Providing Open Environmental Data – The Scalable and Web-Friendly Way

Maria C. Borges, Frank Pallas, and Marco Peise

Information Systems Engineering Group, TU Berlin, Berlin, Germany
{mb,fp,mp}@ise.tu-berlin.de

Abstract. The emergence of low-cost environmental sensors has presented an opportunity for environmental data, as a substantial pool of real-time and historical data becomes openly available. This environmental open data provides potential for new opportunities to enhance environmental applications. However consuming this data as it is currently available presents many challenges, including heterogeneous platforms and data schema, archaic data formats and limited scaling potential. We address these issues in our solution `OpenSense.network`. This paper describes in detail the development of the platform, including data model, system architecture and data collection approach. The presented architecture is able to serve huge amounts of data, through the deliberate employment of a decentralized time-series database in combination with a powerful spatial and relational database. Furthermore, we pay special attention to data consumption in our approach and suggest a web-friendly JSON-based API and a discoverable graphic user interface.

Keywords: open data, environmental monitoring, web APIs, scalability

1 Introduction

Traditional environmental monitoring relies on a very low number of static stations equipped with costly yet highly precise monitors. These are complemented by sophisticated computational models that help fill in the spatial gaps where no monitoring takes place. Even though these models are effective in their main applications, especially in the domains of forecasting and scenario analysis, they depend on highly specialized knowledge and input data and can be prone to errors such as skewed samples or biased model design [2]. Furthermore, the limited number of stations also avoids highly local phenomena like hotspots of low air quality from being recognized and incorporated in respective models. Recently, efforts have thus been made to expand the collection of environmental data beyond these reference stations. With the emergence of low-cost environmental sensors, such as the weather station from Netatmo[1] and senseBox[2], or

[1] http://www.netatmo.com
[2] http://sensebox.github.io/en/

simple "do-it-yourself" solutions like the so-called "Airrohr" suggested by luftdaten.info[3], the deployment is now easier than ever, facilitating a coverage and spatial resolution that would not be possible on the basis of traditional high-cost stations.

These inexpensive devices, though inferior in accuracy due to their economical design, provide a substantial pool of real-time and historical data. At the same time, governments and organizations have also recognized the potential of making their environmental data available to the public. Together, these two developments create manifold opportunities for enhancing existing environmental applications and for establishing completely new ones.

In particular, the exploitation of the generated open data can lead interested developers to create innovative applications with added economic and civic value. Combinations of different data sources can also provide valuable insights, i.e. by revealing new correlations, if appropriate visualization and browsing tools are made available [11]. Furthermore, open data can create a well-accepted data foundation for researchers. It can serve as the basis for the evaluation of novel mechanisms, as well as provide a baseline for performance evaluations [4]. Lastly, open environmental data can help engage citizens in environment-related activities, letting them monitor their local air quality, increasing awareness and perhaps motivating further preservation efforts.

However, consuming open environmental data as it is currently available presents several challenges. In particular:

- the data is usually stored in different platforms with heterogeneous data schemas. Consuming data from different sources thus typically involves disparate code. Developers are therefore subject to high integration costs, which hinders reuse and recombination [12].
- many of the available platforms provide their data in archaic semi-structured formats, such as .csv, while modern internet applications typically rely on the more intuitive json format for communication, which could circumvent cumbersome parsing on the application side.
- data provision itself is often realized through a large number of archive files hosted on FTP- or HTTP-servers. These files must typically be downloaded entirely, even if only small portions of contained data are actually of interest. Besides substantial traffic overhead, this also conflicts with established communication schemes and implementation practices for modern internet applications, introducing significant obstacles for actual data use.
- existing pools for open environmental data often exhibit limited scalability in matters of storage and throughput, leading to compromises like limited availability of historical data or the provision of rolling snapshots instead of allowing real-time queries.

