

HW1: Mid-term assignment report

Eduardo Henrique Ferreira Santos [93107], v2021-05-14

<https://github.com/eduardosantoshf/AirQuality>



1	Introduction	1
1.1	Overview of the work.....	1
1.2	Current limitations.....	2
2	Product specification.....	3
2.1	Functional scope and supported interactions.....	3
2.2	System architecture.....	6
2.3	API for developers	8
3	Quality assurance	10
3.1	Overall strategy for testing	10
3.2	Unit and integration testing.....	10
3.3	Functional testing.....	10
3.4	Static code analysis	14
3.5	Continuous integration pipeline [optional].....	16
4	References & resources	17

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 objective of this project was to create a **REST-API service**, along with the implementation of tests to verify if everything in the application is working properly. These tests included:

- Unit tests.
- Unit tests with dependency isolation using mocks.
- Integration tests on API.
- Functional tests.

The **AirQuality** product allows its user to get air quality data from a specific city, showing, for instance, the AQI (Air Quality Index) and some concentration levels (CO, O3, SO2, NO2), as well as the predominant pollen type of that specific city.

The data regarding the air quality for each city is gathered from the [WeatherBit's](#) Air Quality external API, and can be searched on the products' webpage, either by the city name or by the latitude and longitude.

There was also implemented a local cache, making repeated calls being stored for 1 minute so that the retrieval of information is faster.

Regarding the cache, there were two types of cache implemented:

- **CityCache** and **CityCacheItem** – this type of cache uses a capacity attribute, this establishes the max number of cities to store, using a FIFO (First-In, First-Out) method, which means that, if the maximum limit is reached, the first item that entered the HashMap, is the first item to be removed from it.
- **CityTTLCache** – this type of cache implements TTL (Time-To-Leave), meaning that, when that time passes, all of the cities store for more than that time are removed from the HashMap.

1.2 Running the Application

AirQuality can be locally deployed. That can be done by running the following command on the [repository's](#) root directory: *docker-compose up*

When everything is up and running, the website can be accessed at: <http://localhost:8000/>

1.3 Current limitations

This product only used one external API, this means that, if that API is, for some reason, down, it won't be able to present any data at all. This would be the next thing to implement.

Another thing that could be implemented in the future is the search for air quality data by day, this could be presented in graphs so that it could give an overview about that specific city, for the last n days.

2 Product specification

2.1 Functional scope and supported interactions

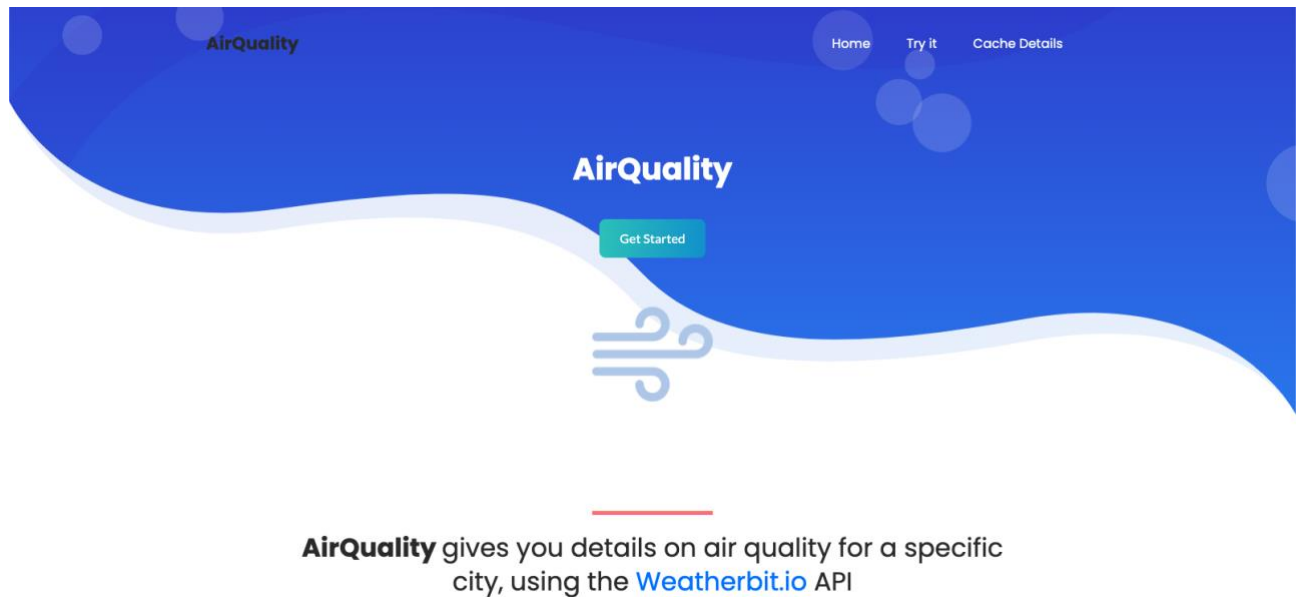


Figure 1 – AirQuality's Webpage

The user interacts with the product through a single page application, this makes the interactions easier and more intuitive.

As previously mentioned, we can search for air quality data for a specific city by name, as show in the picture bellow:

