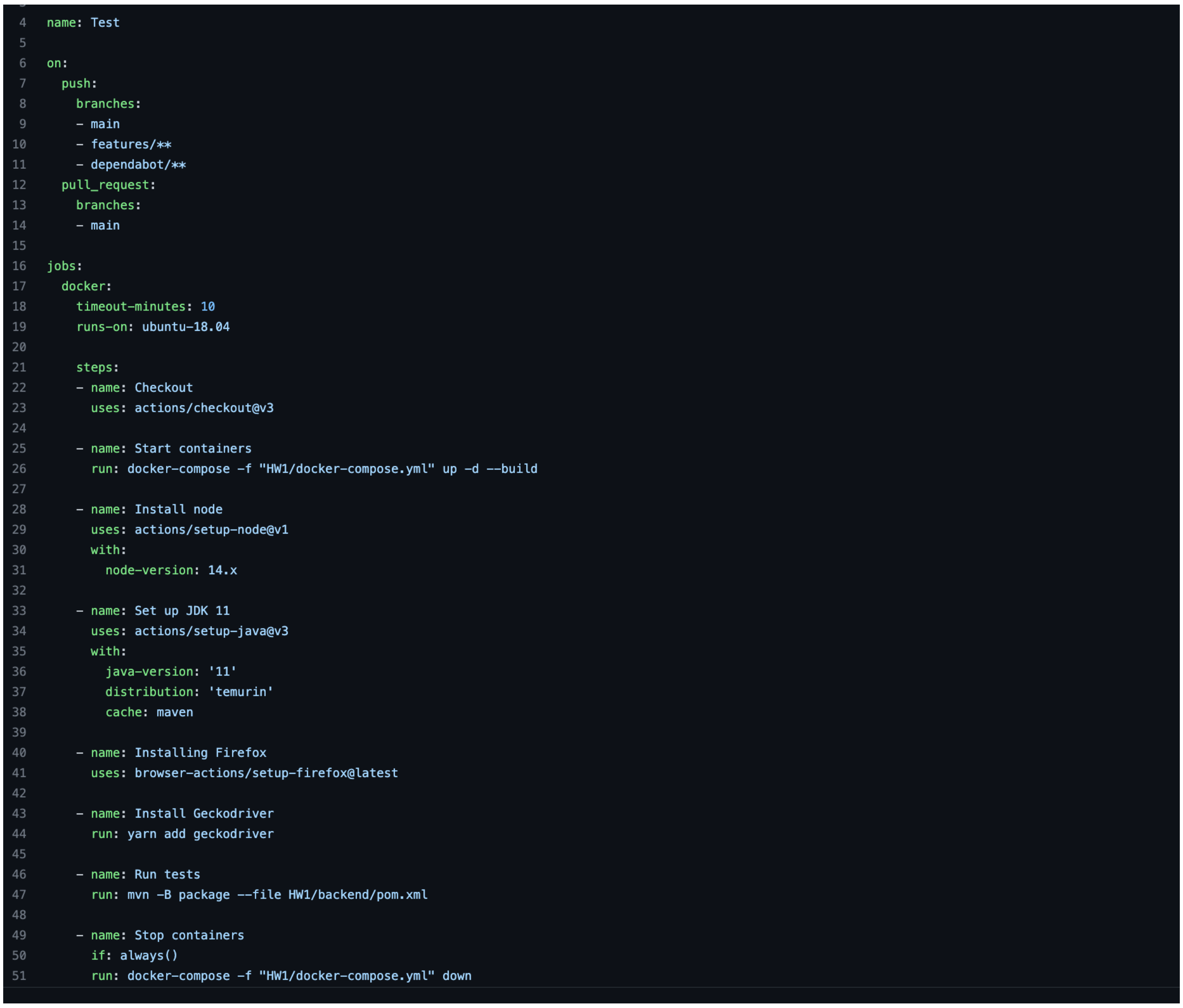
68.0% Coverage on 369 Lines to cover — 25 Unit Tests — 6.4% Duplications on 821 Lines — 3 Duplicated Blocks

Figure 16: SonarQube Analysis Results

As we can see, the application didn’t have any bugs so it was graded ‘A’ on Reliability; didn’t have vulnerabilities so it was graded ‘A’ on Security and was also graded ‘A’ on Security Review and Maintainability.
The project had a lot of code smells, having a debt of 1d1h, but I didn’t find any of them to be very problematic and since I lacked time I left it that way.
Now speaking about the coverage, it was marked 68%. This is not a very high percentage but in my analysis I found that most of the methods not tested were ‘setters’, ‘hashCode’ and ‘equals’ that were auto generated for every class of the project. If we ignored those methods, the coverage would be very much higher.

3.5 Continuous integration pipeline

I implemented a Continuous Integration Pipeline using GitHub Actions.
The action was to run the docker app, install the dependencies and install the Firefox browser. After that, it would build the maven project, running the tests.
I had some difficulties defining this action because of the Selenium tests.
Every Unit Test and Integration Test passes without problems, but the Functional Tests don't pass. I didn't understand the problem because locally, on my machine, every test runs and passes, but on GitHub Actions it doesn't.



```yaml
name: Test

on:
  push:
    branches:
    - main
    - features/**
    - dependabot/**
  pull_request:
    branches:
    - main

jobs:
  docker:
    timeout-minutes: 10
    runs-on: ubuntu-18.04

    steps:
    - name: Checkout
      uses: actions/checkout@v3

    - name: Start containers
      run: docker-compose -f "HW1/docker-compose.yml" up -d --build

    - name: Install node
      uses: actions/setup-node@v1
      with:
        node-version: 14.x

    - name: Set up JDK 11
      uses: actions/setup-java@v3
      with:
        java-version: '11'
        distribution: 'temurin'
        cache: maven

    - name: Installing Firefox
      uses: browser-actions/setup-firefox@latest

    - name: Install Geckodriver
      run: yarn add geckodriver

    - name: Run tests
      run: mvn -B package --file HW1/backend/pom.xml

    - name: Stop containers
      if: always()
      run: docker-compose -f "HW1/docker-compose.yml" down
```

Figure 17: maven.yml file

4 References & resources

Project resources

Resource:	URL/location:
Git repository	https://github.com/MiguelF07/TQS_98599
Video demo	https://youtu.be/HsbP_gtL0ac

Reference materials

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