Birzeit University

Master in Software Engineering

Service-Oriented Software Engineering (SWEN 6307)

Term Project

Service Composition

Instructor: Dr. Nariman Ammar

Prepared by:

Ihab Mishriki

Subhi Mara'beh

Saadeddin Hroub

Date: 31\05\2017

Abstract

Service composition is aggregation of two or more services, where small services together are combined to form bigger service. The service initiator is needed to initiate the composition, else, the services represent only a point-to-point exchange[1]. This report shows service composition between web services from two teams: our team (team 7) was the service provider for team 4, on the same time we were there client. Our service is “weather, Exchange, Map”, while the other team service is “Directory for Academic programs”.

Introduction:

Services composition is similar to components composition, services are communicating together via a network, but the inter-service communication is slower comparing to inter-component communication within the same application. Using enterprise service bus reduces the performance. Service compositions are classified to simple logic for point to point exchanges, while complex composition becomes widely used due to the development in technology.

Our project is using 3 services, the first service is google Map, the second service is currency exchange, and the last is weather. The idea of our project is to composite weather and currency exchange services with google map, so when click on any country on the google map, the application shows the weather and the currency of that country.

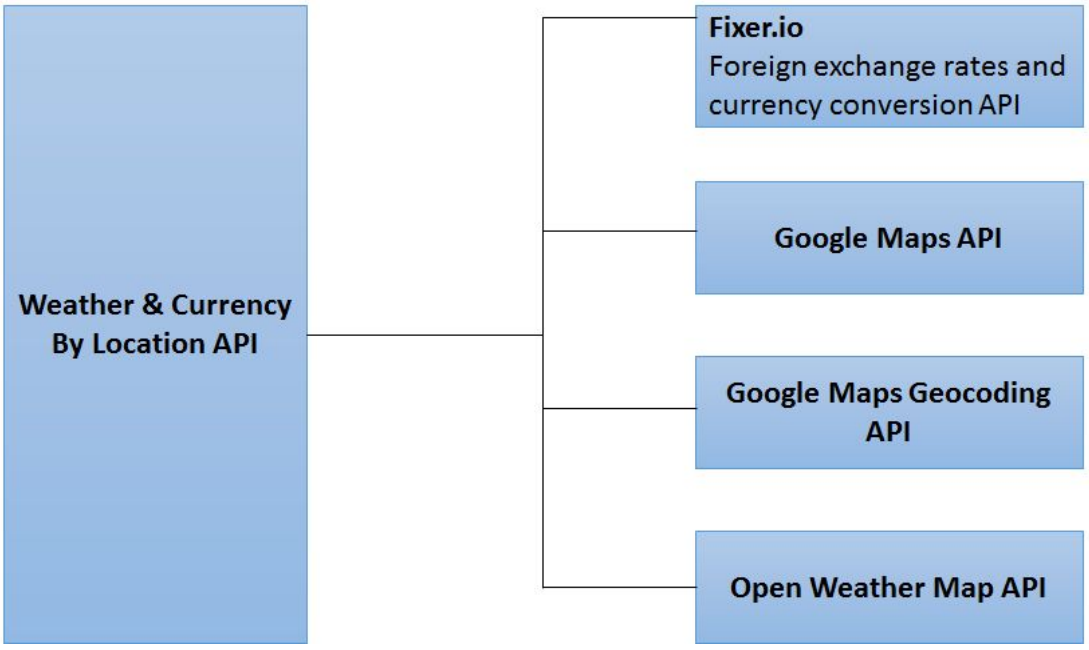


Fig. (1): Service composition diagram.

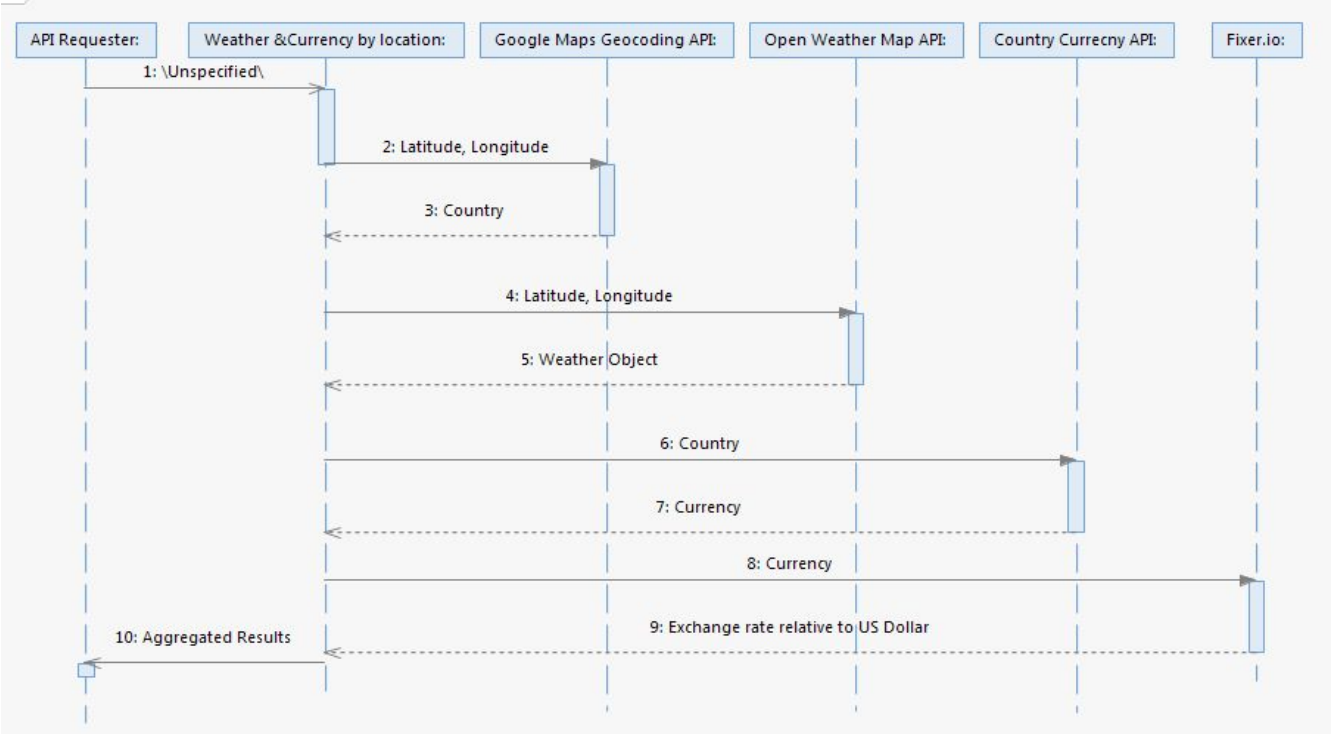


Fig. (2): Sequence diagram for the composite service.