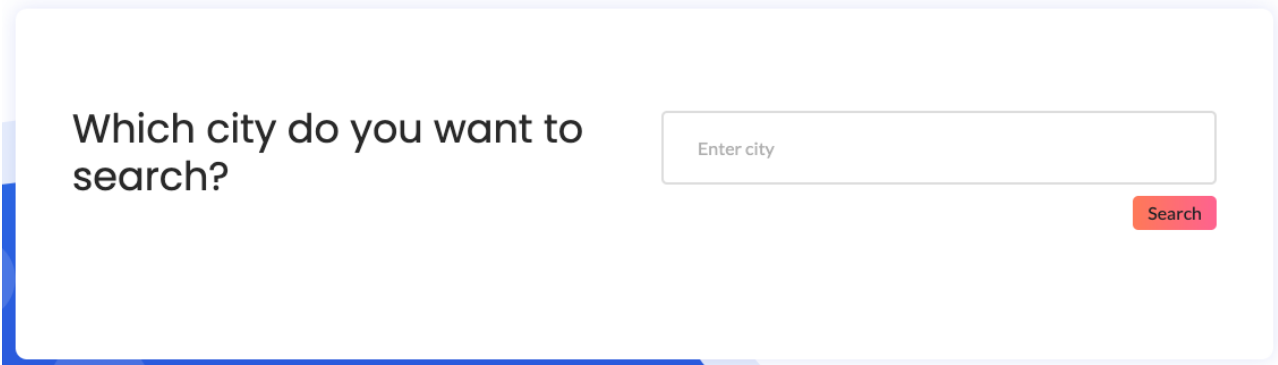
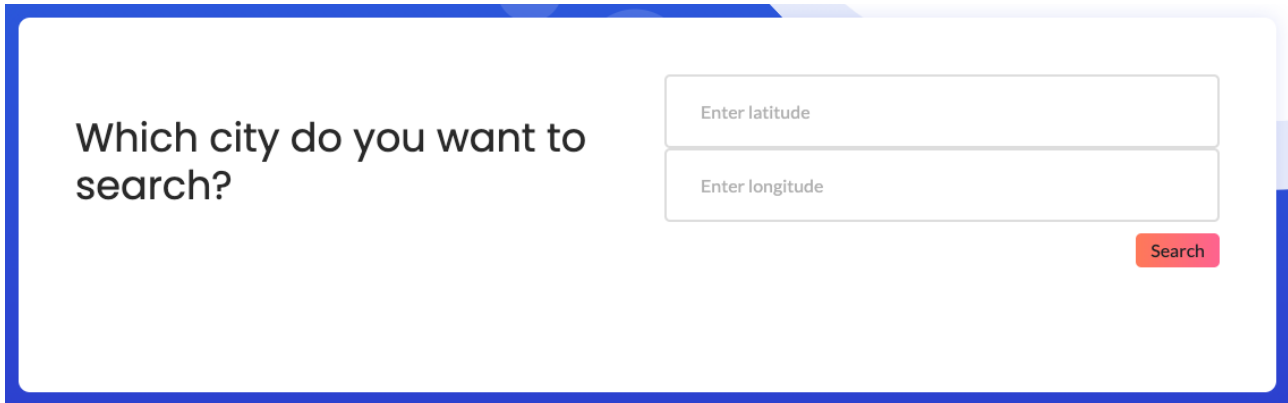
The image shows a search form with the text 'Which city do you want to search?' on the left. To the right is a text input field with the placeholder 'Enter city'. Below the input field is a red 'Search' button.

Figure 2 - Search City by Name

We can also search for air quality data for a specific city by latitude and longitude, as show in the picture bellow:

A web form with a blue border. On the left, the text "Which city do you want to search?" is displayed. To the right, there are two stacked input fields. The top field is labeled "Enter latitude" and the bottom field is labeled "Enter longitude". Below these fields, on the right side, is a red "Search" button.

Which city do you want to search?

Enter latitude

Enter longitude

Search

Figure 3 - Search City by Latitude and Longitude

Lastly, we can check the cache details:

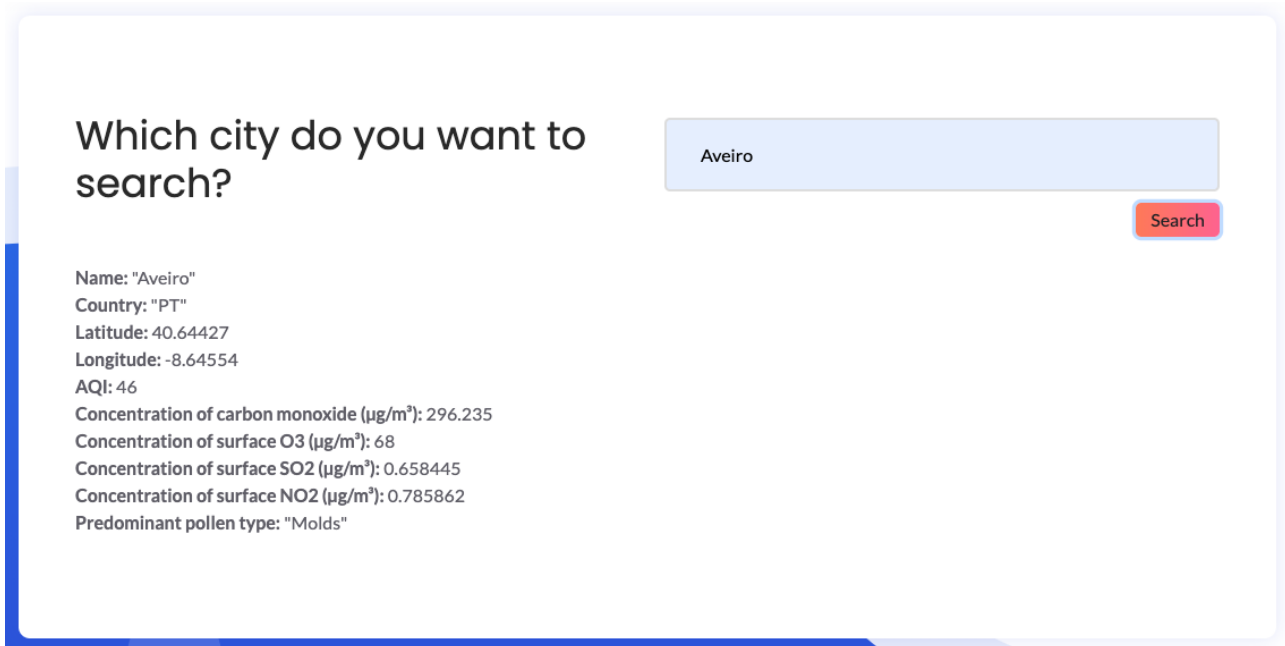
A web form with a blue border. On the left, the text "Cache Details" is displayed. On the right side, there is a red "Search" button.

