HW1: Mid-term assignment report

Fábio Alexandre Andrade Carmelino [93406], 2021-05-14

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 chosen **theme** for this it was '**Air Quality**'.

Through this application, it is possible to search for the air quality index, the quantity of some components present in the air and also, the latitude and longitude of every city available in the external source of information.

1.2 Current limitations

All intended functionalities were implemented, so there are no limitations.

2 Product specification

2.1 Functional scope and supported interactions

Whenever someone wants to find out about the air quality of a city, they can do so through this API. The different possible ways of searching are the following:

- **Coordinates** (Latitude and Longitude)
- **Name of a city**

Both of the above **can also be accompanied by a date in 'yyyy-mm-dd' format**. In case that no date is specified, the results will be for the current day. Otherwise, they will for the requested day.
Lastly, there is an endpoint to get statistics on the total of requests received by the API together with the hits and misses over the in-memory-cache.

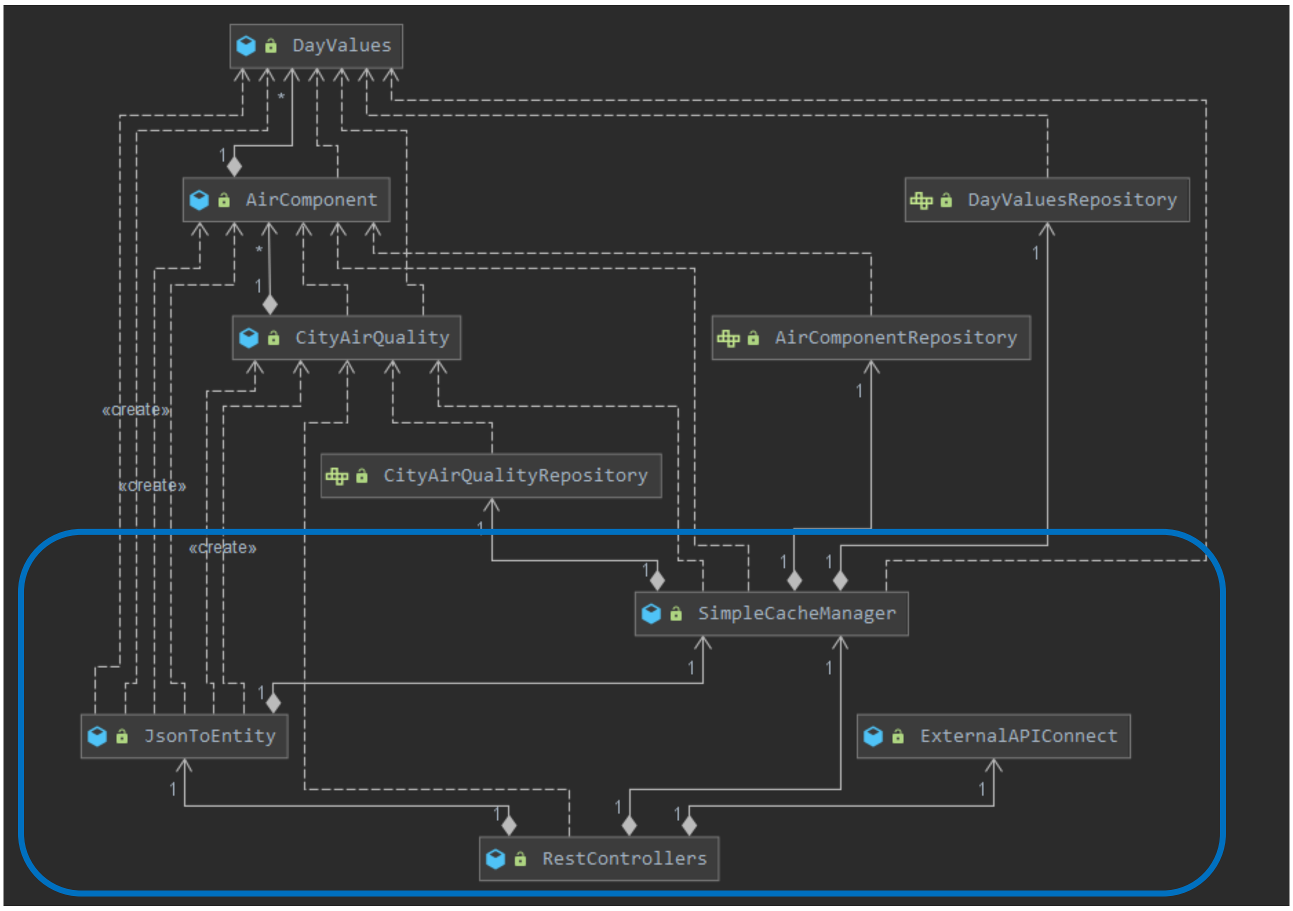
2.2 System architecture

To implement this application the following technologies were used:

- Spring Boot **Rest Controllers** for the API
- **Entities** and **JPA Repositories** for the in-memory-cache using **H2-database**

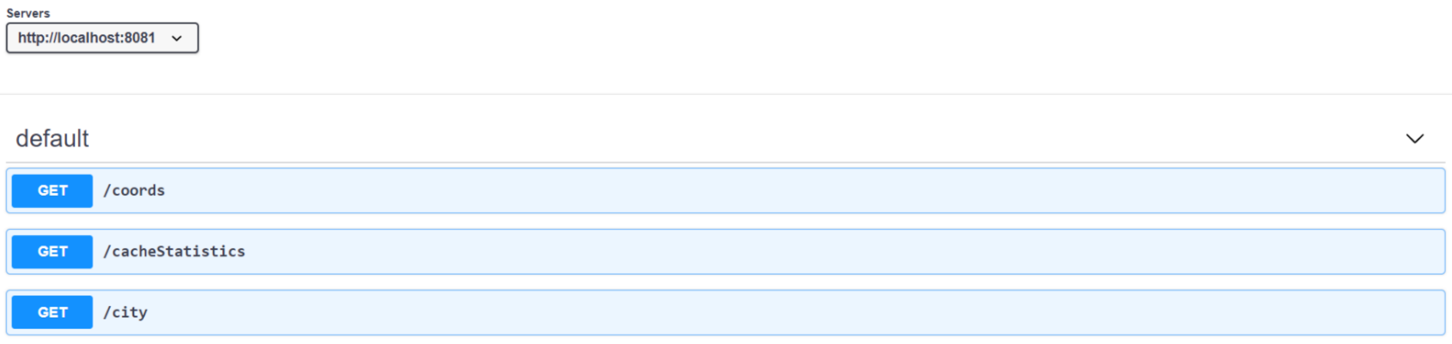
Whenever a request is made, it is received by the Rest Controllers.
When it is a request for the statistics of the cache, it will be answered immediately after the controller asks the cache for those values.
Given that it is a request for an Air Quality Search, the controllers will ask the cache if it has the answer for it saved. This cache interacts with 3 repositories ('**CityAirQualityRepository**', '**AirComponentRepository**' and '**DayValuesRepository**'), and each of these repositories manages its respective type of objects (entity). The entity managed by the first one, between other information such as the name, coordinates, and air quality index (aqi) has a list of 'AirComponent' (entity managed by the second repository). And the 'AirComponent' has a Map whose keys are the current day and each of the following 5 days and the values are the minimum, maximum and average values of the component. Both the 'AirComponent' and 'DayValues' entities were implemented for the sake of saving a forecast of the air components values.
After receiving an answer from the **SimpleCacheManager**, in case it is affirmative, the controller will respond with the cached information, otherwise, it will ask the '**ExternalAPIConnect**' component to perform a request to the external API.
Finally, after receiving an answer from the external source there are only 2 options left. If the requested city exists, the controller will ask the '**JsonToEntity**' component to process the answer and pass it to the cache to be saved. If the city does not exist, an answer informing so will be returned.