Our API accepts URL parameters and returns JSON objects

latest.json

```
1  {"base":"USD","date":"2017-04-27","rates":{"GBP":0.77585,"ILS":3.644}}
```

Fixer.io: Foreign exchange rates and currency conversion API

it is an API for currency conversion and exchange rates, the availability of this API is 99.96%, it has throughput equals to 8160 rpm while the response time is 7.45 ms. Fixer.io which is a free JSON API, offers historical exchange rates, these rates were published by the European Central Bank, where the rates update is performed daily on 4PM central european time. [3]

API sample input:
http://api.fixer.io/latest?symbols=ILS,GBP&base=USD

API sample output:
{"base":"USD","date":"2017-04-28","rates":{"GBP":0.77285,"ILS":3.622}}

Google Maps API:

Google developed it as a web mapping, this service offers satellite imagery, street views, traffic, routes, … etc, the beginning of Google Maps was a C++ program for Desktop, then in 2005 it was published as Google Maps using XML, AJAX, and JavaScript Technologies. Google Maps has an API so the maps can be embedded on third-party websites, users can participate in expanding and developing the maps through Google Map Maker. In september 2008, Google released Google Maps for mobile, which has the GPS navigation. Google Maps became the top app for smartphones in Aug 2013 as it is used by 54% of smartphone owners [4]

Google Maps Geocoding API:

It is a service offered by Google which gives geocoding of a specific address and its reverse geocoding. Geocoding means convert the location address to geographic coordinates in terms of longitude and latitude, it help in pointing the positions on the map. On the contrary of geocoding, reverse geocoding means converting the coordinates to addresses. Geocoding and reverse geocoding APIs can be accessed using an HTTP interface [5].

API sample input:

latlng — The latitude and longitude values specifying the location for which you wish to obtain the closest, human-readable address.

API sample Output:


```
{
   "results" : [
      {
         "address_components" : [
            {
               "long_name" : "277",
               "short_name" : "277",
               "types" : [ "street_number" ]
            },
            {
               "long_name" : "Bedford Avenue",
               "short_name" : "Bedford Ave",
               "types" : [ "route" ]
            },
            {
               "long_name" : "Williamsburg",
               "short_name" : "Williamsburg",
               "types" : [ "neighborhood", "political" ]
            },
            {
               "long_name" : "Brooklyn",
               "short_name" : "Brooklyn",
```

```
            "types" : [ "sublocality", "political" ]
         },
         {
            "long_name" : "Kings",
            "short_name" : "Kings",
            "types" : [ "administrative_area_level_2", "political" ]
         },
         {
            "long_name" : "New York",
            "short_name" : "NY",
            "types" : [ "administrative_area_level_1", "political" ]
         },
         {
            "long_name" : "United States",
            "short_name" : "US",
            "types" : [ "country", "political" ]
         },
         {
            "long_name" : "11211",
            "short_name" : "11211",
            "types" : [ "postal_code" ]
         }
      ],
      "formatted_address" : "277 Bedford Avenue, Brooklyn, NY 11211, USA",
      "geometry" : {
         "location" : {
            "lat" : 40.714232,
            "lng" : -73.9612889
         },
         "location_type" : "ROOFTOP",
         "viewport" : {
            "northeast" : {
               "lat" : 40.7155809802915,
               "lng" : -73.9599399197085
            },
            "southwest" : {
               "lat" : 40.7128830197085,
               "lng" : -73.96263788029151
            }
         }
      },
      "place_id" : "ChIJd8BlQ2BZwokRAFUEcm_qrcA",
      "types" : [ "street_address" ]
   },

... Additional results[] ...
```


Open Weather Map Service:

Open weather Map is an open source service that offers weather information for more than 200,000 cities in the world, the weather information is updated frequently depending on more than 40,000 weather stations using global models. Weather data is available in XML, JSON, and HTML format.[6]

The current weather data for one location is called using different determinants: the user can call for weather data be city name, or by city name and country code, then API will give the matching results, the matching results may be list of locations due to the similarities in the city or location names, . The second determinant is call be city ID which will give the exact

result only, this call method is recommended because it gives clear and unambiguous results for the city. The other call methods are call by geographic coordinates, call by ZIP code, it also offers getting the weather data for different cities together,

API sample input:

http://samples.openweathermap.org/data/2.5/weather?lat=35&lon=139&appid=b1b15e88fa797225412429c1c50c122a1

API sample output:


```
{
  "coord": {
    "lon": 139.01,
    "lat": 35.02
  },
  "weather": [
    {
      "id": 800,
      "main": "Clear",
      "description": "clear sky",
      "icon": "01n"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 285.514,
    "pressure": 1013.75,
    "humidity": 100,
    "temp_min": 285.514,
    "temp_max": 285.514,
    "sea_level": 1023.22,
    "grnd_level": 1013.75
  },
  "wind": {
    "speed": 5.52,
    "deg": 311
  },
  "clouds": {
    "all": 0
  },
  "dt": 1485792967,
  "sys": {
    "message": 0.0025,
    "country": "JP",
    "sunrise": 1485726240,
    "sunset": 1485763863
  },
  "id": 1907296,
  "name": "Tawarano",
  "cod": 200
}
```

Architecture :

The architecture of the composed services is illustrated in Fig. (3) below:

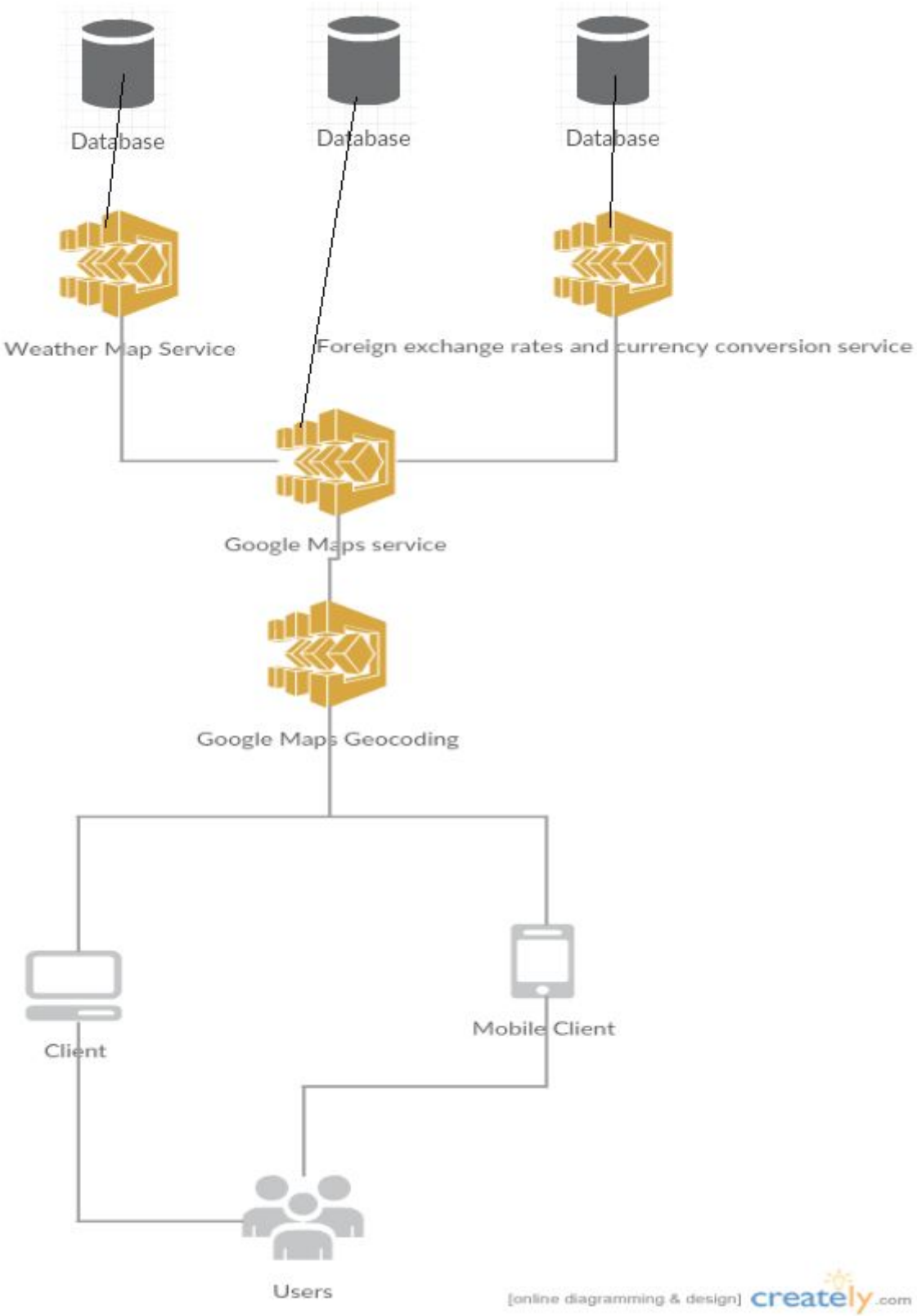


Fig. (3): The architecture of the composite service.

Technology:

We used different technologies to implement our composite service as follows:

Dropwizard:

it is a Java framework which is used for developing high performance and operations friendly RESTful web services. This is achieved as Dropwizard pulls mature and stable Java libraries, and make these libraries in lightweight and simple package which gives the developer the required focus on doing the things. Moreover, Dropwizard supports the configuration tools, logging tools, application metrics, and operational tools. Dropwizard minimizes the time to release high quality web service [7]

Dropwizard is in the middle line between being a library and a framework. The goal of Dropwizard is to provide the web application with accepted performance and reliable implementation. Due to the fact that Dropwizard is being on reusable libraries, the application will be lightweight and focused, which means faster to market with little maintenance product. Drop-wizard is an open source Java framework for the rapid development of RESTful APIs(web services), it glues framework which bundles popular Java libraries and frameworks to ease building new RESTful web services, so it is kind of ecosystem which contains all the dependencies (such as Jersey, Jackson or Jetty) bundled into single package or can be added as separate module.

Without using dropwizard; we will end up Collecting all dependencies, and Class loading issues which results as version mismatch between various java libraries, Dropwizard solves this problem, as it pulls together stable, mature libraries into a simple light-weight package. Below is an incomplete list of some of the libraries that Dropwizard uses: [7]

- Jersey– reference implementation of JAX-RS, the API for RESTful web services in the Java platform
- Jackson – library for processing JSON data
- Jetty – HTTP server and Java Servlet container
- Metrics – library for capturing JVM- and application-level metrics for monitoring
- Guava– utility library
- Logback – logging library

Swagger for Documentation [8]

Usually nobody likes to write documentation, on the other hand REST-based web-services are widely used and documentation for public web-services is needed, there are many documentation technologies used, we selected swagger in our project as it is justified below:

Swagger is an open source framework which helps in design, build, document, and consume RESTful APIs, moreover it is a project used to describe and document RESTful APIs. The Swagger specification defines a set of files required to describe such an API, these files can then be used by the Swagger-UI project to display the API and Swagger-Codegen to generate clients in various languages. On the other hand, an additional utilities can also take advantage of the resulting files, such as testing tools. The files describing the RESTful API in accordance with the Swagger specification are represented as:

- JSON objects and conform to the JSON standards.
- YAML: superset of JSON is used to represent a Swagger specification file.

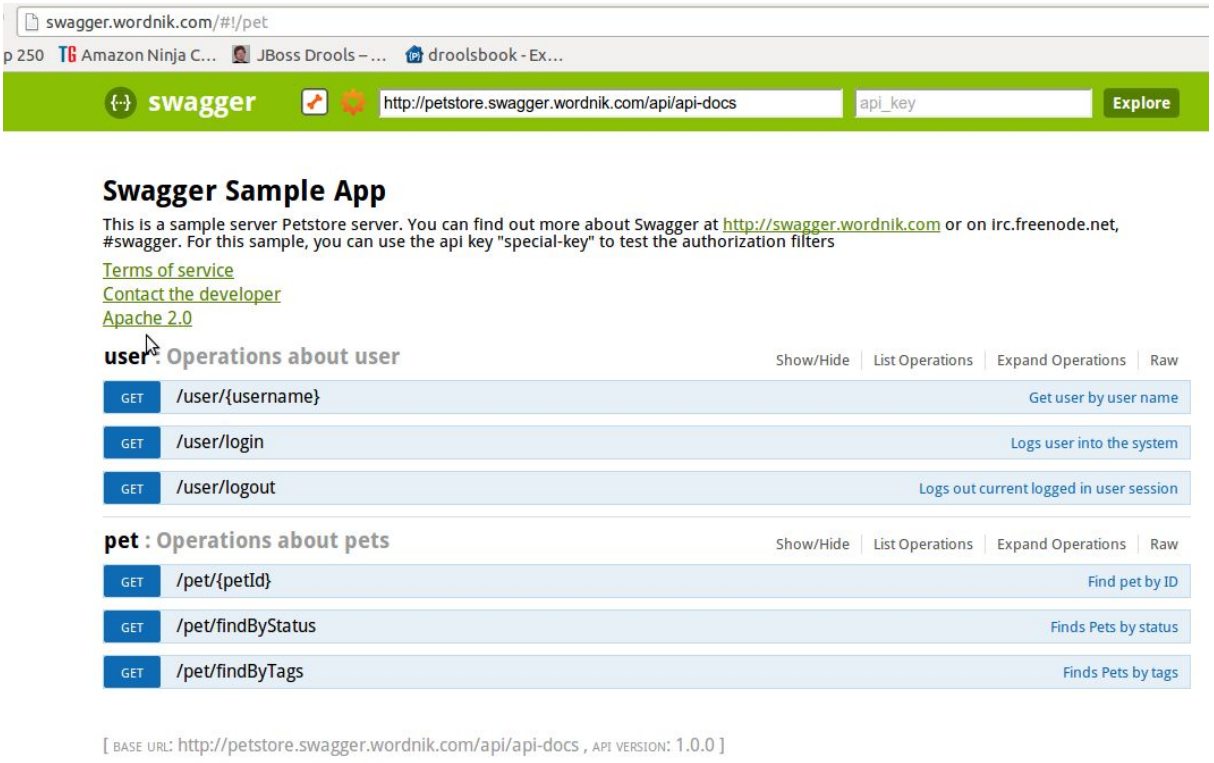


Fig (4). Screenshot for swagger sample applications

Swagger has the following tools [8]:

Swagger Editor Documentation: it is an open source editor to design, define and document RESTful APIs in the Swagger Specification. It has Apache license, the project is open for developers contribution through suggestions, ideas, bug reports and pull requests in the Swagger Editor GitHub repository.

Swagger Codegen Documentation:it is an open source code-generator to build server stubs and client SDKs directly from a Swagger defined RESTful API.

Swagger UI Documentation: This is an open source project to visually render documentation for a Swagger defined API directly from the API's Swagger specification.