In this paper, we present `OpenSense.network`, our approach to solving the above-mentioned challenges associated with a great number of available open environmental data platforms. OpenSense.network is designed to collect open data

[3] http://www.luftdaten.info

from diverse sources, including traditional environmental sensing networks like the German national weather service *Deutscher Wetterdienst*, as well as participatory sensing platforms and communities such as *luftdaten.info*. Our solution is designed with the data consumer in mind, providing a graphic interface to explore the spatial data sensors, and an API that allows developers to directly access raw data through a powerful query interface. We specifically address scalability issues through careful data modeling and deliberate system architecture design choices.

The remainder of this paper is structured as follows: In section 2, we introduce our foundational data model and its underlying considerations. In section 3, we then present our system architecture specifically tailored to the scalable serving of huge amounts of open environmental sensor data. Section 4 reports on our importing efforts for two particularly interesting and diverse data sources – the German Weather Service (DWD) and the community-driven project luftdaten.info – and the insights we gained from these efforts. In section 5, we elaborate on the possibilities for data consumption provided by OpenSense.network (including a web API and a rudimentary map interface). Section 6, in turn, briefly discusses related work in the form of alternative data platforms while section 7 concludes and provides an outlook to future work based on our platform's current limitations.

2 Data Model

The aim of OpenSense.network is to provide a single data platform for different environmental datasets. These originate from various sources with heterogeneous data schemas. The primary focus of our data model was to provide a robust definition of the distinct components: it should be rich enough to encompass different environmental observations and phenomena, yet it should stay intuitive and navigable representing only the information necessary for discovering and analyzing sensor data.

We build on the standardization efforts of the Open Geospatial Consortium (OGC), in particular their Sensor Model Language [1] and the SensorThings API [8]. These projects were conceived with the goal of establishing standards to interconnect sensing devices and enable sensing-as-service, similar to a "sensor cloud". Their data models serve as a good foundation, however we were able to make several abstractions in order to achieve a simpler and more comprehensive schema.

We propose an open environmental data model with the following entities: *sensors* represent our central element, as the source of environmental data. They generate a stream of *values*, i.e. individual observations of environmental phenomena. Each sensor observes a specific phenomenon or property, which we have decided to designate a *measurand*. These can be measured in different *units*. *Users* own sensors, for which they can assign a *license* from a specific set. These licenses regulate permissions for data produced by the sensors. Figure 1 shows

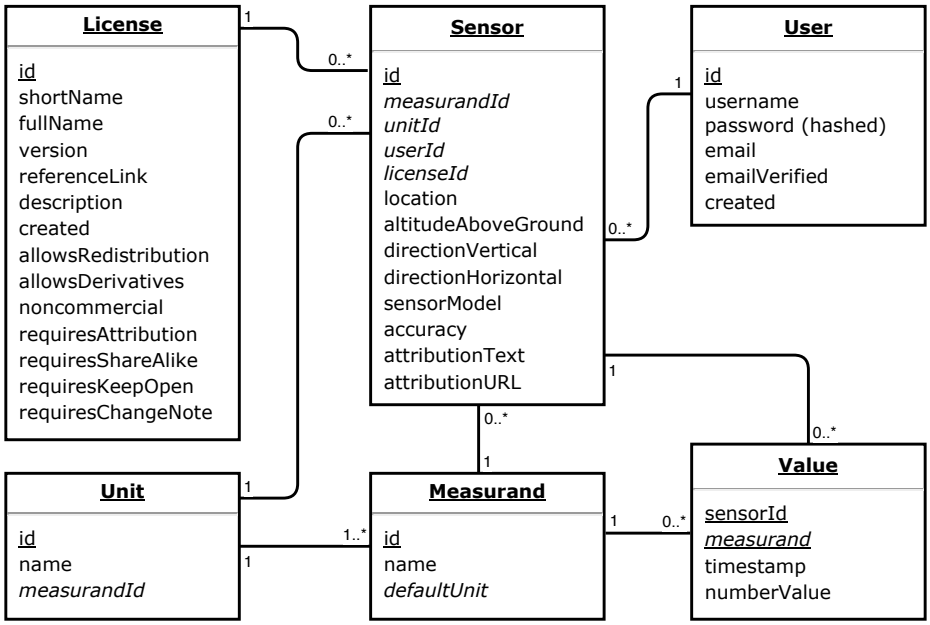

Fig. 1. Data Model of OpenSense.network

a complete overview of our data model including relationships and attributes: primary keys are underlined, while foreign keys can be identified in italics.

