Birzeit University

Master in Software Engineering

Service-Oriented Software Engineering (SWEN 6307)

Assignment 2

Designing a Service Composition Scenario

Instructor: Dr. Nariman Ammar

Prepared by:

Ihab Mishriki

Subhi Mara'beh

Saadeddin Hroub

Date: 29\04\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].

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.

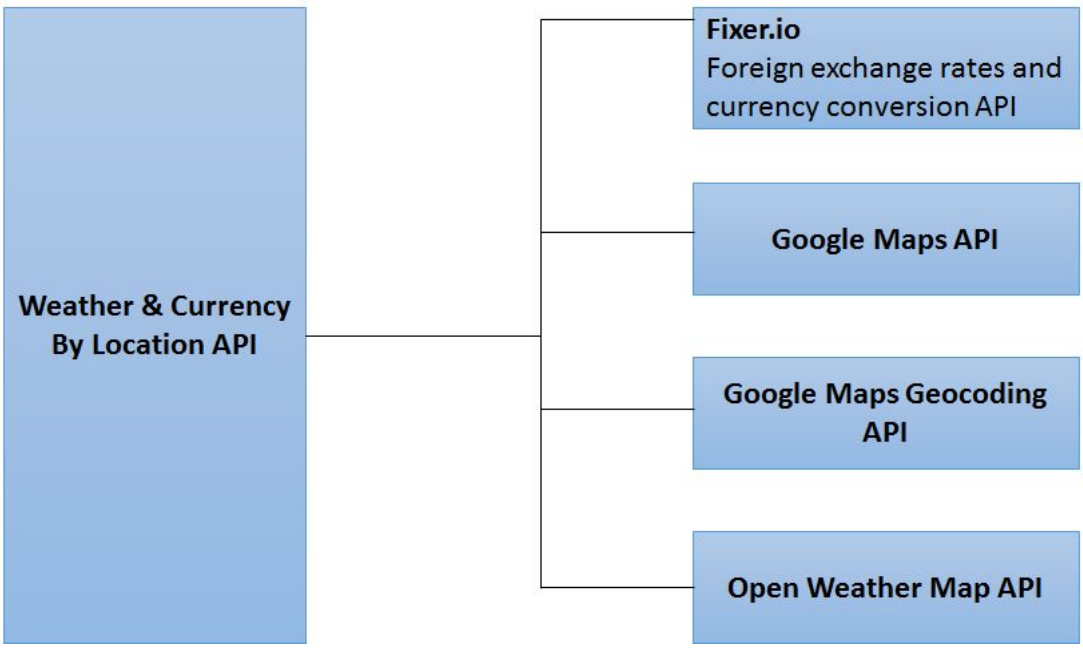

Fig. (1): Service composition diagram.