Heroku for deployment [9]

Heroku which is used for deployment in our project is a cloud platform as a service, the developers do not care about the deployment infrastructure when using Heroku in their projects, and the main focus is kept on the product itself. Moreover, Heroku has some features are follows:

- The Instant Deployment using Git push: which means build of the application is done using the developer build scripts.
- Heroku has a lot of Add-on resources including applications, databases, ..etc
- Heroku is scalable as it allows performing each single component of the application independently without affecting the performance and functionality of the application. Moreover, the processes in Heroku are separated from each others
- finally , Heroku offer easy access to the logs for each components and process in the application, moreover, it offers 750 computation hours for free to test it.
- Heroku has excellent tutorials which makes it learnable tool, also it has very good performance.

Heroku has many competitors worldwide such as: VMware, Windows Azure, Amazon Web Services, Google App Engine, HP Cloud Services, and OpenShift from Red Hat

Maven [10]

Maven means *accumulator of knowledge*, it was an attempt to make process building easier in Jakarta Turbine project, were many projects with separated Ant build files and JARs that have minor differences. There was a real need to have a standard way to build the projects with clear definition of project contents, and easy to publish its information and share JARs between several projects.So Maven is that magic tool which is used to build and manage any Java-based project. This means Maven makes the Java developer's' job easier by minimizing the development time. Moreover, Maven tries to deal with the following:

- Maven makes the build process easier: using maven does not cancel the necessity of understanding the detailed mechanisms of development, but it hides a lot of low level details.
- It provides uniform build system: it allows the build process of the project to use its POM (project Object Model) with some plugins by all Maven projects. When the developer become familiar with the build is done in Maven, he will become familiar with Maven projects build, which saves the developers time.
- It provides quality information about the software project: Maven gives useful information about the project, some of these data are from the Maven POM, the other is from the project resources. Maven can offers: the change log document which is created directly from the source control, on the other hand, it offers cross referenced sources in addition to the mailing and dependency lists, and finally the unit test

reports with test coverage. This improves the quality of information about the projects and provides better documentation.

- Maven provides best practices in software development: Maven goal is to collect the current best practices for development and ease the guidance of projects in that way. On the other hand, Maven is helpful in projects release management and bug tracking. Moreover, it suggests guidelines to layout the project's structure which makes navigation of other Maven projects easier and faster.
- It supports easy migration to the new features: if helps Maven clients to upgrade their installations to get benefit from any new update on Maven.

Implementation:

We used REST for the implementation of these services.

We were client for Team 4 (Directory for Academic programs), their web service searches for the graduates students using their seat number and their graduation year, we built them client as indicated in the figure below, also we give them chance to check the university based on the students scores. To build this client, we used CSS3, JQuery, HTML5, and many other libraries for JQuery like “loadingoverlay”, “JGrid”.

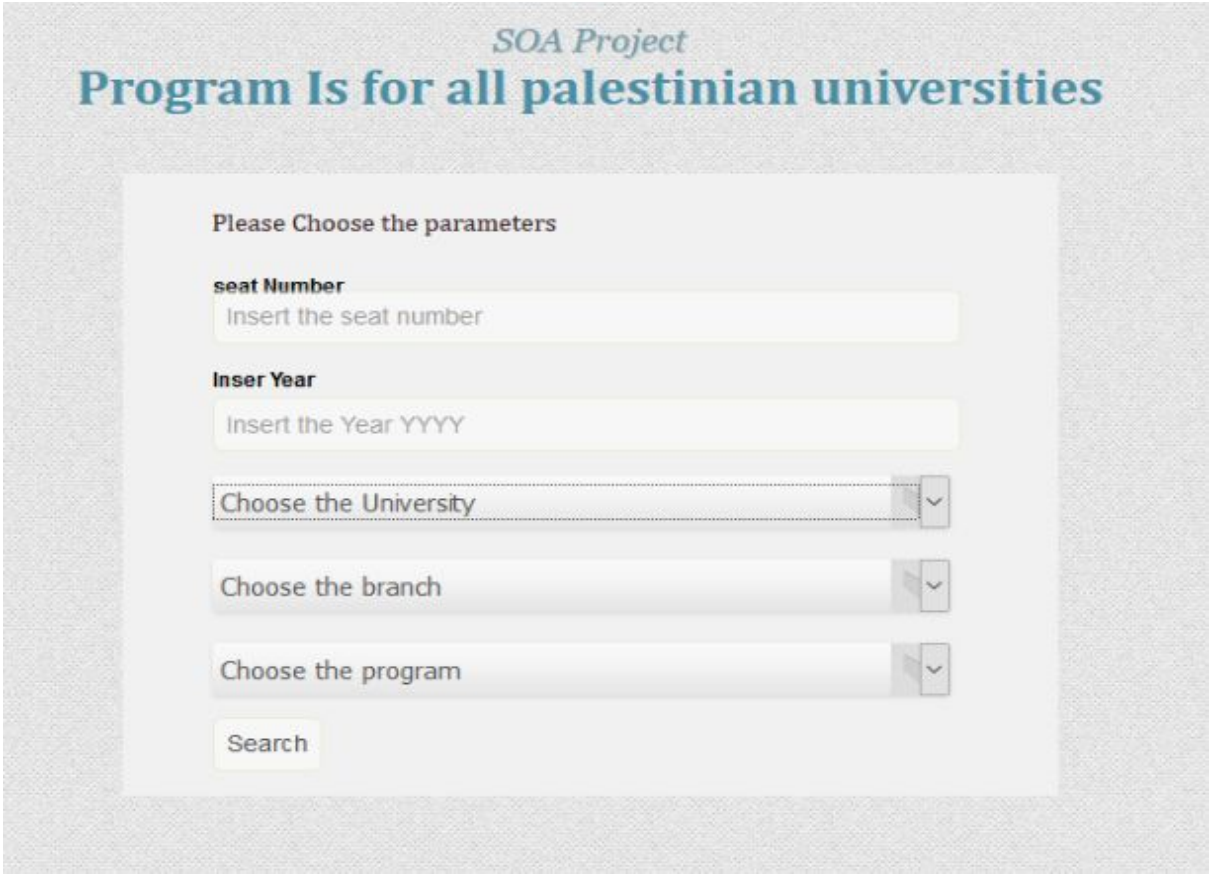


Fig (5). screen shot from our client that searches for graduated students based on year, seat number, university, branch, … etc