Diagram



2.3 API Endpoints

- /city?city_name&date
- /coords?latitude&longitude&date
- /cacheStatistics



API Documentation Link: https://app.swaggerhub.com/apis-docs/FabioSparta/AirQualityAPI_done/0.1#/

3 Quality assurance

3.1 Overall strategy for testing

The strategy used was to first implement the features and then write the tests to verify if they were performing as intended.
The tools used were:

- **Rest-Assured** for API's integration test
- **MockMvc** for the API's unit test
- Solely **JUnit** and **Hamcrest** for the Cache and JsonToEntity components test
- **Selenium + Cucumber** for the interface tests

3.2 Unit and integration testing

Unit Tests performed over the API: (*File: RestControllerUnitTest)*

- whenNoCacheButCityExists_thenUseExternalAPIAndJsonToEntity__AndStatus200()
- whenCacheAndCityExists_thenUseCache_AndStatus200()
- whenNonExistentCity_thenStatus404()
- whenValidDateFormatAndCityExists_thenStatus200()
- whenDateInvalidFormat_thenStatus400()
- whenNoResponseFromExternalAPI_thenStatus503()

- whenNoCacheButCoords_thenUseExternalAPIAndJsonToEntity__AndStatus200()
- whenCacheAndCoords_thenUseCache_AndStatus200()
- whenValidDateFormatAndCoords_thenStatus200()
- whenCoordsInvalid_thenStatus400()
- whenDateInvalid2_thenStatus400()
- whenNoResponseFromExternalAPI2_thenStatus503()

For these tests all possibilities of answers that may result from the parameters state (if the parameters a valid or not - 'city' and 'date' for the first rest-controller and 'latitude', 'longitude' and 'date' for the rest-controller) together with the different answers that may come from the SimpleCacheManager and ExternalAPIConnect were considered.

Integration Tests Performed over the API: (*File: RestControllerIT)*

- whenCityNameExists_thenReturnCityTodayForecast()
- whenNonExistentCity_thenReturnNotFound()
- whenCityNameExistsAndDateExists_thenReturnCityDateForecast()
- whenDateInvalidFormat_thenReturnBadRequest()

- whenCityLatitudeLongitude_thenReturnClosestCityTodayForecast()
- whenCoordsInvalid_thenReturnBadRequest()
- whenCoordsAndDateExists_thenReturnCityDateForecast()
- whenDateInvalid2_thenStatus400()

For these tests, the focus is to evaluate the response of the controller. "Positive" responses should be returned when the cities, dates, and coordinates are correct. And "Negative" responses in any other situations. 'New York' was used as subject of test. Given its name or coordinates, 'New York' should be present in the answer and the 'date' should also be the one requested or the current date if none was specified.

Unit Test Performed over JsonToEntity Component: (*File: JsonToEntityUnitTest)*

- whenTransformIsCalled_thenCAQCreatedAsExpected()

A single test was performed to verify if given a json String, the object created has every value saved correctly in the correct attributes.

Unit Test Performed over the Cache Component: (*File: SimpleCacheManagerTest)*

- whenSavedCityQuery_thenCityQueryShouldExist()
- whenSavedCityName_thenCityNameShouldExist()
- whenSaved2Cities_then2CitiesShouldExist()
- whenSavedCityLatitudeLongitude_thenCityShouldExist_With_0_25ErrorTolerance()
- whenCitiesExist_thenCacheHitsIncremented()
- whenCitiesDontExist_thenCacheMissesIncremented()
- whenLastAccessTooOld_thenCityShouldHaveBeenDeleted()

Focused on verifying if the cache is doing the operations of Saving / Deleting / Finding objects correctly. The deletion of objects is done by a thread that is running on a fixed interval by comparing the timestamps of creation and last access of the cached objects with the defined limits in the variables 'LAST_ACESS' and 'EXPIRATION'. Any cached object whose last access or creation is too "old", will be deleted.

