**SDK Plan**

First Step: Model the data. List API URLS and return data. Using this return data create a model.

API Doc for current weather: <http://openweathermap.org/current>

USE CITY ID WHEN POSSIBLE

**Call current weather data for one location by geographic coordinates:**

<http://api.openweathermap.org/data/2.5/weather?lat=USER_DEFINED_LATITUDE&lon=USER_DEFINED_LONGITUDE&appid=USER_DEFINED_APP_ID>

http://api.openweathermap.org/data/2.5/weather?id= CITY\_ID &appid= USER\_DEFINED\_APP\_ID

**Call current weather data for several cities:**

[http://api.openweathermap.org/data/2.5/box/city?bbox=12,32,15,37,10&cluster=yes](http://api.openweathermap.org/data/2.5/box/city?bbox=12,32,15,37,10&cluster=yes&appid=b1b15e88fa797225412429c1c50c122a)

[http://api.openweathermap.org/data/2.5/group?id=CITY\_ID, CITY\_ID, CITY\_ID &units=metric](http://api.openweathermap.org/data/2.5/group?id=CITY_ID,%20CITY_ID,%20CITY_ID%20&units=metric)

**bbox** bounding box [lon-left,lat-bottom,lon-right,lat-top]

**cluster** use server clustering of points. Possible values ​​are [yes, no]

Response Format:

Parameters:

* coord
  + coord.lon City geo location, longitude
  + coord.lat City geo location, latitude
* weather (more info Weather condition codes)
  + weather.id Weather condition id
  + weather.main Group of weather parameters (Rain, Snow, Extreme etc.)
  + weather.description Weather condition within the group
  + weather.icon Weather icon id
* base Internal parameter
* main
  + main.temp Temperature. Unit Default: Kelvin, Metric: Celsius, Imperial: Fahrenheit.
  + main.pressure Atmospheric pressure (on the sea level, if there is no sea\_level or grnd\_level data), hPa
  + main.humidity Humidity, %
  + main.temp\_min Minimum temperature at the moment. This is deviation from current temp that is possible for large cities and megalopolises geographically expanded (use these parameter optionally). Unit Default: Kelvin, Metric: Celsius, Imperial: Fahrenheit.
  + main.temp\_max Maximum temperature at the moment. This is deviation from current temp that is possible for large cities and megalopolises geographically expanded (use these parameter optionally). Unit Default: Kelvin, Metric: Celsius, Imperial: Fahrenheit.
  + main.sea\_level Atmospheric pressure on the sea level, hPa
  + main.grnd\_level Atmospheric pressure on the ground level, hPa
* wind
  + wind.speed Wind speed. Unit Default: meter/sec, Metric: meter/sec, Imperial: miles/hour.
  + wind.deg Wind direction, degrees (meteorological)
* clouds
  + clouds.all Cloudiness, %
* rain
  + rain.3h Rain volume for the last 3 hours
* snow
  + snow.3h Snow volume for the last 3 hours
* dt Time of data calculation, unix, UTC
* sys
  + sys.type Internal parameter
  + sys.id Internal parameter
  + sys.message Internal parameter
  + sys.country Country code (GB, JP etc.)
  + sys.sunrise Sunrise time, unix, UTC
  + sys.sunset Sunset time, unix, UTC
* id City ID
* name City name
* cod Internal parameter

API Doc for 5 day/ 3 hour forecast data: <http://openweathermap.org/forecast5>

Accepts city id and coordinates:

http://api.openweathermap.org/data/2.5/forecast?id=CITY\_ID&appid=APP\_ID

<http://api.openweathermap.org/data/2.5/forecast?lat=LATITUDE&lon=LONGITUDE>

##### Parameters:

* code Internal parameter
* message Internal parameter
* city
  + city.id City ID
  + city.name City name
  + city.coord
    - city.coord.lat City geo location, latitude
    - city.coord.lon City geo location, longitude
  + city.country Country code (GB, JP etc.)
* cnt Number of lines returned by this API call
* list
  + list.dt Time of data forecasted, unix, UTC
  + list.main
    - list.main.temp Temperature. Unit Default: Kelvin, Metric: Celsius, Imperial: Fahrenheit.
    - list.main.temp\_min Minimum temperature at the moment of calculation. This is deviation from 'temp' that is possible for large cities and megalopolises geographically expanded (use these parameter optionally). Unit Default: Kelvin, Metric: Celsius, Imperial: Fahrenheit.
    - list.main.temp\_max Maximum temperature at the moment of calculation. This is deviation from 'temp' that is possible for large cities and megalopolises geographically expanded (use these parameter optionally). Unit Default: Kelvin, Metric: Celsius, Imperial: Fahrenheit.
    - list.main.pressure Atmospheric pressure on the sea level by default, hPa
    - list.main.sea\_level Atmospheric pressure on the sea level, hPa
    - list.main.grnd\_level Atmospheric pressure on the ground level, hPa
    - list.main.humidity Humidity, %
    - list.main.temp\_kf Internal parameter
  + list.weather (more info Weather condition codes)
    - list.weather.id Weather condition id
    - list.weather.main Group of weather parameters (Rain, Snow, Extreme etc.)
    - list.weather.description Weather condition within the group
    - list.weather.icon Weather icon id
  + list.clouds
    - list.clouds.all Cloudiness, %
  + list.wind
    - list.wind.speed Wind speed. Unit Default: meter/sec, Metric: meter/sec, Imperial: miles/hour.
    - list.wind.deg Wind direction, degrees (meteorological)
  + list.rain
    - list.rain.3h Rain volume for last 3 hours, mm
  + list.snow
    - list.snow.3h Snow volume for last 3 hours
  + list.dt\_txt Data/time of caluclation, UTC

### **Units format**

##### Description:

Standard, metric, and imperial units are available.