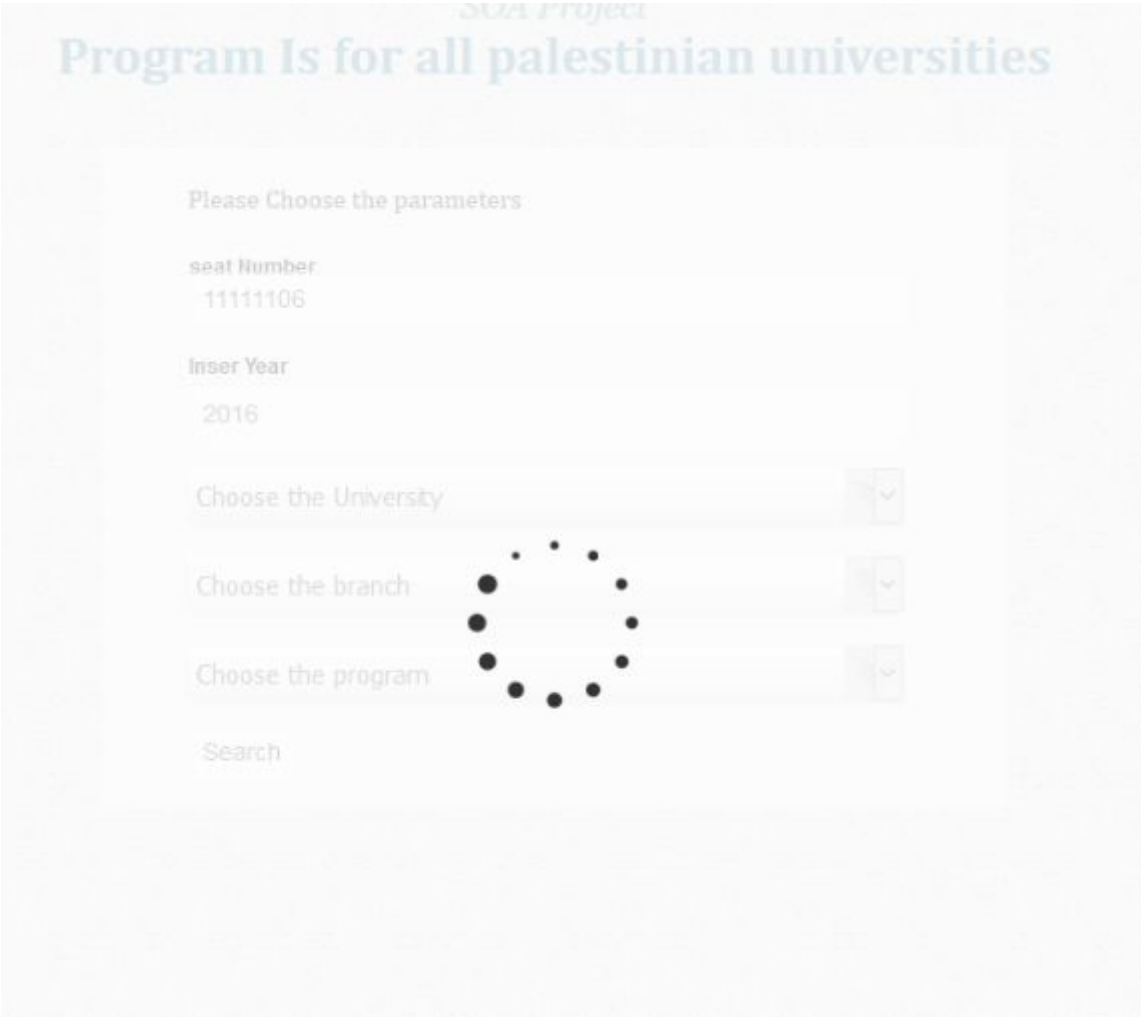


Fig (6). Our client is getting the information from the other team service.

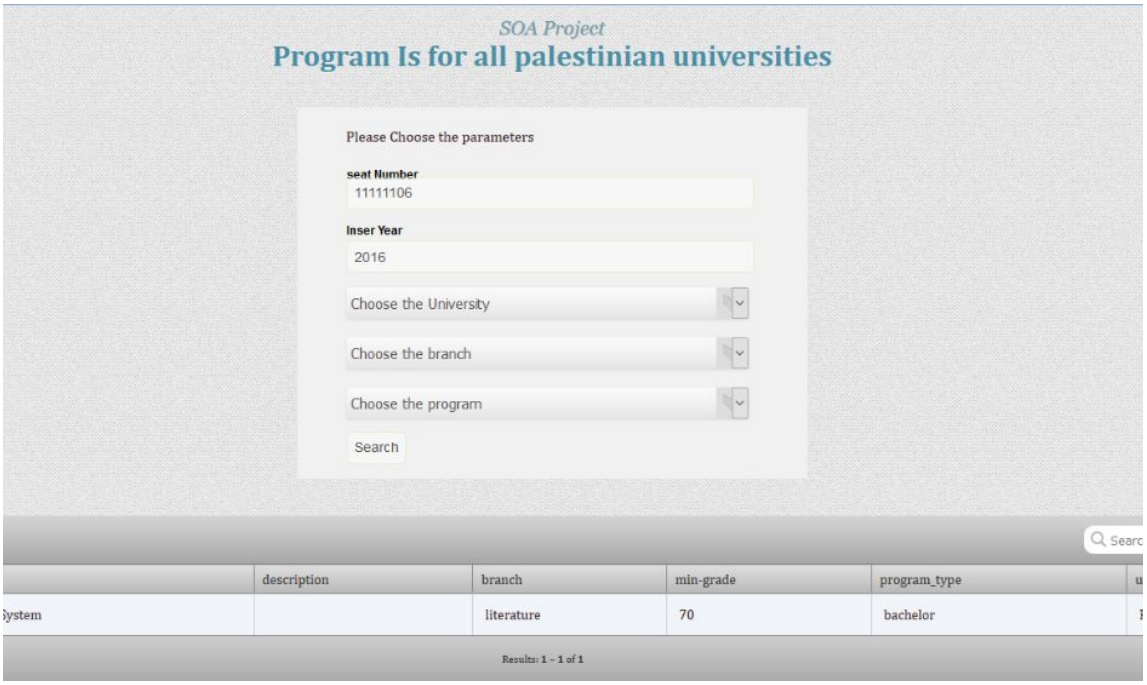


Fig (7). Part of the search operation from our client to the other team service.

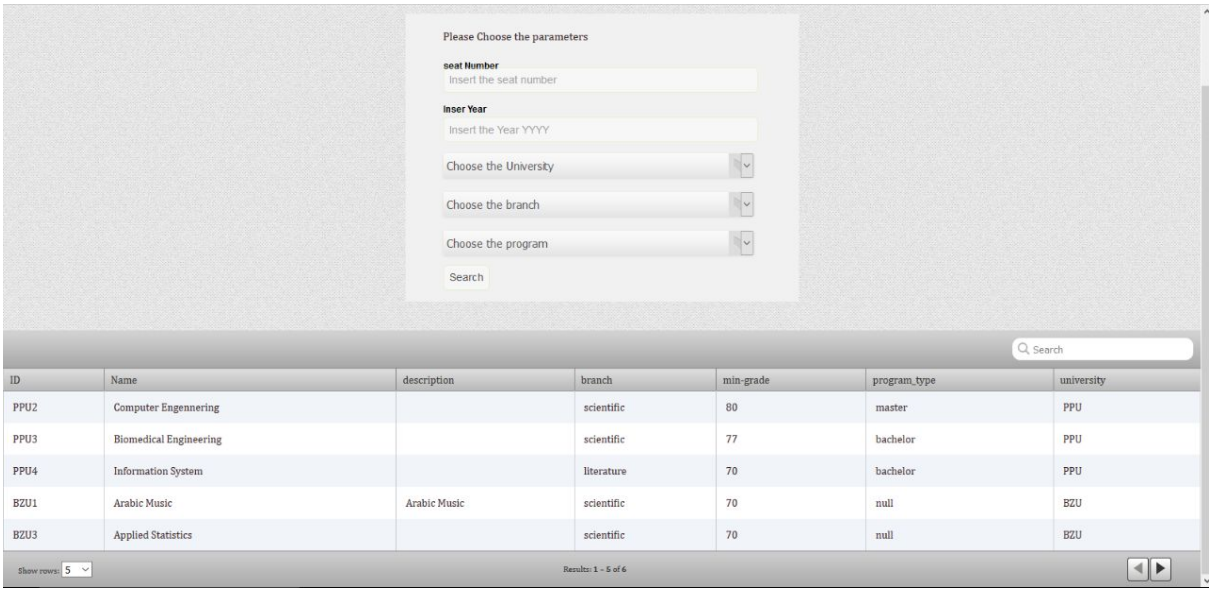


Fig (8). The results of search operation.

