International Journal of Computer Applications (0975 - 8887)
Volume 181 - No.17, September 2018

Air2Day: An Air Quality Monitoring Adviser in Morocco

Mohamed Akram Zaytar
PhD Research Student
Department of Computer Engineering
Faculty of Science and Technology
Route Ziaten, Tangier, Morocco

Mohamed Amrani
Research Student
Department of Computer Engineering
Faculty of Science and Technology
Route Ziaten, Tangier, Morocco

Chaker El Amrani
Associate Professor
Department of Computer Engineering
Faculty of Science and Technology
Route Ziaten, PO. Box 416, 90000
Tangier, Morocco

ABSTRACT

This article aims to present an end-to-end software solution capable of providing up to date weather and pollution values and health recommendations based on User profiles and personal health data, while making use of environmental satellite data processed in the back-end. this system demonstrates the possible range of applications of satellite-backed environmental systems that can assist and potentially replace the current expensive sensor-based systems, especially in developing countries in Africa.

General Terms

Android development, Remote Sensing, Software Engineering

Keywords

Android, Air Quality, Pollution, Weather, RESTful API

1. INTRODUCTION

In the current day and age, air pollution represents a major challenge in Africa, It's causing more premature deaths than unsafe water or childhood malnutrition causes [1], with the increasing exploitation of Africa's rich natural and mineral resources, Population growth and urbanization, traffic emissions, transported dust, power plants and open burning could cause a health and climate crisis reminiscent of those seen in China and India [2], the increasingly alarming factors surrounding outdoor air pollution and climate change was the primary reason for us to build this system.

To Cover a wide geographic area, A sensor-based network is the general solution, as seen in other parts of the world (China, Europe, the USA [3]). usually, the network is comprised of multiple IoT devices and sensors connected to the internet to deliver rich local data to central servers to be cleaned, preprocessed, and visualized on web or mobile interfaces as in the form of numerical inferred measurements that describe the weather and pollution based on the users' GPS location, this IoT approach is expensive and very hard to establish given the limited resources and difficult procedures one needs to go through to install the sensors to cover Morocco, because of that, a relatively novel concept is proposed that's based on the idea of using satellite measurements and imagery to tackle the problem of pollution monitoring in Africa and especially Morocco. Satellite-based data is of high quality [4] and offer a cheap alternative to the traditional sensor-based environmental data that can be of high use not just in air quality monitoring but to in numerous other environmental problems like agriculture, weather, industrialization and urbanization monitoring.

The proposed system architecture is comprised of multiple connected sub-systems, starting out from the original data source and ending with the mobile client interface. firstly, the data is receieved from near-real time broadcasting meteorological satellites such as the geostationary satellite (MSG) and the polar Metop satellites (Metop-A/B) [5], into a local station named MDEO (The Mediterranean Dialogue Earth Observatory) in Tangier, Morocco, after getting the native data, snapshots of backups were stored in a FS server, the next step is to clean, preprocess and store structured data into a relational GeoDB server, connected directly to A RESTful API which is considered the middleman between the Database and the Clients or user facing interfaces.

Finally, a native android application was built, and it's responsible for getting the structured data using the API to visualizing it, and provide recommendations based on the provided personal user information. the back-end systems when combined with the user facing mobile interface, result in rich weather and pollution map visualizations and useful personalized health recommendations for online users.

2. MDEO : BIG ENVIRONMENTAL DATA

MDEO, or the Mediterranean Dialogue Earth observatory, is a NATO-sponsored project that aims to tackle serious environmental problems ranging from disaster warning, pollution monitoring, numerical short and long term forecasting of weather and pollution variables, with a wide spectrum of potential applications such as Health, Energy, Water, Weather, Agriculture, Industrial/Urbanization monitoring, Early warning of natural disasters like Floods, Earthquakes, and so on. To achieve this, MDEO's based hardware and software stack is capable of Acquiring, processing, storing, and archiving the near-real time data it receives in fixed intervals of time. MDEO is composed of a ground station, a DVB-S2 receiver to get and decipher the raw data into bulks of archived files, it also contains a cluster of servers responsible for the acquisition and processing of the files, a GPU server connected to the cluster and responsible for GPU backed HPC/AI/ML related processes [6]. In the software department, the

main software piece is called TeraCast, it is responsible for different tasks such as receiving the broadcasted data, preprocess it using different internal modules, and make it available for download in various formats such as HDF5 and BUFR, using the FTP protocol.