In order to illustrate our open environmental data model, we refer to the following example: A user named Alice deployed a small weather station by her window. This station measures temperature in celsius and humidity in percent. Alice decides to contribute to OpenSense.network and settles for an Open Data Commons Attribution License. According to our data model, user Alice owns two different sensors: "sensor1" with measurand "temperature" in unit "celsius", and "sensor2" with "humidity" in "percent". Both produce a different set of values and are assigned the open data attribution license.

We discuss key decision points of our data model below:

Sensors: We settled for a flattened data model abstracting away stations containing various individual sensors. Even though this differs from the data model of various sources, like DWD and luftdaten.info, we consciously did so for being able to also include other sources using a flattened model and to reduce the overall complexity of the data provided by our platform. Including stations in our data model would require cumbersome geographical matching through location joins, which we deemed not necessary for the platform (such aggregations could eventually be implemented on the frontend side).

Furthermore, we have limited the first prototype of the platform to time-series datasets from fixed static sensors. The location of the sensor is therefore

summarized in a single attribute field. Reasons for this decision will be further explored in section 3.2. Additionally, we have included a third location dimension, namely *altitudeAboveGround*. We chose this attribute instead of the more commonly used altitude, because we want to facilitate the introduction of sensors in the network on the end-user side. Users are much more likely to accurately estimate the altitude of a sensor above a ground that they can see, than the altitude above sea level. Computation of the real altitude could then be delegated to the backend. We have also included the angular relationship of a sensor in the attributes *directionVertical* and *directionHorizontal*, as suggested by the OGC [1].

We also incorporated an *accuracy* attribute, motivated by the research in [2], who identify severe limitations in low-cost air quality sensors. An accuracy score would thus enable the filtering of the data for use-cases that require a certain level of data-quality such as research.

Measurands and Units: We have opted to only support a curated list of measurands and units in our platform. This helps avoid redundancies in the database, stops the misuse of the platform for purposes other than environmental monitoring, and prevents input errors. Besides that, a curated list of units also allows for unit conversion on the backend side (when the units allow it), thus enabling the combination of same-measurand datasets into a single time-series table.

License: Open datasets are shared under a specific group of policies that regulate the access, re-use, and redistribution of the data. These policies are typically summarized under a license, with each license providing a particular set of permissions. In our platform, we specifically model these licenses as entities, so that their relationship to individual sensors can later be utilized to filter datasets according to specific permissions, similar to a Google Image search by usage rights.

3 System Architecture

To store and retrieve data produced by a high number of environmental sensors around the world, a highly scalable solution is imperative. In order to avoid the pitfalls of other large-scale open data platforms such as [3] and [10], which use a single relational database to store data, we rely on the principles of polyglot persistence [9] to design our solution. Polyglot persistence describes the concept of using different data storage solutions to handle different data storage needs.

In the use-case of environmental monitoring, large sensor networks continuously measure meteorological and geological phenomena, generating time-series datasets that can span over long periods of time. These sensor observations require metadata in order to provide context to the data. Consequently, we identify two different storage needs:

1. The capacity to meet the demands of modern environmental sensor networks, which produce relentless streams of time series data. A solution must be able to accommodate a growing dataset and workloads with high read and insert rates.
2. The ability to query and group sensors based on their contextual information. This includes the measured phenomena (measurand) and the measured unit, the geographical location of the sensor, the license agreement under which users can reuse the data, and other secondary attributes such as accuracy or altitude.

