# Overview

The WeatherApp web site will allow users who register to log in and retrieve weather information for selected locations. By default, each user will be limited to 5 locations although users with admin privileges can increase this limit. The locations can be specified by either zip code or name/city (also possibly country as a stretch goal, but this will default to the U.S. if there isn’t time to implement the global feature).

Additional stretch goals include:

* Allow users to subscribe so that they can received daily email notifications.
* Admin statistics (for example, most popular locations, most saved, etc.)
* Highest/lowest/average historical temperatures

WeatherApp will be using the OpenWeatherMap API to retrieve weather information.

Based on the API at <https://openweathermap.org/current>, the current weather data can be accessed by city name, by city ID, by geographic coordinates, and by zip code. For WeatherApp, we will not ask the user to enter a city ID or geographic coordinates. However, it should be noted that geographic coordinates seem to be returned in the API response if needed for visualization:

##### **API response:**

{

"coord": {"lon": -122.08,"lat": 37.39},

"weather": [

{

"id": 800,

"main": "Clear",

"description": "clear sky",

"icon": "01d"

}

],

"base": "stations",

"main": {

"temp": 282.55,

"feels\_like": 281.86,

"temp\_min": 280.37,

"temp\_max": 284.26,

"pressure": 1023,

"humidity": 100

},

"visibility": 16093,

"wind": {

"speed": 1.5,

"deg": 350

},

"clouds": {

"all": 1

},

"dt": 1560350645,

"sys": {

"type": 1,

"id": 5122,

"message": 0.0139,

"country": "US",

"sunrise": 1560343627,

"sunset": 1560396563

},

"timezone": -25200,

"id": 420006353,

"name": "Mountain View",

"cod": 200

}

The primary means of accessing the OpenWeatherApi information will be by city name and zip code:

### **By city name**

##### **Description:**

You can call by city name or city name, state code and country code. API responds with a list of weather parameters that match a search request.

##### **API call:**

**api.openweathermap.org/data/2.5/weather?q={city name}&appid={your api key}**

**api.openweathermap.org/data/2.5/weather?q={city name},{state code}&appid={your api key}**

**api.openweathermap.org/data/2.5/weather?q={city name},{state code},{country code}&appid={your api key}**

##### **Parameters:**

**q** city name, state code and country code divided by comma, use ISO 3166 country codes. You can specify the parameter not only in English. In this case, the API response should be returned in the same language as the language of requested location name if the location is in our predefined list of more than 200,000 locations.

### **By ZIP code**

##### **Description:**

Please note if country is not specified then the search works for USA as a default.

##### **API call:**

**api.openweathermap.org/data/2.5/weather?zip={zip code},{country code}&appid={your api key}**

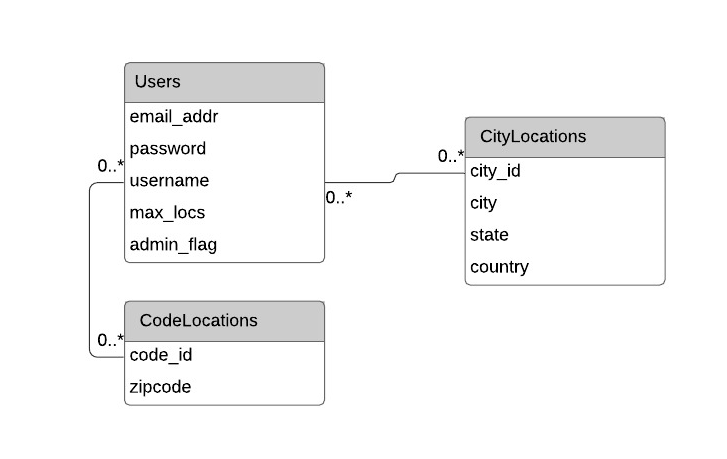
##### **Parameters:**

**zip** zip code

# Database Outline

* User: records the details of each registered user:
  + username: varchar, not NULL, unique, PK
  + password: varchar, not NULL
  + email\_addr: varchar (if not NULL, user is assumed to be subscribed for daily emails)
  + max\_locs: int, not NULL, default=5 (# of locations that can be saved)
  + admin\_flag: bool, not NULL (users with admin privileges can change the max\_locs for other users on the number of locations that can be saved)
  + Relationship: a M:M relationship between Users and CodeLocations is implemented with username as a FK inside UserCodeLocations
  + Relationship: a M:M relationship between Users and CityLocations is implemented with username as a FK inside UserCityLocations
* CodeLocation: records postcode locations saved by user
  + code\_id: int, auto\_increment, unique, not NULL, PK
  + zipcode: varchar
  + Relationship: a M:M relationship between Users and CodeLocations is implemented with code\_id as a FK inside UserCodeLocations
* CityLocation: records city locations saved by user
  + city\_id: int, auto\_increment, unique, not NULL, PK
  + city: varchar, not NULL
  + state: varchar
  + country: varchar
  + Relationship: a M;M relationship between Users and CityLocations is implemented with city\_id as a FK inside UserCityLocations
* UserCodeLocations:
  + username: varchar, not NULL, FK (from Users)
  + code\_id: int, not NULL, FK (from CodeLocations)
* UserCityLocations:
  + username: varchar, not NULL, FK (from Users)
  + city\_id: int, not NULL, FK (from CityLocations)

# Entity-Relationship Diagram



# Schema

Users(

username,

password,

email\_addr,

max\_locs,

admin\_flag)

CodeLocations(

code\_id,

zipcode)

CityLocations(

city\_id,

city,

state,

country)

UserCodeLocations{

username,

code\_id)

UseCityLocations(

username,

city\_id)