Cache Details

Search

Figure 4 - Search Cache Details

For instance, if we searched for the city *Aveiro*, this would be the output:



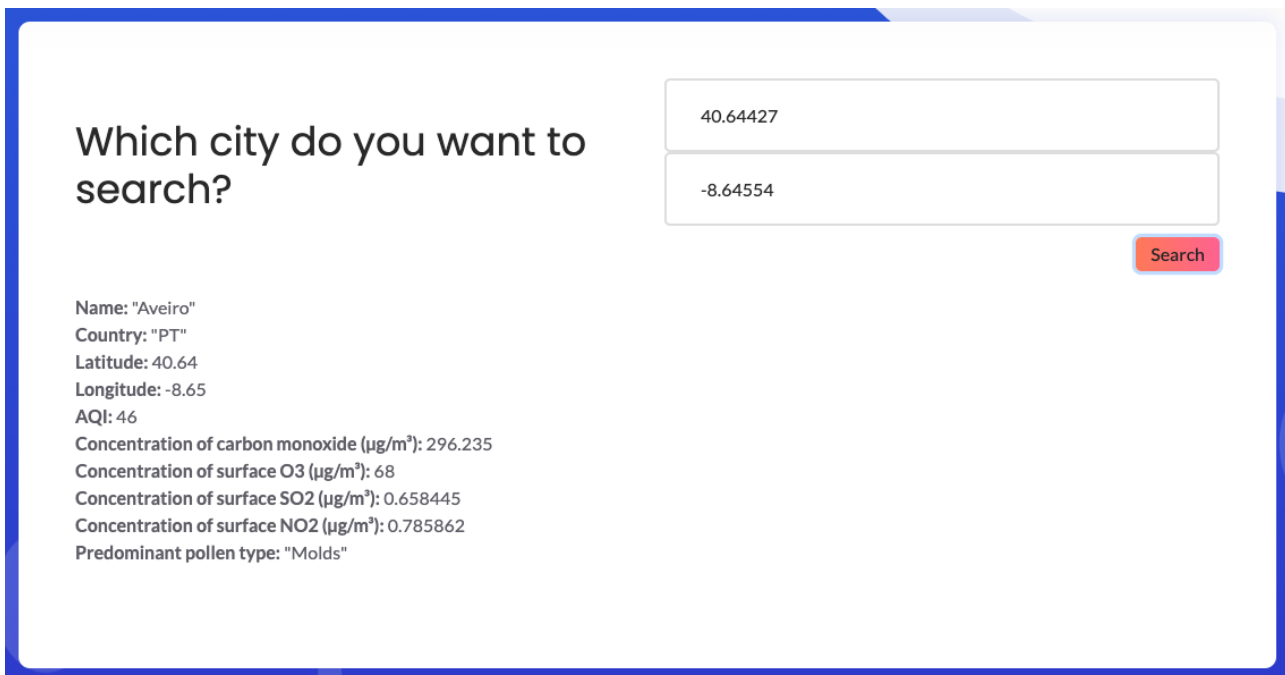
Which city do you want to search?

Aveiro

Search

Name: "Aveiro"
Country: "PT"
Latitude: 40.64427
Longitude: -8.64554
AQI: 46
Concentration of carbon monoxide ($\mu\text{g}/\text{m}^3$): 296.235
Concentration of surface O3 ($\mu\text{g}/\text{m}^3$): 68
Concentration of surface SO2 ($\mu\text{g}/\text{m}^3$): 0.658445
Concentration of surface NO2 ($\mu\text{g}/\text{m}^3$): 0.785862
Predominant pollen type: "Molds"

Figure 5



Which city do you want to search?

40.64427

-8.64554

Search

Name: "Aveiro"
Country: "PT"
Latitude: 40.64
Longitude: -8.65
AQI: 46
Concentration of carbon monoxide ($\mu\text{g}/\text{m}^3$): 296.235
Concentration of surface O3 ($\mu\text{g}/\text{m}^3$): 68
Concentration of surface SO2 ($\mu\text{g}/\text{m}^3$): 0.658445
Concentration of surface NO2 ($\mu\text{g}/\text{m}^3$): 0.785862
Predominant pollen type: "Molds"

Figure 6



Figure 7

2.2 System architecture

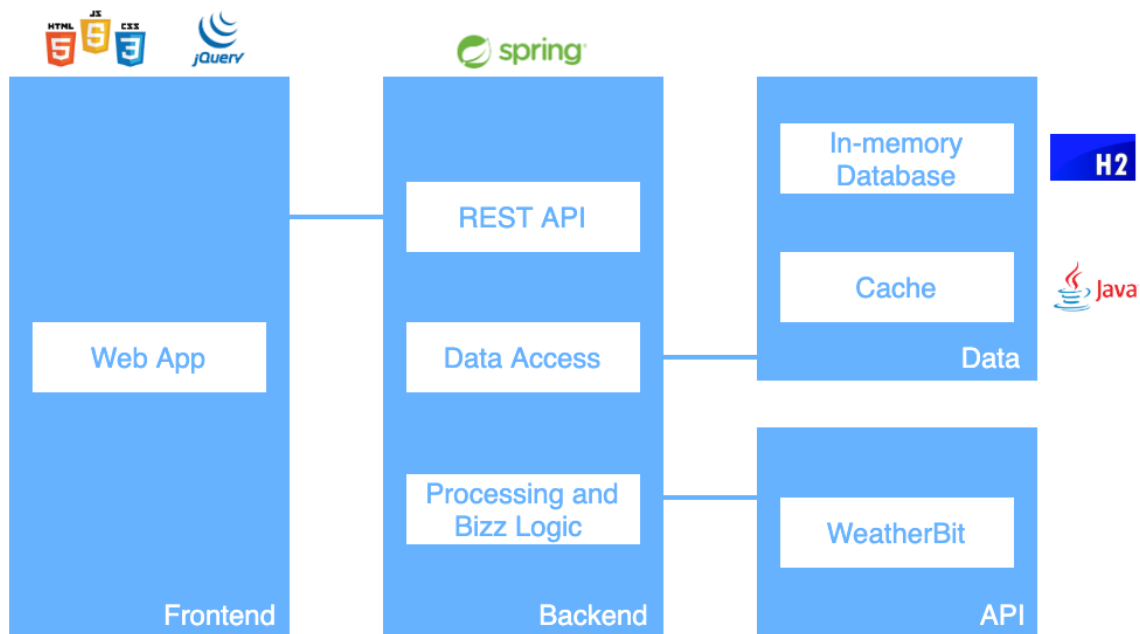


Figure 8 - AirQuality's System Architecture

For the frontend part the **HTML**, **CSS**, and **JS** were used, as well as **jQuery's AJAX** to make the API requests.