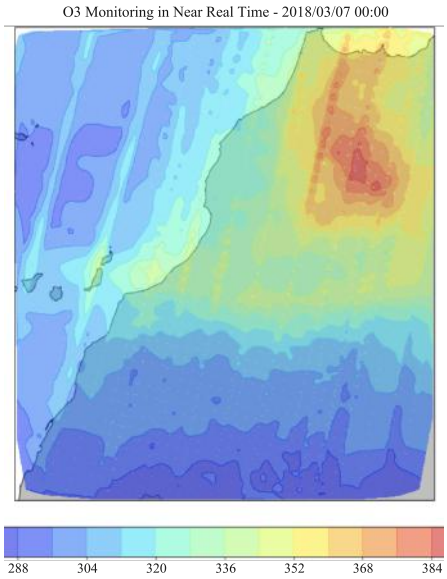

Fig. 1: MSG : Interpolation of MDEO's O3 measurements - Morocco

Because MDEO receives the land surface temperature data from a geostationary satellite, it scans Morocco every 15 minutes [7], as for pollutants values, MDEO gets the data from two polar satellites named Metop-A and Metop-B, MDEO receives scans of the whole earth surface in intervals of 24 hours, as a result, MDEO processes scattered measurements of air pollution of Morocco each 12 hours.

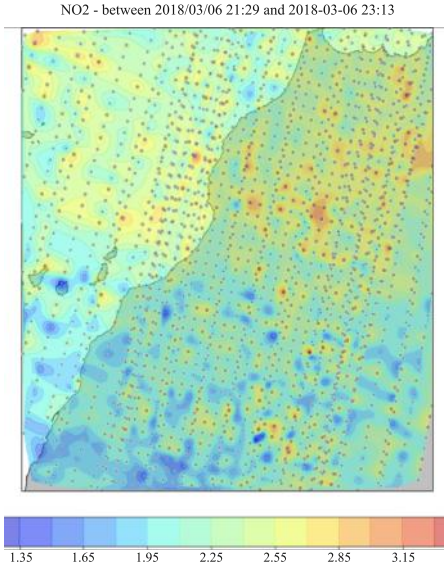

Fig. 2: Metop: Interpolation of MDEO's NO2 measurements - Morocco

3. API ARCHITECTURE

The data stored by MDEO's systems reside in an FS server in multiple formats, some of them are native/binary (BUFR, GRIB ..) [8] and some are not (HDF5, NetCDF) but not quite structured or ready for fast access by an API, Therefore, before building the API, an extra layer was built to check the FTP server periodically for recent archived files and retrieve any new files to another server responsible for decoding the data measurements, filtering the values and turning them into tabular-like lists, and finally inserting the resulting bulks of rows in the GeoDB. Both data products for the temperature and pollution were provided by MDEO using the HDF5 format. Only recent data from Morocco was filtered to not overload the DB with millions of rows per hour.

Before modelling and building the RESTful API, A relational DB was chosen to extend regular SQL functionalities with Geographical utilities that are essential for the clients, to assure this, the resulting DB is an instance of the PostGreSQL spacial database extender named PostGIS [9], its sole purpose is to add support for geographic objects allowing location queries to be run in raw SQL. For the first version of the Instance, two tables were created for the temperature and the pollutants values in Morocco, and by adding indexes on the location and Date related columns, performance and speed were assured across the millions of rows sent regularly.

The geoAPI was built and deployed on an external server directly connected to MDEO's GeoDB, it was built on top of Django's REST Framework in combination of GeoDjango, the resulting two tables represent the two essential sets of variables, temperature variables, and pollutants variables.

Every table row represent a single measurement of a specific geographical point at an exact date and time.

The Temperature End-Point Variables :

```
GET /api/temperature/
```

Table 1. : Temperature Table Variables

Variable	Unit of Measurement	Description
value	Celsius	The Temperature measurement value
value_error	Percentage (%)	the maximum value error percentage
quality_flag	$\mathbb{N}$	Represents the condition at which the temperature measurement was taken, it has a well defined range of distinct possible values [10]
Point	$\mathbb{R}^2$	The exact geographical point of measurement, it is represented by a tuple of the 2-D coordinates (latitude,longitude)
Date	Y-m-d H:M	The full timestamp of the measurement, down to minutes in precision

For Pollution, MDEO receives measurements on multiple pollutants from the same satellite sensor, resulting in measurements for the same geographic point on the same datetime stamp for each pollutant of interest.