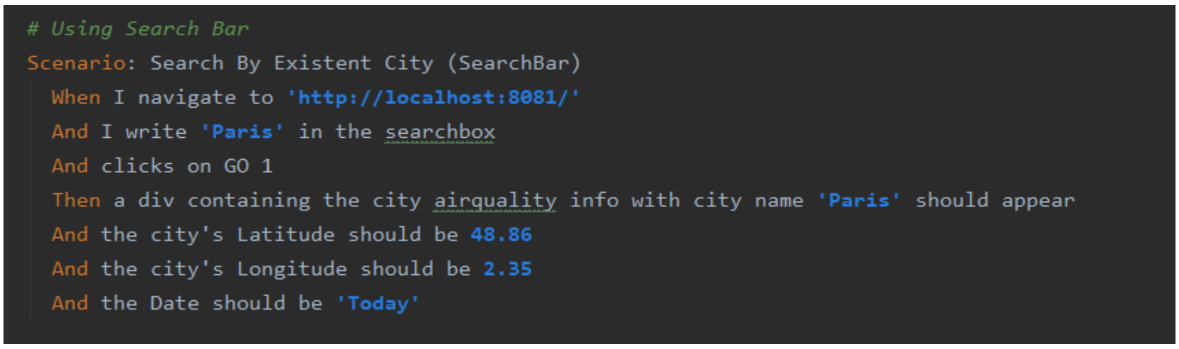
3.3 Functional testing

Tests performed for over the App's Interface:

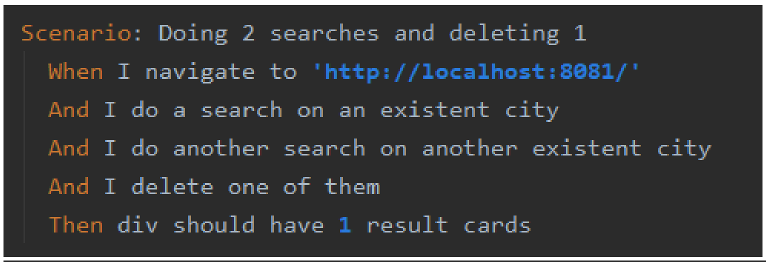
There are 3 different possible ways of searching by using the interface of this application. To use the search bar and write any name the user wants, to use the select box and choose one of the few options statically available or to write coordinates and receive the answer for the city closest to them. All of these options may be complimented with a date that can be select from 5 buttons with the current date and the next 4 days.

Tests were created considering the possibilities described above. Furthermore, since every time a search is done, the answer is dynamically presented via ajax in a new card and cards can be deleted by pressing on an 'x' at their top-right corner, that behavior was also tested

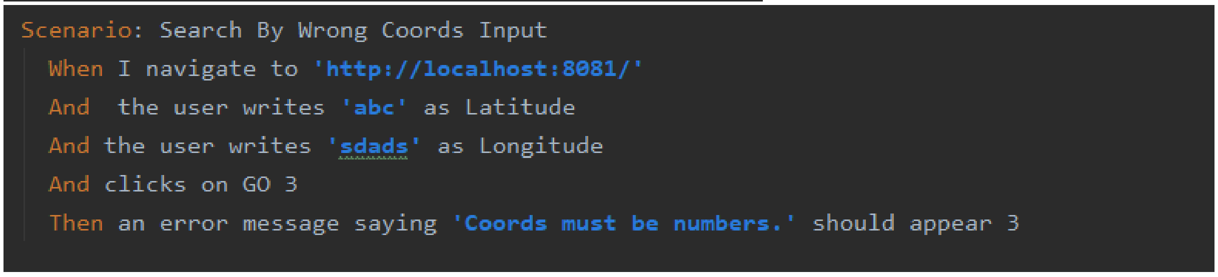
Examples:



```
# Using Search Bar
Scenario: Search By Existent City (SearchBar)
  When I navigate to 'http://localhost:8081/'
  And I write 'Paris' in the searchbox
  And clicks on GO 1
  Then a div containing the city airquality info with city name 'Paris' should appear
  And the city's Latitude should be 48.86
  And the city's Longitude should be 2.35
  And the Date should be 'Today'
```