Here is the link for our swagger (API documentation for our service)

http://soacourse.herokuapp.com/swagger#!/default/geocode

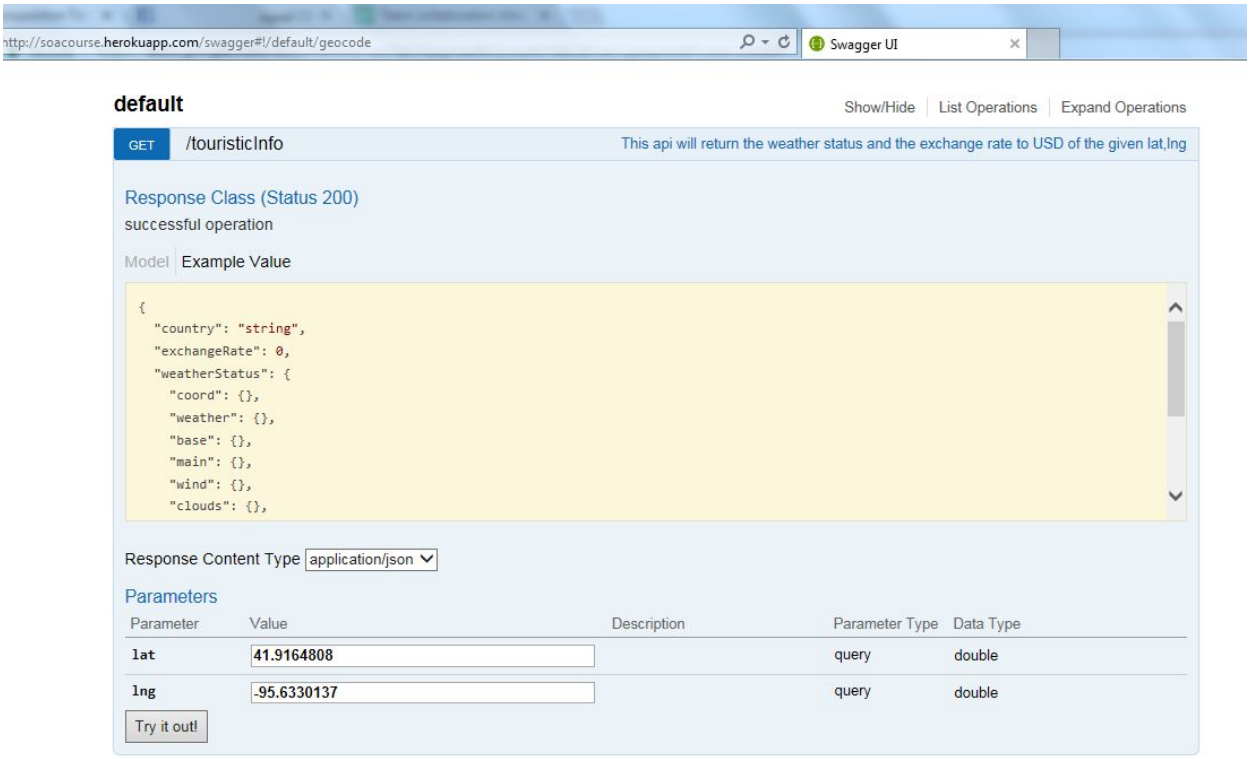


Fig (9). Screenshot from our service that returns the weather status and the exchange rate to USD for a given latitude and longitude

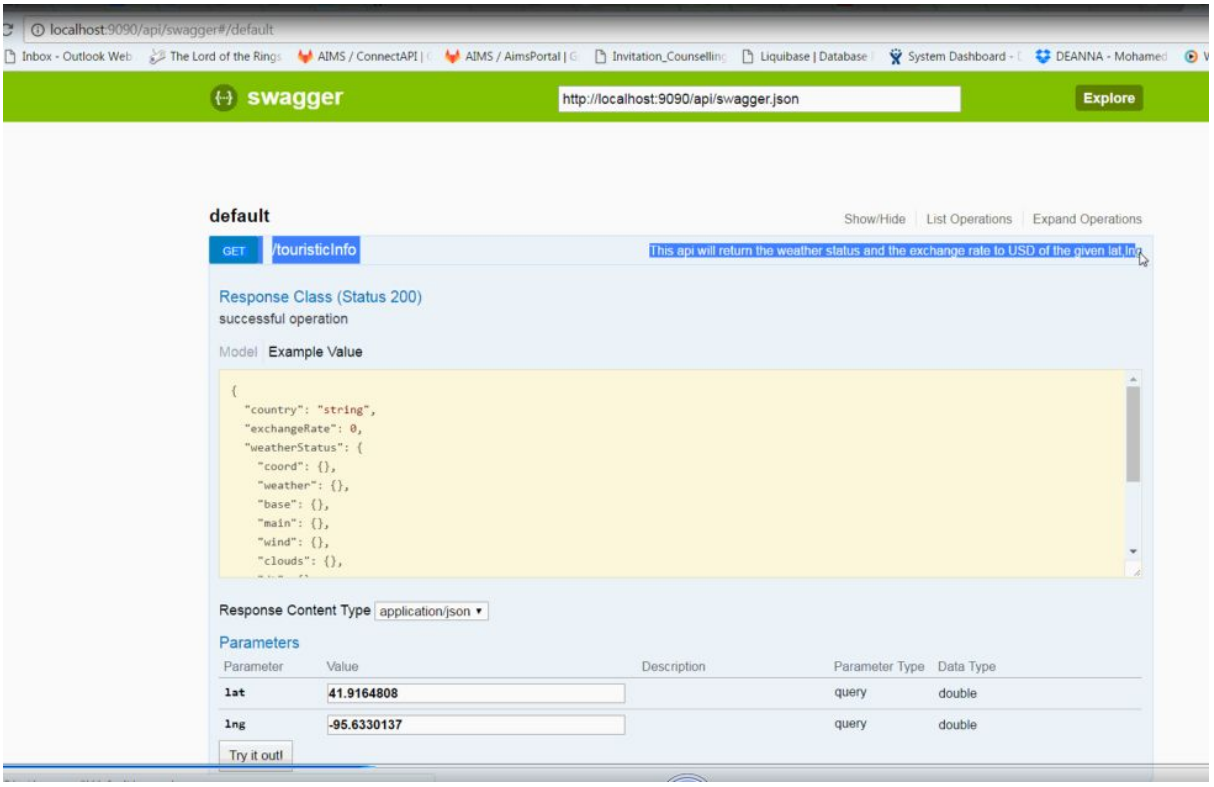


Fig (11). Swagger is used as a documentation tool in our service.