The backend was built using **Spring Boot**, a module of **Spring Framework**. It was also used an in-memory database, in this case the **H2 Database**.

2.3 Spring Boot Project UML Class Diagram

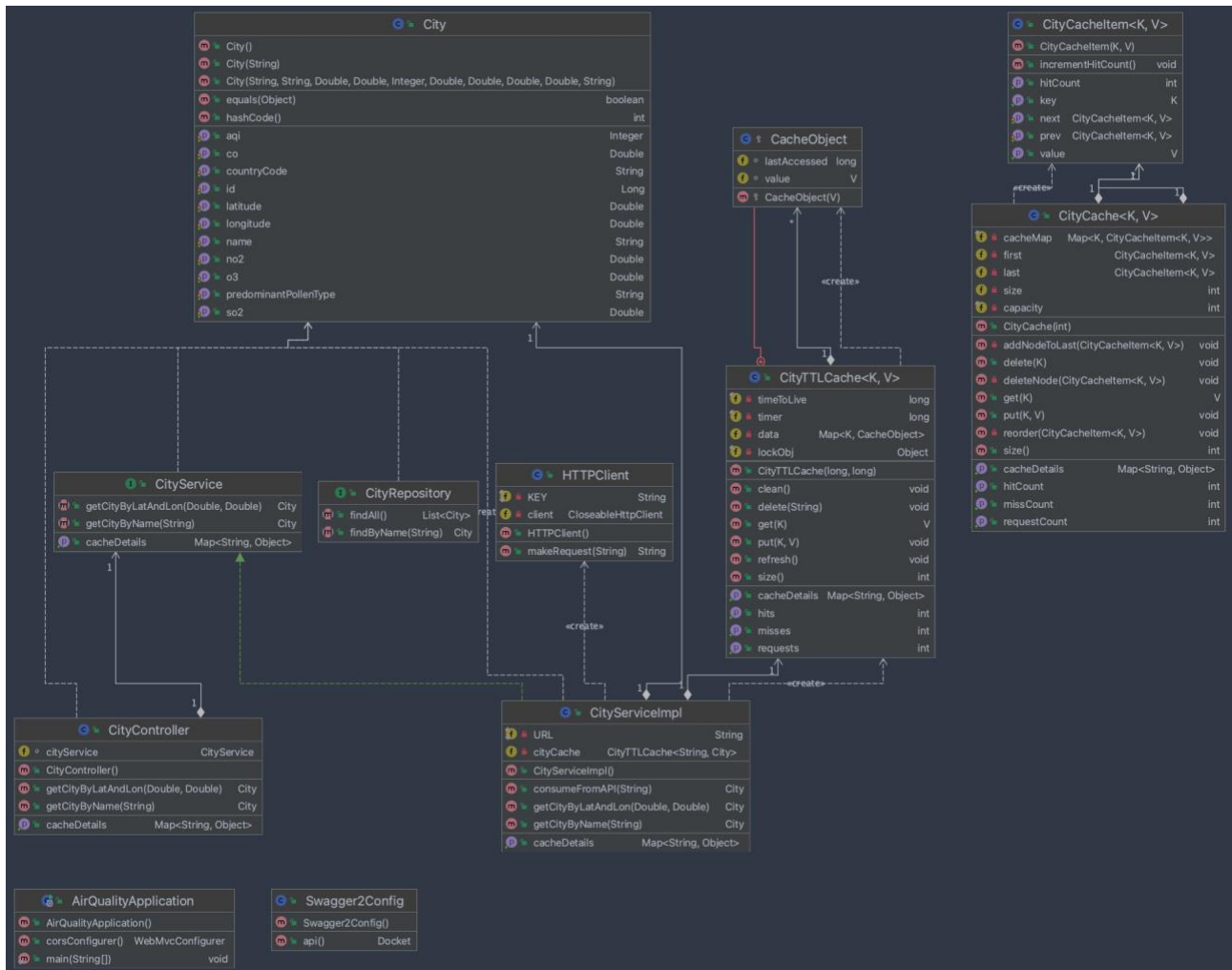


Figure 9 - Project UML Class Diagram

As we can see, this implementation follows common practices to implement a Spring Boot solution, this being said, we have the following components on it:

- **City** (@Entity) – represents a domain concept.
- **CityRepository** (@Repository) – the interface defines the data access methods used on the City entity, based on the JpaRepository framework.
- **CityService** and **CityServiceImpl** (@Service) – this defines the interface as well as its implementation of a service related to the “business logic” of the app.
- **CityController** (@Controller) – this handles the HTTP requests, delegating them to the **CityService**.

2.4 API for developers

For documenting the API, **Swagger2** was used.

To access the API's documentation page, the app needs to be up and running. After that, the documentation can be accessed at: <http://localhost:8080/swagger-ui.html>

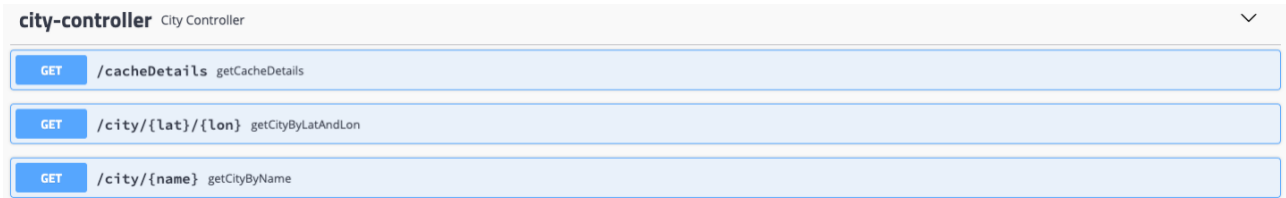


Figure 10 - API's Swagger2 Page

The endpoints of this API are:

- **GET /cacheDetails** – this will give us the cache details:
 - Parameters: no parameters
 - Output:

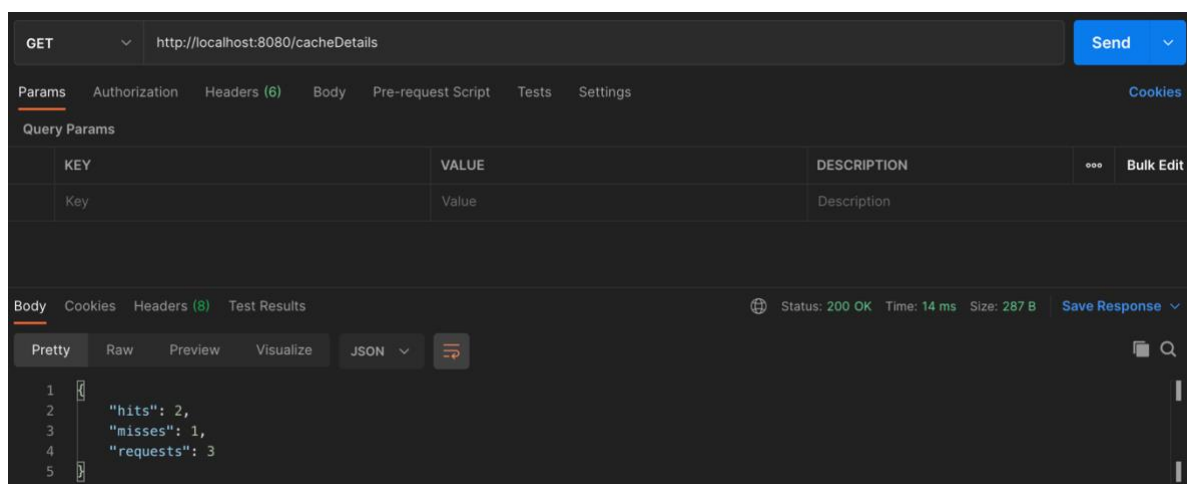
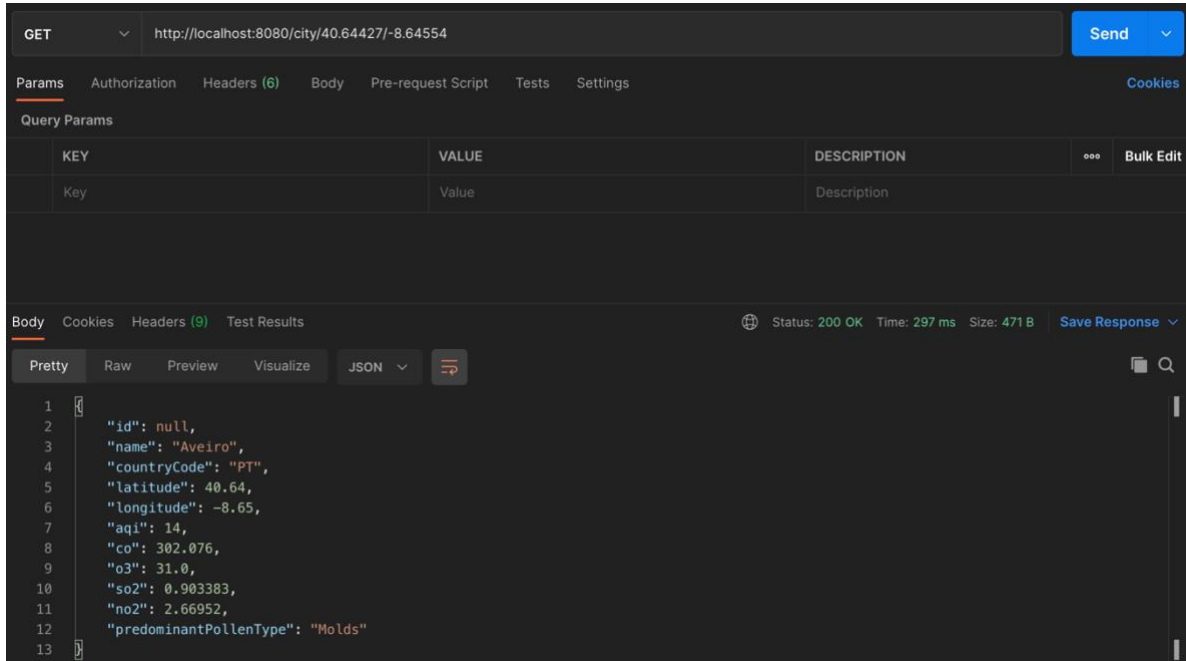


Figure 11 - GET /cacheDetails example

- **GET** /city/{lat}/{lon} – this will give us the city, given the latitude and longitude:
 - Parameters:
 - lat (Double)
 - lon (Double)
 - Output:



GET

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

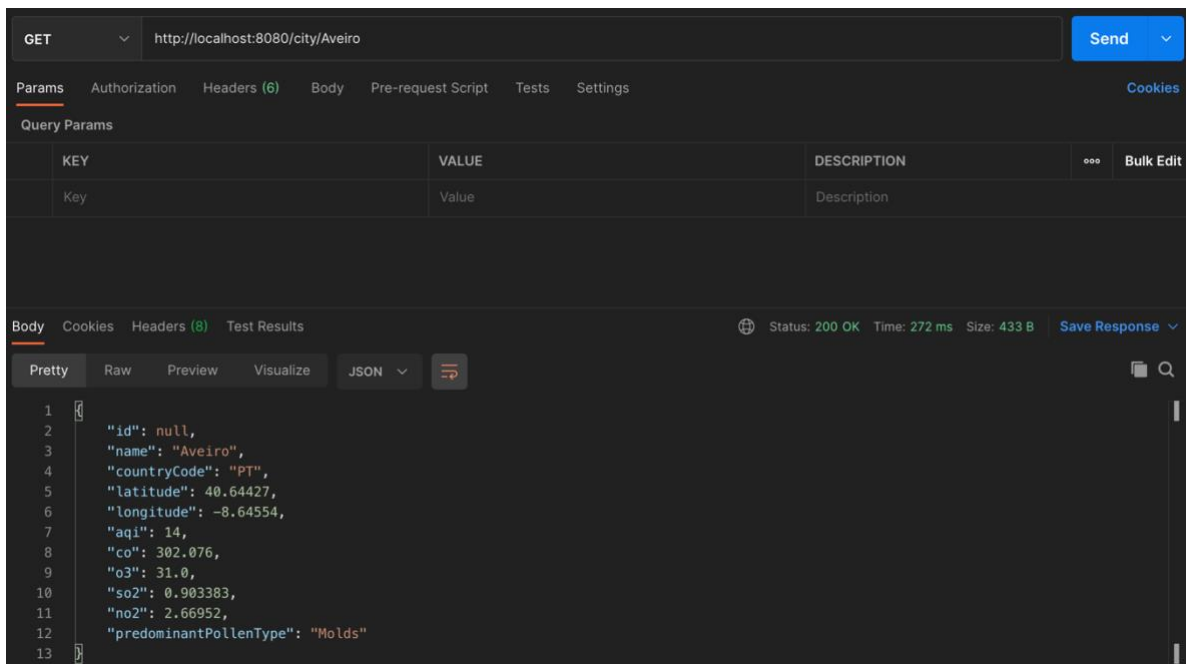
KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (9) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": null,
3   "name": "Aveiro",
4   "countryCode": "PT",
5   "latitude": 40.64,
6   "longitude": -8.65,
7   "aqi": 14,
8   "co": 302.076,
9   "o3": 31.0,
10  "so2": 0.903383,
11  "no2": 2.66952,
12  "predominantPollenType": "Molds"
13 }
```

- **GET** /city/{name} – this will give us the city, given the name:
 - Parameters:
 - name (String)
 - Output:



GET

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": null,
3   "name": "Aveiro",
4   "countryCode": "PT",
5   "latitude": 40.64427,
6   "longitude": -8.64554,
7   "aqi": 14,
8   "co": 302.076,
9   "o3": 31.0,
10  "so2": 0.903383,
11  "no2": 2.66952,
12  "predominantPollenType": "Molds"
13 }
```

3 Quality assurance

3.1 Overall strategy for testing

Regarding the strategy for testing, it was adopted a **TDD** (Test Driven Development). Software requirements were converted to test cases before software was fully developed, and the tracking of all software development was made by testing the software repeatedly.

Regarding the order of the tests' implementation, I started from the controller, then the service, then the repository.

3.2 Unit and integration testing

The tools/frameworks used for unit and integration testing were:

- **Junit 5**
- **Mockito**
- **Spring Boot MockMVC**

3.2.1 Unit Tests

Regarding unit testing, the following tests were implemented:

- Cache tests:
 - **CityCacheTest**
 - **CityTTLCacheTest**

```
@BeforeEach
public void setUp() { cityCache = new CityCache<>( capacity: 2); }

@Test
void putTest() {
    City aveiro = new City( name: "Aveiro");
    cityCache.put("Aveiro", aveiro);

    assertEquals(aveiro, cityCache.get("Aveiro"));
}

@Test
void getTest() {
    City porto = new City( name: "Porto");
    cityCache.put("Aveiro", porto);

    assertEquals(porto, cityCache.get("Aveiro"));
}
```

Figure 12 - Example of **CityCacheTest**

```

@BeforeEach
void setUp() { cityTTLCache = new CityTTLCache<>(timeToLive: 5, timer: 5); }

@Test
void putTest() {
    City aveiro = new City( name: "Aveiro");
    cityTTLCache.put("Aveiro", aveiro);

    assertEquals(aveiro, cityTTLCache.get("Aveiro"));
}

@Test
void getTest() {
    City porto = new City( name: "Porto");
    cityTTLCache.put("Aveiro", porto);

    assertEquals(porto, cityTTLCache.get("Aveiro"));
}

@Test
void deleteTest() {
    City aveiro = new City( name: "Aveiro");
    cityTTLCache.put("Aveiro", aveiro);
    int cacheSize = cityTTLCache.size();

    cityTTLCache.delete( key: "Aveiro");

    assertEquals( expected: cityTTLCache.size() + 1, cacheSize);
}

```

Figure 13 - Example of *CityTTLCacheTest*

- **CityRepositoryTest:**

```

@Autowired
private TestEntityManager entityManager;

@Autowired
private CityRepository cityRepository;

@Test
void whenFindAveiroByName_ThenReturnAveiroCity() {
    City aveiro = new City( name: "Aveiro");
    entityManager.persistAndFlush(aveiro);

    City foundCity = cityRepository.findByName(aveiro.getName());
    assertEquals(foundCity, aveiro);
}

@Test
void whenInvalidCityName_thenReturnNull() {
    City cityFromDb = cityRepository.findByName("Does Not Exist");
    assertNull(cityFromDb);
}

```

Figure 14 - Example of *CityRepositoryTest*

3.2.2 Unit Tests with Mocks

Regarding unit testing with Mocks, the following tests were implemented:

- **CityServiceTest:**

```
@Mock(lenient = true)
private CityRepository cityRepository;

@InjectMocks
private CityServiceImpl cityService;

@BeforeEach
void setUp() {
    City aveiro = new City( name: "Aveiro");
    City porto = new City( name: "Porto");
    City lisboa = new City( name: "Lisboa");

    List<City> cities = Arrays.asList(aveiro, porto, lisboa);

    Mockito.when(cityRepository.findByName(aveiro.getName())).thenReturn(aveiro);
    Mockito.when(cityRepository.findByName(porto.getName())).thenReturn(porto);
    Mockito.when(cityRepository.findByName(lisboa.getName())).thenReturn(lisboa);
    Mockito.when(cityRepository.findByName("wrong_name")).thenReturn(null);
    Mockito.when(cityRepository.findAll()).thenReturn(cities);
}

@Test
void whenValidName_thenCityShouldBeFound() throws IOException, URISyntaxException {
    String name = "Aveiro";
    City foundCity = cityService.getCityByName(name);

    assertThat(foundCity.getName().isEqualTo(name);
}
```

Figure 15 - Example of *CityServiceTest*

3.2.3 Integration test

Regarding integration testing, the following tests were implemented:

- **CityControllerTest:**

```
@Autowired
private MockMvc mvc;

@MockBean
private CityService cityService;

@Test
void getCityByNameTest() throws Exception {
    City aveiro = new City( name: "Aveiro");

    when(cityService.getCityByName("Aveiro")).thenReturn(aveiro);

    mvc.perform(get( uriTemplate: "/city/{name}", ...uriVars: "Aveiro")
        .contentType(MediaType.APPLICATION_JSON))
        .andExpect(status().isOk())
        .andExpect(jsonPath( expression: "$.name", is( value: "Aveiro")));
}

@Test
void getCityByLatAndLonTest() throws Exception {
    City aveiro = new City(
        name: "Aveiro", countryCode: "PT",
        latitude: 40.64427, longitude: -8.64554,
        aqi: 41, co: 270.784, o3: 57.0, so2: 0.708736, no2: 0.57969,
        predominantPollenType: "Molds");

    when(cityService.getCityByLatAndLon( lat: 40.64427, lon: -8.64554)).thenReturn(aveiro);

    mvc.perform(get( uriTemplate: "/city/{lat}/{lon}", ...uriVars: 40.64427, -8.64554)
        .contentType(MediaType.APPLICATION_JSON))
        .andExpect(status().isOk())
        .andExpect(jsonPath( expression: "$.latitude", is( value: 40.64427)))
        .andExpect(jsonPath( expression: "$.longitude", is( value: -8.64554)));
}
```

Figure 16 - Example of *CityControllerTest*

3.3 Functional testing

When it comes to functional testing, the **Selenium Web Driver** was used, along with this we used the **Firefox Driver**, because the Chrome Driver is known to give some problems.

```
WebDriver browser = new FirefoxDriver();

@BeforeEach
void setUp() { WebDriver browser; }

@Test
void searchCityByName() {
    browser.get("http://127.0.0.1:8000/");
    JavascriptExecutor js = (JavascriptExecutor) browser;
    js.executeScript( s: "window.scrollTo(0,750)");
    browser.manage().window().setSize(new Dimension( width: 1116, height: 697));
    browser.findElement(By.id("cityToBeSearched")).click();
    browser.findElement(By.id("cityToBeSearched")).sendKeys( ...charSequences: "Aveiro");
    browser.findElement(By.id("searchBtn")).click();
    assertEquals( expected: "Name: \"Aveiro\"", browser.findElement(By.id("name1")).getText());
}
```

Figure 17 - Example of *WebpageTest*

This kind of tests presented some issues, one of them was, because the webpage had some fade delays, when running the **Selenium** tests, it presented us with errors saying that some *ids* weren't found, this happened because the *divs* with those *ids* weren't visible yet on the webpage.

This problem was fixed with the following lines of code:

```
browser.manage().window().setSize(new Dimension( width: 1116, height: 698));
try {
    if (browser.findElements(By.id("searchBtn2")).size() <= 0 && !browser.findElement(By.id("searchBtn2")).isDisplayed()) {
        WebDriverWait wait2 = new WebDriverWait(browser, timeOutInSeconds: 120);
        wait2.until(ExpectedConditions.visibilityOf(browser.findElement(By.id("searchBtn2"))));
        browser.findElement(By.id("latitude")).click();
        browser.findElement(By.id("latitude")).sendKeys(...charSequences: "40.64427");
        browser.findElement(By.id("longitude")).click();
        browser.findElement(By.id("longitude")).sendKeys(...charSequences: "-8.64554");
        browser.findElement(By.id("searchBtn2")).click();
        assertEquals( expected: "Name: \Aveiro\\"", browser.findElement(By.id("name2")).getText());
    }
} catch (StaleElementReferenceException e) {
    return;
}
```

Figure 18 - Fixed error "id not found"

3.4 Static code analysis

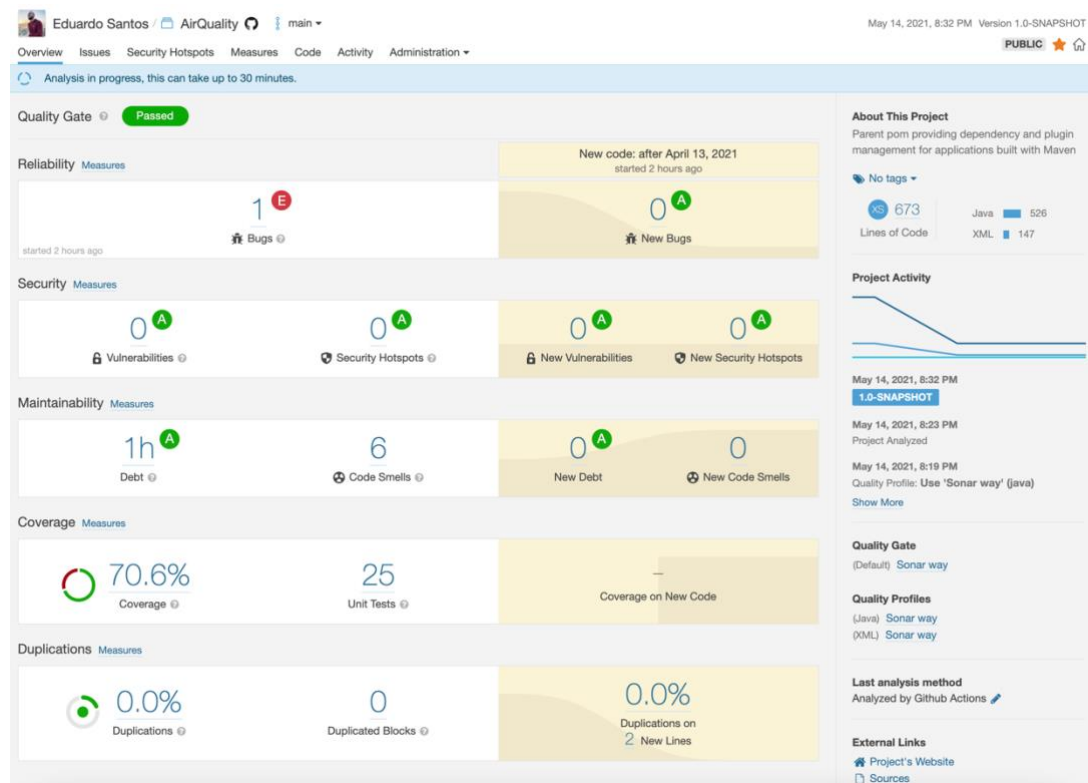


Figure 19 - SonarQube's static code analysis dashboard for this project

For the static code analysis of the project, **SonarQube** was used. This allowed us to check multiple aspects of the project:

- **Reliability:**
 - Bugs
- **Security:**
 - Vulnerabilities
 - Security Hotspots
- **Maintainability:**
 - Debt (estimated time it will take to fix all code smells)
 - Code Smells
- **Coverage:**
 - Coverage %
 - Unit Tests
- **Duplications:**
 - Duplications %
 - Duplicated Blocks

We have also the **Quality Gate**:

“A Quality Gate is a set of measure-based Boolean conditions. It helps you know immediately whether your project is production-ready. If your current status is not Passed, you'll see which measures caused the problem and the values required to pass.”