The Pollutants End-Point Variables :

```
GET /api/pollutants/
```

Table 2. : Pollutants Table Variables

Variable	Unit of Measurement	Description
Point	$\mathbb{R}^2$	The exact geographic point of measurement, represented by a tuple of the 2-D coordinates (latitude,longitude)
HCHO	$molecules/cm^2$	Formaldehyde's vertical density measurement of the specified geographical point
NO_2	$molecules/cm^2$	Nitrogen dioxide's vertical density measurement of the specified geographical point
NO_2Tropo	$molecules/cm^2$	Nitrogen dioxide's vertical density on the tropospheric level for the specified geographical point
O_3	Dobson Units	Ozone vertical density measurement of the geographical point
SO_2	Dobson Units	Sulfur dioxide's vertical density of the geographical point at the time of the measurement
Date	Y-m-d H:M	The full timestamp of the measurement, down to minutes in precision

To make use of the following End-points, several necessary filters were built to ease the task of data retrieving and processing, the following filters can be used in combination (using the & operator) to retrieve the most useful results needed by the client, the most essential filters (used for both temperature and pollutants end-points) are :

—**Rectangle Area** : Retrieving data inside a predefined rectangle of interest, example :

```
GET /api/temperature/?
 lat1=[LAT_1]&lat2=[LAT_2]
 &
 lon1=[LON_1]&lon2=[LON_2]
```

—**Date Range** : Retrieve data in a predefined data range of interest, example :

```
GET /api/temperature/?
 date_start=[%Y-%m-%d %H:%M]
 &
 date_end=[%Y-%m-%d %H:%M]
```

—**Circle** : Return all points within a certain radius from a central point, example :

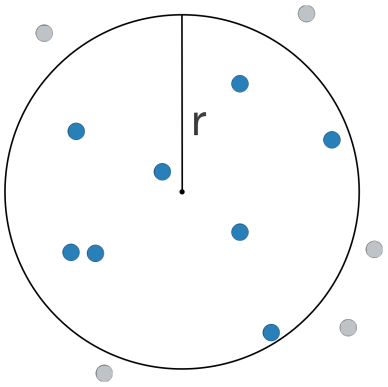

Fig. 3: Return Points surrounding a center point

```
GET /api/temperature/?
 center_point=[LONGITUDE,LATITUDE]
 &
 radius=[RADIUS](in Miles)
```

—**Closest** : Return the closest point to a predefined center point with a predefined date range, setting the closest parameter to 1, example :

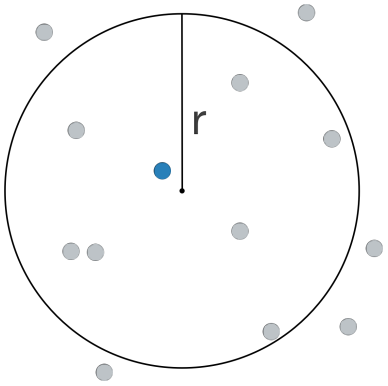

Fig. 4: Return the closest point to the center

```
GET /api/temperature/?
 date_start=[%Y-%m-%d %H:%M]
 &
 date_end=[%Y-%m-%d %H:%M]
 &
 center_point=[LONGITUDE,LATITUDE]
 &
 closest=1
```

In Using the following end-points and filters, web and mobile clients can get Temperature and Pollution data for the purposes of graphing visualizations and inferring general health Recommendations for end-users.

4. ANDROID CLIENT APPLICATION

A mobile application was built to provide the end-user with the following functionalities :

—App Authentication using a username/password combo to support profile data persistence.
—The Ability to edit the personal profile at any time to get new personalized recommendations.
—The ability to consult weather conditions and Current personalized recommendations by tapping on the home screen.
—The ability to visualize Temperature and Pollution maps separately on different screens.
—The ability to enable/disable certain settings or consult support, a setting screen is provided for these purposes.

The following figure showcases all of the activities that a typical user can do on the app.

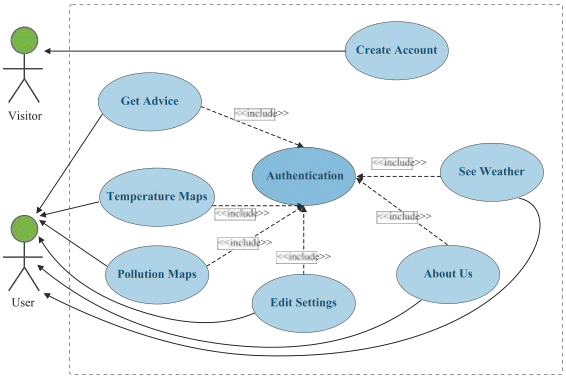

Fig. 5: Use Case Diagram

The android application was built using Java on Android Studio, supported by A local SQLite database to keep the personal and recommendation data on the phone, it was Prototyped using LucidChart, the communication between the client and the RESTful API was based on token-based HTTP authentication and responses are received in Plain JSON.

The following class diagram was exported to demonstrate the SQLite Tables used to model the user stories.