In order to serve these incompatible demands, we propose a hybrid approach similar to the one taken in [7], resulting in an efficient delivery of large time series datasets with expressive metadata. We use three components within our solution: (1) a write-efficient time series database (2) a relational database enhanced with geospatial query capabilities and (3) a horizontally scalable API that processes requests and issues them to the databases. Thus, our OpenSense.network backend was conceived to balance the trade-offs between traditional relational databases, which enable powerful query functionality but offer limited scalability, and modern NoSQL databases, which can scale tremendously but sacrifice query complexity by only supporting simple data schemas.

Furthermore, we deliberately decided against a "data-lake" and analytics approach, typically employed by some organizations to deposit structured and unstructured data originating from different sources. Our platform aims at collecting environmental data for a specific curated set of phenomena. In line with the nature of the data and its anticipated use, we instead opted for conscious schema design. A powerful API can then utilize the schema to best advantage and provide efficient access to raw data, which data lakes struggle to do.

Our solution, illustrated in figure 2, is implemented using open-source software and without resorting to proprietary computing or storage services in the cloud. This ensures long-term independence and facilitates replication of our architecture in similar projects. The project is currently hosted on a small cluster, relying on the university's infrastructure. In the following, we reason our choice of databases, taking into account the requirements of our use-case. We pursue scalability by design, in order to provide a platform that is able to handle and serve growing datasets without significant performance impacts.

3.1 Time Series Database

Relational databases offer poor support for time-series data, which can only be stored by inserting one row per measurement. At the rate modern time-series data is captured, this can quickly add up to massive, inefficient tables. NoSQL approaches make use of their non-relational data model to provide considerable advantages in this use-case. Numerous general-purpose and specialized databases have emerged as well-suited for this kind of data. Instead of storing one measurement per row, these databases are able to store many values in each row. *InfluxDB* and *OpenTSDB* are the most prominent examples of specialized

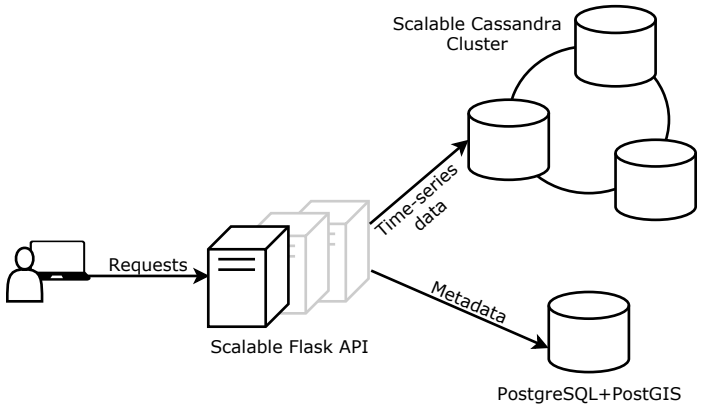

Fig. 2. Overview of OpenSense.network's system architecture

time-series databases, while more general purpose databases such as *HBase* and *Cassandra* can be readily adapted to be used for such use-cases. These databases share many similarities, mainly in regards to the way they structure data. Still, they differ in their offered functionality and distribution architecture, which directly impacts workloads (write versus read operations).

InfluxDB, the time-series solution used in [7], offers an open-source core version. This version, however, does not come equipped with a clustered deployment of the database, as clustering services are not open-source. The deployment of a single-node solution was thus deemed unsuitable, as it would quickly develop into a performance bottleneck.

OpenTSDB is a database management system that extends *HBase* with time-series capabilities. It offers a number of tools to work with such datasets, such as aggregation, downsampling and rate conversion. However, *OpenTSDB* aggregates data by default and requires an aggregation operator for every query. For a database that aims at providing raw open-data, this feature was considered non-desirable.

Cassandra and *HBase* are both general-purpose wide-column stores, offering roughly the same functionality. They differ in their clustering design. HBase uses a master-slave distribution, while Cassandra offers a fully decentralized ring architecture. This scales very well, as nodes can be easily added to increase write and read throughput. Due to our requirement to accommodate a growing dataset and workloads with high read and insert rates, we believe Cassandra to be the most appropriate solution for our use-case.