Screenshots from the code:

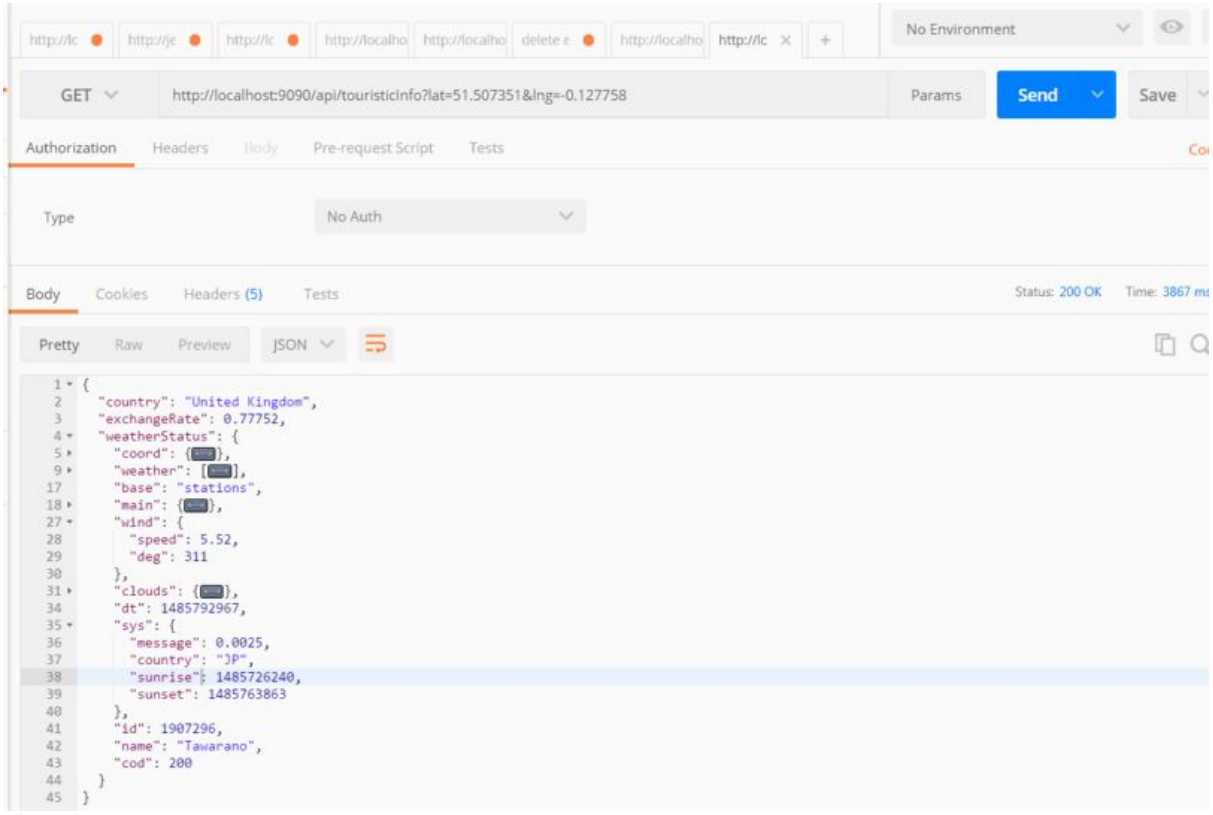
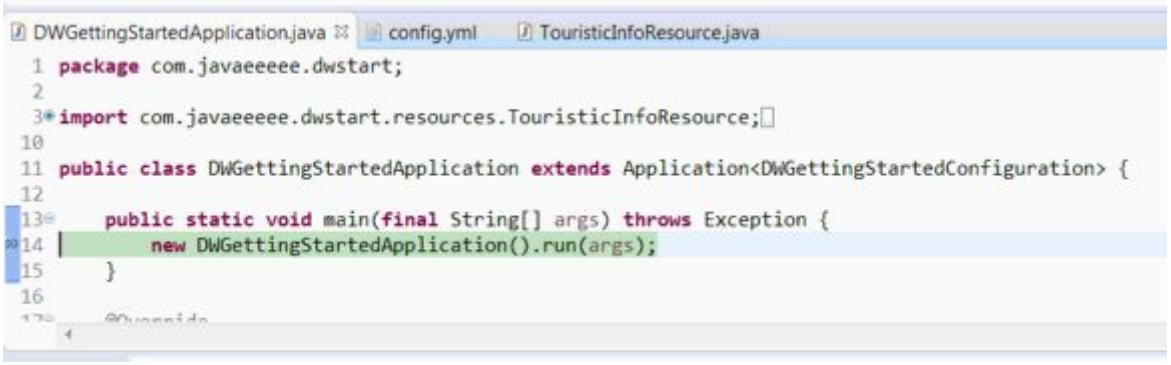
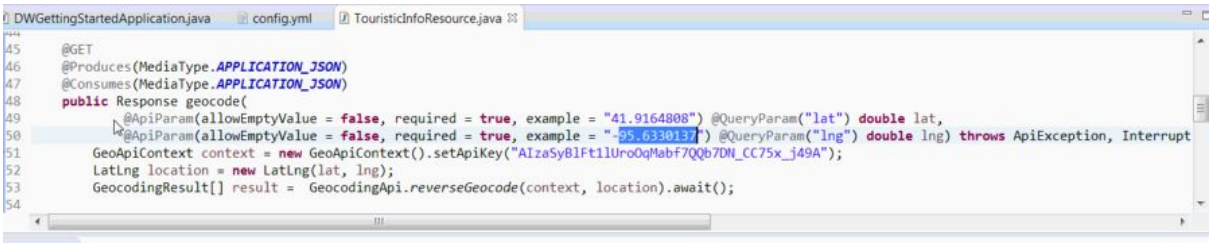
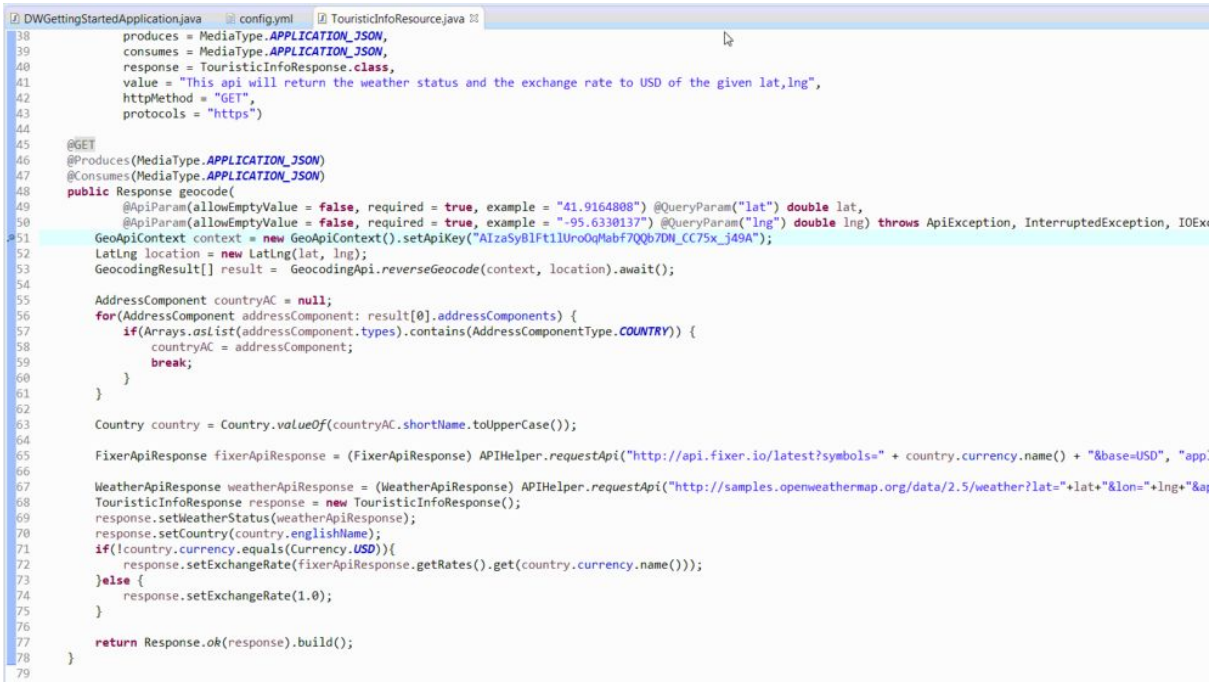