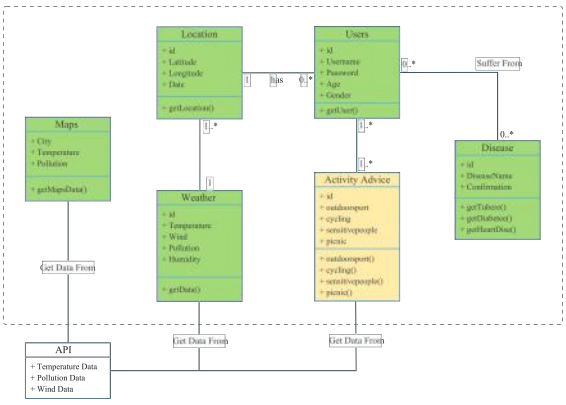

Fig. 6: Class Diagram

The client exploited another layer of data from OpenWeatherMap to further support the satellite numerical data on the occurrence of outages relating to MDEO, and to add other useful weather variables that aren't available on MDEO such as wind speed and humidity. for graphical Plotting, the application leveraged Google's geolocation APIs [11].

Recommendations were mostly inferred from the density of ozone measured in Dobson Units (in addition to the users' personal information and the general weather conditions) on a given time, and because the client receives data in near real time, this allows the user to get near real time health recommendations based on the following table :

Table 3. : Ozone Recommendations

Ozone Quantity (in DU)	Who needs to be concerned ?	What Should The Users Do ?
290-310	No One	it's a great day to be active outside
310-340	Some People who might be unusually sensitive to ozone	Consider reducing prolonged or heavy outdoor exertion. Watch for symptoms such as coughing or shortness of breath.
340-360	Sensitive Groups include people with lung disease such as asthma, older adults, children and teenagers, and people who are active outdoors	Reduce Prolonged or heavy outdoor exertion. Take more breaks, do less intensive activities, and watch for symptoms such as coughing or shortness of breath. Schedule outdoor activities in the morning
360-400	Everyone	Avoid Prolonged or heavy outdoor exertion. Schedule outdoor activities in the morning. Consider moving activities indoors.
400-500	Everyone	Avoid All outdoors Physical Activities. Do all activities Indoors or reschedule to a time when air quality is better.

Other variables that contribute to the recommender system include Temperature, Wind Speed, Humidity, the Age of the user, and the presence of some chronic diseases.

The final health recommendations give general advices on Outdoor Activity (Going out), Doing Sports, Cycling, Picnic, and are targeted for people of old age, suffering from allergies, Heart disease, Diabetes, Tuberculosis.

5. RESULTS

5.1 User experience

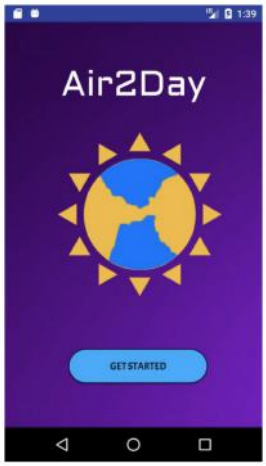

Fig. 7: Welcome Screen

Fig. 8: Personal Info Screen

After signing up and filling the personal health information, the application retrieved and updated the API data each time the user refreshed the screen, resulting in a near real time health recommendations on outdoor activities, a ”contact us” screen is also available to provide support for people who have questions about the recommendations or how the app works in general.

Fig. 9: Settings Screen

Fig. 10: Pollution Map Screen

The home Screen for online users displays the latest values of pollution and weather based on both the MDEO and the OpenWeatherMap RESTful APIs, and the same variation happens with the recommendations on outdoors activities based on the most up-to-date personal information, the current weather and air pollution conditions.

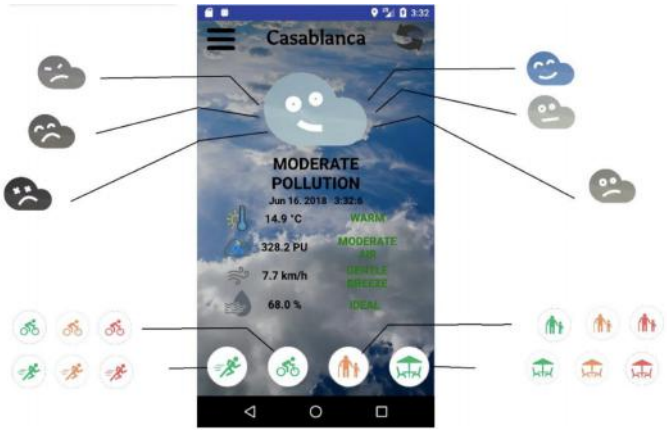

Fig. 11: Dynamic Weather, Pollution, and Recommendations

5.2 Performance

Because the API filters were optimized for fast retrieval, the application was quick enough to serve all of requests under a second for most of the time. But for complex recommendations, it was necessary to separate the multiple needed processes to retrieve and recommend into smaller ones in different parts of the UX to make the overall user experience smoother for online users. Some of these techniques are showcased next :

—When a user first opens up the app, it asks for GPS data and load weather and pollution data as a background process as the user fills in his personal information.