Key Design: In order to ensure best performance in Cassandra, the key schema has to be carefully selected: given a good distribution of keys between the nodes, Cassandra is essentially linearly scalable [6]. Cassandra developers have identified three different key patterns for storing time-series data [5]. The first creates

a wide row of data for each sensor, with the measured values as the column values. The second pattern, for use-cases where the data produced by one sensor quickly grows impractical (e.g. storing data every millisecond) involves tagging the row key with additional information, for example the date or month. With this approach, one is able to limit the size of individual rows, having just one row per day or month. This, however, limits the query ability of the table, as range queries spanning beyond the time interval of each row would have to be queried against multiple rows on possibly different nodes. The last common key design pattern tags each data point with a time-to-live. This is especially useful for use-cases where older data expires or becomes purposeless.

Only the first two key-patterns were deemed suitable for OpenSense.network, as the database aims at fully supporting historical datasets. Between the two other options, we opted for the first key design pattern, as we don't anticipate any sensor rows reaching beyond impracticality.

3.2 Spatial and Relational Database

In order for time-series data to be found, understood, accessed and fully exploited, the platform should support queries based on the contextual information of sensors. Interactive discovery and combination of environmental datasets is only possible if users can effectively sort through the huge dataset to find what they need. Relationships between sensors, measurands, units and licenses play a significant role in our data model, reviewed in the previous section, and should therefore be supported. Moreover, location is of particular importance in the exportability of the sensors, as it is the foremost filtering criteria. Thus, we focused on a solution that could provide us with sophisticated query functionality, in combination with a powerful spatial engine.

Relational databases structure data to recognize relations between stored items and seemed therefore tailor-made for the particular use-case of sensor metadata. Furthermore, they can be extended with spatial capabilities that store spatial data alongside non-spatial data, so that the two can be accessed together. Entries are stored in a table and the spatial index is built separately. The database engine is then able to make use of multiple indices and exploit them to their maximum advantage in order to quickly retrieve records that fulfill all criteria.

PostgreSQL and PostGIS' dominance in the spatial database domain is undeniable. They offer over 50 spatial relationship functions that can be combined with other conditions in SQL queries. We believe it to be the database that could offer us the most sophisticated query functionality. It is also the database adopted by numerous other open data systems, including [7] and [3].

Our only reluctance regarding this decision is the well documented limited scalability of relational databases. These scalability limitations are only aggravated in the context of spatial data, as spatial indexes are computationally expensive to maintain. In order to manage this limitation, we have limited the platform to fixed location sensors, and leave open the possibility to horizontally

partition the sensor table along the spatial index, dividing sensors into a coarse geographical grid.

4 Importing

Importing data from various existing sources and providing it in a scalable and web-friendly way to data consumers is the core goal of OpenSense.network. For this aim, we developed an easily expandable importer framework. So far, importers have been implemented for two particularly valuable and comparably diverse sets of open environmental data:

German Weather Service (DWD): In its "Climate Data Center"[4], DWD provides historical meteorological data for various measurands (temperature, air pressure, wind, etc.) ranging back to the 1950s. DWD currently operates about 500 stations and provides respective data in hourly resolution.[5] Data is provided as zipped .csv files via an FTP server, with values for a single station being distributed over various files in different directories (e.g. one file for wind data of station 1234 and another file for respective air pressure and temperature data). "Historical" and "recent" data is held separately, whereas "recent" covers a rolling window of 500 days.