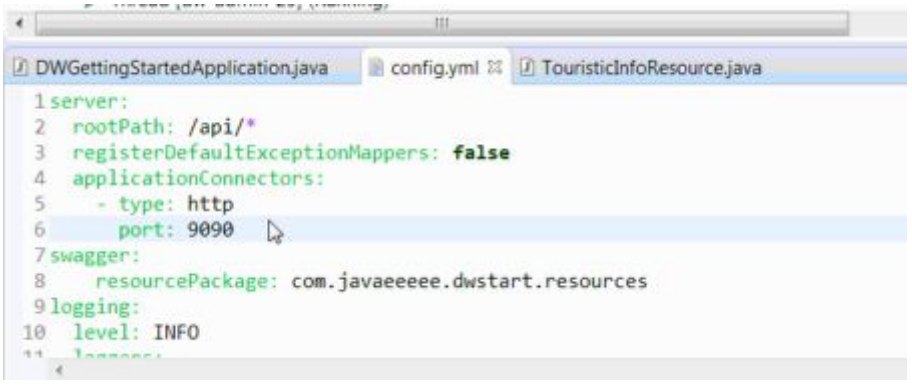
Fig (10). Partial code in our service.







```
DWGettingStartedApplication.java  config.yml  TouristicInfoResource.java
38            produces = MediaType.APPLICATION_JSON,
39            consumes = MediaType.APPLICATION_JSON,
40            response = TouristicInfoResponse.class,
41            value = "This api will return the weather status and the exchange rate to USD of the given lat,lng",
42            httpMethod = "GET",
43            protocols = "https")
44
45    @GET
46    @Produces(MediaType.APPLICATION_JSON)
47    @Consumes(MediaType.APPLICATION_JSON)
48    public Response geocode(
49            @ApiParam(allowEmptyValue = false, required = true, example = "41.9164808") @QueryParam("lat") double lat,
50            @ApiParam(allowEmptyValue = false, required = true, example = "-95.6330137") @QueryParam("lng") double lng) throws ApiException, InterruptedException, IOEx
51        GeoApiContext context = new GeoApiContext().setApiKey("AIzaSyBlFt1lUroOqMabf7QQb7DN_CC75x_j49A");
52        LatLng location = new LatLng(lat, lng);
53        GeocodingResult[] result =  GeocodingApi.reverseGeocode(context, location).await();
54
55        AddressComponent countryAC = null;
56        for(AddressComponent addressComponent: result[0].addressComponents) {
57            if(Arrays.asList(addressComponent.types).contains(AddressComponentType.COUNTRY)) {
58                countryAC = addressComponent;
59                break;
60            }
61        }
62
63        Country country = Country.valueOf(countryAC.shortName.toUpperCase());
64
65        FixerApiResponse fixerApiResponse = (FixerApiResponse) APIHelper.requestApi("http://api.fixer.io/latest?symbols=" + country.currency.name() + "&base=USD", "app
66
67        WeatherApiResponse weatherApiResponse = (WeatherApiResponse) APIHelper.requestApi("http://samples.openweathermap.org/data/2.5/weather?lat="+lat+"&lon="+lng+"&a
68        TouristicInfoResponse response = new TouristicInfoResponse();
69        response.setWeatherStatus(weatherApiResponse);
70        response.setCountry(country.englishName);
71        if(!country.currency.equals(Currency.USD)){
72            response.setExchangeRate(fixerApiResponse.getRates().get(country.currency.name()));
73        }else {
74            response.setExchangeRate(1.0);
75        }
76
77        return Response.ok(response).build();
78    }
79
```



```
DWGettingStartedApplication.java  config.yml  TouristicInfoResource.java
1 server:
2   rootPath: /api/*
3   registerDefaultExceptionMappers: false
4   applicationConnectors:
5     - type: http
6       port: 9090
7 swagger:
8     resourcePackage: com.javaeeeee.dwstart.resources
9 logging:
10   level: INFO
```

```
DWGettingStartedApplication.java  config.yml  TouristicInfoResource.java
1  package com.javaeeeee.dwstart;
2
3  import com.javaeeeee.dwstart.resources.TouristicInfoResource;
10
11 public class DWGettingStartedApplication extends Application<DWGettingStartedConfiguration> {
12
13     public static void main(final String[] args) throws Exception {
14         new DWGettingStartedApplication().run(args);
15     }
16
```

```
DWGettingStartedApplication.java  config.yml  TouristicInfoResource.java
48    public Response geocode(
49            @ApiParam(allowEmptyValue = false, required = true, example = "41.9164808") @QueryParam("lat") double lat,
50            @ApiParam(allowEmptyValue = false, required = true, example = "-95.6330137") @QueryParam("lng") double lng) throws ApiException, Interrupt
51        GeoApiContext context = new GeoApiContext().setApiKey("AIzaSyBlFt1lUroOqMabf7QQb7DN_CC75x_j49A");
52        LatLng location = new LatLng(lat, lng);
53        GeocodingResult[] result =  GeocodingApi.reverseGeocode(context, location).await();
54
55        AddressComponent countryAC = null;
56        for(AddressComponent addressComponent: result[0].addressComponents) {
57            if(Arrays.asList(addressComponent.types).contains(AddressComponentType.COUNTRY)) {
```

Discussion:

Service composition is aggregation of two or more services, where small services together are combined to form bigger service. The service initiator is needed to initiate the composition. We faced some difficulties during creating our client to consume team 4 service, this is because of lack of documentation about the other team services at the beginning, after that we overcome these difficulties

In this project we work with 2 teams, we were client for team 4 while they were service provider. Then we make the opposite by offering them our service and they become a client (service requester), this was clear in the DEMO video which is uploaded to the GitHub. In addition to this, we play the service provider role with team 3 who uses our service.

Future work:

We plan to develop our service in the future, we think that adding the following features to the service will make our project more attractive and accepted by the users. One of the suggested future work is to show the culture of the clicked country, if the user click on the cmap, then map determines the country name, the exchange rate currency for that country, and a brief information about the common culture in that location (country). Another suggested future work is to show the most important tourist places in that country with possibility to divide them into fantasia, historical, relaxation places, . .. etc. moreover, we may give suggestions and advices to the users what to do in that location, and what things that he should avoid to do.

References:

[1] http://serviceorientation.com/soaglossary/service_composition

[2] https://www.tutorialspoint.com/soa/soa_service_composition.htm

[3] http://fixer.io/

[4] https://developers.google.com/maps/web-services/

[5] https://openweathermap.org/current

[6] https://openweathermap.org/current

[7] http://www.dropwizard.io/1.1.0/docs/

[8] http://swagger.io/

[9] https://www.heroku.com/about

[10] https://maven.apache.org/what-is-maven.html