—When a user decides on refreshing the app to get new data, the app explicitly asks the API for a start date greater than the last time the app updated the data, this way of incremental updating provides a better user experience when interacting with the recommender system.

—The recommender system process doesn’t take into account historical/past data, it exports recommendations based on single-values (for each variable) of near real time weather and pollution data.

6. CONCLUSION

MDEOs data pipeline provides dozens of weather, climate, and pollution based data products, the wealth of environmental data offers numerous research opportunities when it comes to pollution and weather monitoring [12], numerical forecasting [13], disasters warning and so on. When coupled with fast APIs and UI clients such as web and mobile user interfaces, MDEOs data could have the potential to positively impact health and change ours lives for the better.

Remote sensing has a lot to offer to Africa and especially to Morocco. In the Future, more significant challenges should be tackled concerning Big environmental data processing, Artificial Intelligence, Systems Architectures and scalability in the Cloud, to provide more useful Applications and Research material.

Any partnership and collaboration with third-parties and end-users such as other universities/labs, the Moroccan Ministry of Health, The Ministry of Agriculture and Fisheries, The Civil Protection Agency, The Agency for coordination of Water and Forestry among

numerous other organizations and associations is welcomed, and the support for Environmental and Climate Data Science Research should be reinforced for a better environment that is safe for Everyone.

7. ACKNOWLEDGMENT

The authors are thankful to the Ministry of Higher Education and Scientific Research, and the National Centre for Scientific and Technical Research (CNRST) for funding this study, under project codename : PPR/2015/7.

8. REFERENCES

[1] Nigel Bruce, Rogelio Perez-Padilla, and Rachel Albalak. Indoor air pollution in developing countries: a major environmental and public health challenge. *Bulletin of the World Health organization*, 78:1078–1092, 2000.

[2] Chak K Chan and Xiaohong Yao. Air pollution in mega cities in china. *Atmospheric environment*, 42(1):1–42, 2008.

[3] Ranjit Kaur and Pankaj Deep Kaur. A review on various iot analytics techniques for air pollution detection in fog computing. *International Journal of Computer Applications*, 169(2):1–4, Jul 2017.

[4] Rosemary Munro, Michael Eisinger, Craig Anderson, Jörg Callies, Enrico Corpaccioli, Rüdiger Lang, Alain Lefebvre, Yakov Livschitz, and A Perez Albinana. Gome-2 on metop. 1216:48, 2006.

[5] VK Gaertner and M Koenig. Eumetcast: The meteorological data dissemination service. 2006.

[6] C. El Amrani, G. L. Rochon, T. El-Ghazawi, G. Altay, and T. Rachidi. System architecture of the mediterranean dialogue earth observatory. pages 600–603, July 2013.

[7] Johannes Schmetz, Paolo Pili, Stephen Tjemkes, Dieter Just, Jochen Kerkmann, Sergio Rota, and Alain Ratier. An introduction to meteosat second generation (msg). *Bulletin of the American Meteorological Society*, 83(7):977–992, 2002.

[8] Jean Claude Bergès. Support of wmo binary format (bufr and grib). pages 11–13, 2002.

[9] Jhummarwala Abdul, M.b. Potdar, and Prashant Chauhan. Article: Parallel and distributed gis for processing geo-data: An overview. *International Journal of Computer Applications*, 106(16):9–16, November 2014. Full text available.

[10] JA Sobrino and M Romaguera. Land surface temperature retrieval from msg1-seviri data. *Remote Sensing of Environment*, 92(2):247–254, 2004.

[11] Andrei Popescu. Geolocation api specification. *World Wide Web Consortium, Candidate Recommendation CR-geolocation-API-20100907*, 2010.

[12] Rob Kitchin. The real-time city? big data and smart urbanism. *GeoJournal*, 79(1):1–14, 2014.

[13] Madhavi Anushka Elangasinghe, Naresh Singhal, Kim N Dirks, and Jennifer A Salmond. Development of an ann–based air pollution forecasting system with explicit knowledge through sensitivity analysis. *Atmospheric pollution research*, 5(4):696–708, 2014.