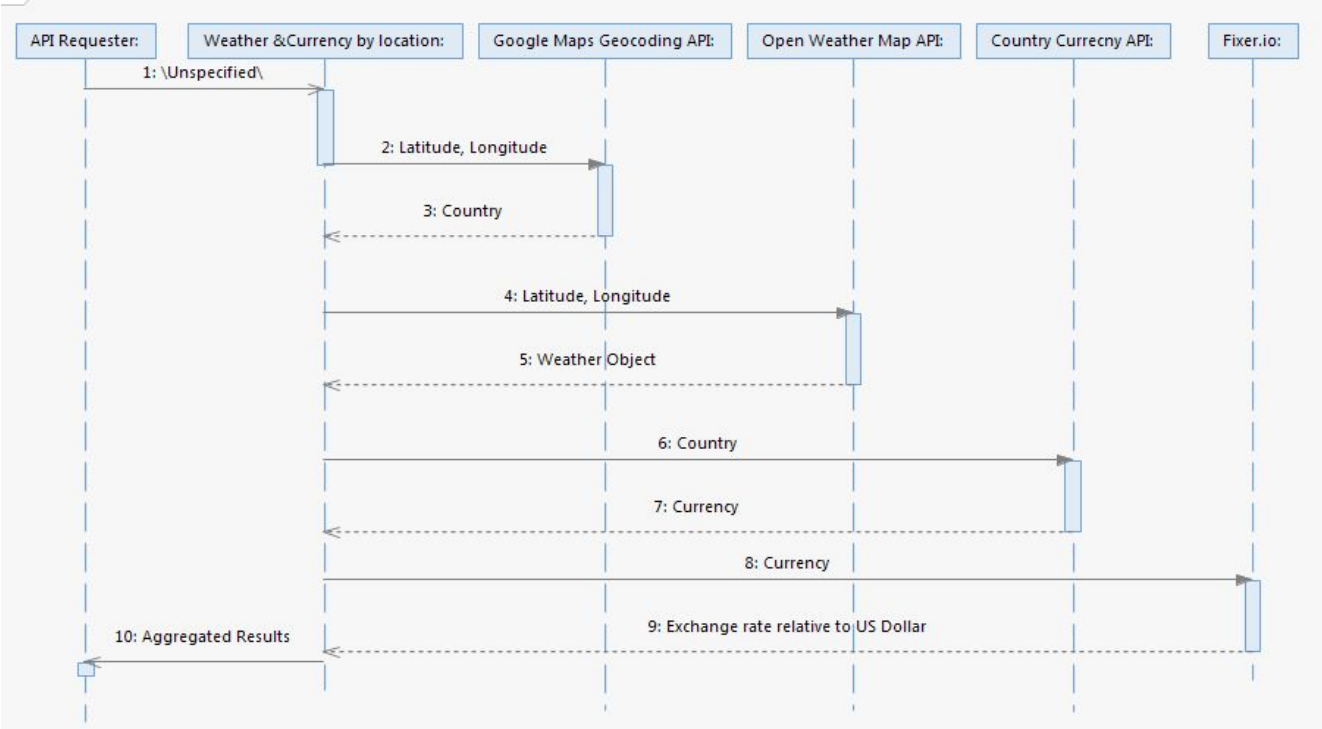

Fig. (2): Sequence diagram for the composite service.

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:

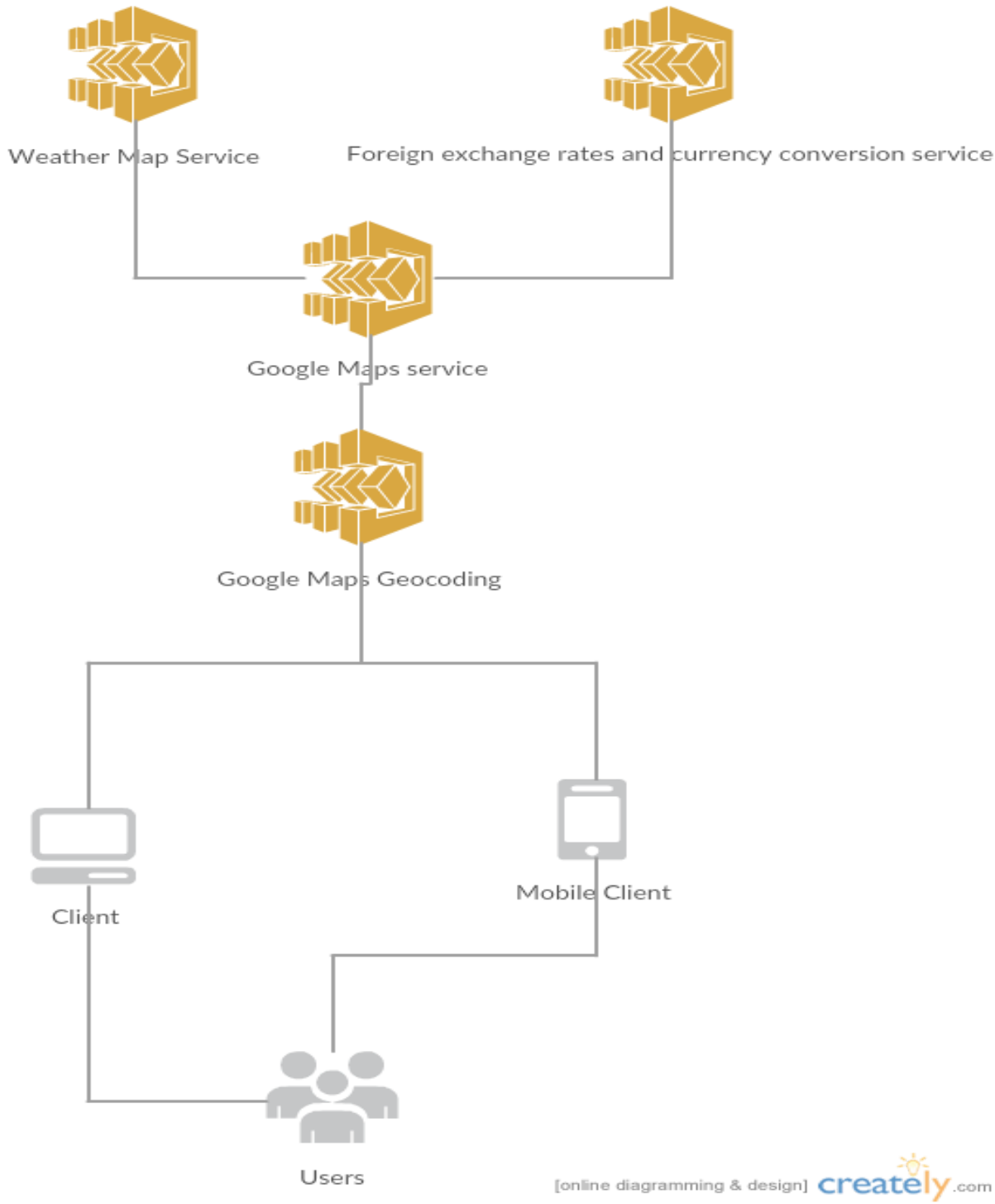

Fig. (3): The architecture of the composite service.

Technology :

We used REST for the implementation of these services.

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