As we can see, the Quality Gate status for this project is “Passed”, meaning that this is ready for production, and could be sold as the final product.

We can notice that there is a level “E” bug, as well as some code smells. What happens is that, sometimes, there are some bad analyses on SonarQube, showing us wrongly diagnosed errors.

3.5 Continuous integration pipeline

A **CI** (Continuous Integration) **pipeline** was implemented, using **Github Actions**. We established 2 jobs:

- Build the Maven project, verifying if the build is successful and if the `.jar` file is created.
- Run the SonarCloud's code inspection scan, automatically integrating the scan on SonarCloud's dashboard.

```
72 lines (62 sloc) | 2.26 KB
1 name: CI Script
2
3 # Controls when the action will run.
4 on:
5   # Triggers the workflow on push or pull request events but only for the main branch
6   push:
7     branches: [ main ]
8
9   # Allows you to run this workflow manually from the Actions tab
10  workflow_dispatch:
11
12 # A workflow run is made up of one or more jobs that can run sequentially or in parallel
13 jobs:
14   # This workflow contains a single job called "build"
15   build:
16     name: Build Maven Project
17     # The type of runner that the job will run on
18     runs-on: ubuntu-latest
19
20     # Steps represent a sequence of tasks that will be executed as part of the job
21     steps:
22       # Checks-out your repository under $GITHUB_WORKSPACE, so your job can access it
23       - name: Step 1 - Checkout main branch from Github
24         uses: actions/checkout@v2
25
26       - name: Step 2 - Set up JDK 1.8
27         uses: actions/setup-java@v1
28         with:
29           java-version: 1.8
30
31       - name: Step 3 - Have Github Actions build Maven project
32         run: mvn -B package --file AirQuality/pom.xml
33
34       - name: Step 4 - List the current directory
35         run: ls -la
36
37       - name: Step 5 - Show files inside the target/ folder
38         run: |
39           cd AirQuality/target
40           ls -la
41
42   sonar:
43     name: SonarCloud Code Inspection
44     runs-on: ubuntu-latest
45     steps:
46       - uses: actions/checkout@v2
47       with:
48         fetch-depth: 0 # Shallow clones should be disabled for a better relevancy of analysis
49       - name: Set up JDK 11
50         uses: actions/setup-java@v1
51         with:
52           java-version: 11
53       - name: Cache SonarCloud packages
54         uses: actions/cache@v1
55         with:
56           path: ~/.sonar/cache
57           key: ${ runner.os }-sonar
58           restore-keys: ${ runner.os }-sonar
59       - name: Cache Maven packages
60         uses: actions/cache@v1
61         with:
62           path: ~/.m2
63           key: ${ runner.os }-m2-${ hashFiles('**/pom.xml') }
64           restore-keys: ${ runner.os }-m2
65       - name: Build and analyze
66         env:
67           GITHUB_TOKEN: ${ secrets.GITHUB_TOKEN } # Needed to get PR information, if any
68           SONAR_TOKEN: ${ secrets.SONAR_TOKEN }
69         run: |
70           cd AirQuality
71           mvn -B verify org.sonarsource.scanner.maven:sonar-maven-plugin:sonar
```

Figure 20 - `main.yml` file, contains script for the CI pipeline

Workflows [New workflow](#)

All workflows

CI Script

main.yml

Filter workflow runs

26 workflow runs

Event Status Branch Actor

This workflow has a workflow_dispatch event trigger. [Run workflow](#)

✓	Added SonarQube dashboard link	main	1 hour ago	...
CI Script #26: Commit 7a493ad pushed by eduardosantoshf				
✓	Minor change to README	main	2 hours ago	...
CI Script #25: Commit ea27669 pushed by eduardosantoshf				
✓	Changed SonarQube organisation	main	2 hours ago	...
CI Script #24: Commit 2d9e0af pushed by eduardosantoshf				
✗	Merge branch 'main' of https://github.com/eduardos...	main	2 hours ago	...
CI Script #23: Commit 2873b9a pushed by eduardosantoshf				
✗	Merge pull request #1 from eduardosantoshf/depend...	main	2 hours ago	...
CI Script #22: Commit 713aad8 pushed by eduardosantoshf				
✓	Added SonarQube badge to README	main	2 hours ago	...
CI Script #21: Commit b59858f pushed by eduardosantoshf				
✓	Merge branch 'main' of https://github.com/eduardos...	main	2 hours ago	...
CI Script #20: Commit ca84cd3 pushed by eduardosantoshf				

Figure 22 - Example of some CI tests

Search or jump to... Pull requests Issues Marketplace Explore

eduardosantoshf / AirQuality

Unwatch 1 Star 0 Fork 0

Code Issues Pull requests **Actions** Projects Wiki Security Insights Settings

✓ Merge branch 'main' of https://github.com/eduardosantoshf/TQSIndividu... CI Script #20 [Re-run jobs](#) [...](#)

Triggered via push 2 hours ago

eduardosantoshf pushed → ca84cd3 main

Status **Success** Total duration 2m 21s Artifacts -

main.yml on: push

- ✓ Build Maven Project 1m 5s
- ✓ SonarCloud Code Insp... 2m 10s

Figure 21 - An example of a succeeded CI test

4 References & resources

Project resources

- Project's source code repository: <https://github.com/eduardosantoshf/AirQuality>
- Video demo – <https://github.com/eduardosantoshf/AirQuality>
- QA dashboard: https://sonarcloud.io/dashboard?id=eduardosantoshf_AirQuality

Reference materials

WeatherBit's API: <https://www.weatherbit.io/api/airquality-current>