##### Parameters:

**units** metric, imperial. When you do not use units parameter, format is Standard by default.

Temperature is available in Fahrenheit, Celsius and Kelvin units.

* For temperature in Fahrenheit use units=imperial
* For temperature in Celsius use units=metric
* Temperature in Kelvin is used by default, no need to use units parameter in API call

List of all API parameters with units [openweathermap.org/weather-data](http://openweathermap.org/weather-data)

##### Examples of API calls:

standard [api.openweathermap.org/data/2.5/find?q=London](http://api.openweathermap.org/data/2.5/find?q=London&appid=b1b15e88fa797225412429c1c50c122a)

metric [api.openweathermap.org/data/2.5/find?q=London&units=metric](http://api.openweathermap.org/data/2.5/find?q=London&units=metric&appid=b1b15e88fa797225412429c1c50c122a)

imperial [api.openweathermap.org/data/2.5/find?q=London&units=imperial](http://api.openweathermap.org/data/2.5/find?q=London&units=imperial&appid=b1b15e88fa797225412429c1c50c122a)

### **Multilingual support**

##### Description:

You can use lang parameter to get the output in your language. We support the following languages that you can use with the corresponded lang values: English - en, Russian - ru, Italian - it, Spanish - es (or sp), Ukrainian - uk (or ua), German - de, Portuguese - pt, Romanian - ro, Polish - pl, Finnish - fi, Dutch - nl, French - fr, Bulgarian - bg, Swedish - sv (or se), Chinese Traditional - zh\_tw, Chinese Simplified - zh (or zh\_cn), Turkish - tr, Croatian - hr, Catalan - ca

##### API call:

http://api.openweathermap.org/data/2.5/forecast/daily?id=524901&lang={lang}

##### Parameters:

**lang** language code

##### Examples of API calls:

[http://api.openweathermap.org/data/2.5/forecast/daily?id=524901&lang=zh\_cn](http://api.openweathermap.org/data/2.5/forecast/daily?id=524901&lang=zh_cn&appid=b1b15e88fa797225412429c1c50c122a)

Map: The OpenWeatherMap solution has a map overlay option but it doesn’t seem to be worth the effort to incorporate given the time limit. As such, I will use a simple annotated map for the time being.

Creating a model.

City Table:

* longitude – integer
* latitude – integer
* cityId – long
* cityName – string
* cityOrder – integer (user definable)

Weather Table

* title – string
* description – string
* temperature - float
* pressure - integer
* humidity - integer
* minimum\_temperature - float
* maximum\_temperature - float
* sea\_level - float
* ground\_level - float
* wind\_speed - float
* wind\_direction - integer
* cloudiness - integerlopu
* rain\_volume - float
* snow\_volume - float
* weather\_time - long
* city\_id – foreign key

**Second step**: Create this model in code. WeatherDB

**Third step**: Create classes for location and weather. Location, Weather

**Fourth Step**: Create class for making api calls and storing results into database. OpenWeatherMap

Basics of the OpenWeatherMap API Class:

Should make calls for:

**current weather**

**Single City Current Weather**

<http://api.openweathermap.org/data/2.5/weather?lat=USER_DEFINED_LATITUDE&lon=USER_DEFINED_LONGITUDE&appid=USER_DEFINED_APP_ID>

http://api.openweathermap.org/data/2.5/weather?id= CITY\_ID &appid= USER\_DEFINED\_APP\_ID

**Multiple City Current Weather**

[http://api.openweathermap.org/data/2.5/box/city?bbox=12,32,15,37,10&cluster=yes](http://api.openweathermap.org/data/2.5/box/city?bbox=12,32,15,37,10&cluster=yes&appid=b1b15e88fa797225412429c1c50c122a)

[http://api.openweathermap.org/data/2.5/group?id=CITY\_ID, CITY\_ID, CITY\_ID &units=metric](http://api.openweathermap.org/data/2.5/group?id=CITY_ID,%20CITY_ID,%20CITY_ID%20&units=metric)

**Single City Forecasts**

http://api.openweathermap.org/data/2.5/forecast?id=CITY\_ID&appid=APP\_ID

<http://api.openweathermap.org/data/2.5/forecast?lat=LATITUDE&lon=LONGITUDE>

Inputs for this class are APP\_ID and LONGITUDE, LATITUDE. The first call to a new city will be a lat, long call. Subsequent calls should use locationID if that is known.

**Fifth step**: Create the class responsible for scheduling API calls, interfacing with database class, and is the main class to be used by other applications. JDWeatherManager

Inputs:

* latitude and longitude for places
* app id
* api call interval time

Outputs:

* methods for retrieving locations
* methods for getting weather data for locations