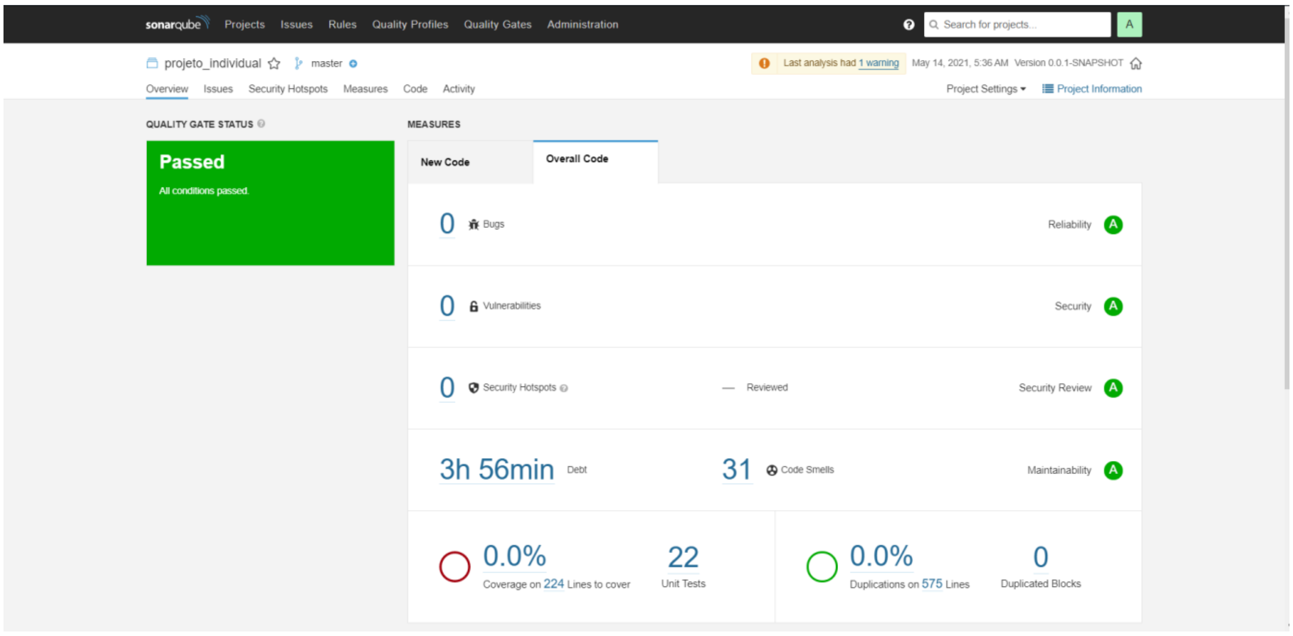


```
Scenario: Doing 2 searches and deleting 1
  When I navigate to 'http://localhost:8081/'
  And I do a search on an existent city
  And I do another search on another existent city
  And I delete one of them
  Then div should have 1 result cards
```



```
Scenario: Search By Wrong Coords Input
  When I navigate to 'http://localhost:8081/'
  And  the user writes 'abc' as Latitude
  And the user writes 'sdads' as Longitude
  And clicks on GO 3
  Then an error message saying 'Coords must be numbers.' should appear 3
```

3.4 Sonar Qube



QUALITY GATE STATUS

Passed

All conditions passed.

MEASURES

New Code | Overall Code

0 Bugs — Reliability A

0 Vulnerabilities — Security A

0 Security Hotspots — Reviewed — Security Review A

3h 56min Debt — 31 Code Smells — Maintainability A

0.0% Coverage on 224 Lines to cover — 22 Unit Tests

0.0% Duplications on 575 Lines — 0 Duplicated Blocks

Most of the Code Smells are for minor reasons like opting to name a variable as 'CAQRepository' instead of the recommended practice that would be 'CaqRepository'. Also, it says that the coverage is 0.0%, something that I could not find the reason for.

4 References & resources

Project resources

- Video demo and project code is in the Git Repository: https://github.com/FabioSparta/TQS_IndividualProj