Luftdaten.info: Luftdaten.info is a community-driven project for collaboratively monitoring air quality through a large number of low-cost DIY stations deployed all around the world. Besides foundational dimensions like temperature, monitored measurands also include the two most-important classes of particulate matter (PM10 and PM2.5). Data is collected in a resolution between 2 and 5 minutes and provided in two different ways: current data, typically covering the last five minutes, can be accessed via an API[6] and historical/archive data is provided in .csv files via HTTP with one file per station and day, currently summing up to more than 9.500 files daily.[7]

For importing these datasets, we had to solve several challenges which will presumably also arise for many other datasets. These challenges and our respective solutions shall therefore be briefly presented below:

4.1 ID Mapping and Metadata Integrity

Every import raises the need to map between identifiers used on the external data source and the ones used internally. The most appropriate place to implement

[4] See ftp://ftp-cdc.dwd.de/pub/CDC/
[5] Recently, 10-min and 1-min data was also added for some stations but these are not imported to OpenSense.network as of yet
[6] See http://api.luftdaten.info/static/v1/data.json
[7] See http://archive.luftdaten.info/

this mapping is before it reaches the database, keeping our internal schema free from any external identifiers. OpenSense.network uses an individual ID for every single sensor, however, both data sources mentioned above only use station IDs and provide data for different measurands under said station ID. Given these circumstances, we opted for enriching the station ID with the respective measurand, thus mapping an exemplary external ID "1234-temperature" to the corresponding internal one.

In addition to this somewhat trivial issue, we also made the more challenging observation that both data sources sometimes reuse the same station ID even when the metadata of the respective station has changed, particularly in the case of location data. DWD keeps the ID of a station even when it is relocated by several hundred meters, according to the provided metadata. Stations from luftdaten.info, in turn, frequently exhibit two different types of location changes: very marginal changes that hint at differences in rounding but also stations that move several hundred kilometers, which suggests station owners have moved to another country or region. To avoid adulterated data, we could therefore not solely rely on provided station IDs and measurands in our mapping, having to constantly check for changes in location metadata. For DWD, we create a new sensor in the case of changed location. The mapping function therefore maps external IDs to *one out of multiple internal IDs*, depending on validity periods. In order to avoid the inappropriate creation of new sensors for luftdaten.info, we currently stop the import of data once a location change is observed. In the future we plan on distinguishing between marginal and significant changes so that a new sensor is only created if a notable move happens.

4.2 Data Retrieval

Both DWD and luftdaten.info provide historical/archive data in the form of multiple .csv files. As a result, our importers have to traverse through several remote directories and download files one after the other. Given the overhead associated with every such download, the overall importing performance was weaker than desired. This was especially true for the case of luftdaten.info with its (as of recently) more than 9.500 files for every covered day. In our current implementation, we did not particularly address this issue, however, we would have appreciated the availability of data in bigger files collecting all stations for a single day, or for single stations but over longer timespans. DWD does provide data in such collections, however these cover the complete lifespan of a single station, sometimes ranging back to the 1950s. To avoid overly long importing times for single stations, we resorted to importing the datasets in smaller 5-year-steps, selectively processing the same file several times. Again, we would thus have appreciated the availability of different levels of data aggregation, in this case covering shorter timespans.

As opposed to historical data, current data needs to be polled repeatedly to be provided on our platform in a timely manner. Here, DWD and luftdaten.info follow different approaches which both result in particular implementation challenges: For DWD, polling current data meant downloading and processing hun-

dreds of single files for every new polling iteration. Each of these files covers the last 500 days, causing a significant overhead and unnecessary strain on the network. For luftdaten.info, in turn, current data is provided via an API and delivered as a single json object that contains all values for all stations collected within a five-minute window. Unlike DWD's approach, this avoids redundancies and provides a more efficient retrieval of data.

Altogether, the retrieval of open environmental data from established sources resulted in significant inefficiencies emanating from the lack of different levels of aggregation. For both sources, providing data in different time and/or station aggregations would have significantly reduced network and computational load – on the importer *as well as* on the data provider side.

4.3 Typical Load Patterns

During our importing efforts, we identified two main load patterns which impact the achievable performance in different ways and call for different optimization approaches. The first pattern in particular, appeared during the import of historical data and is characterized by a large number of values being imported for a comparably small set of sensors. For example, the initial import of one file from the DWD's "recent" branch consists of 12.000 values for each of the contained 2-5 measurands. Executing a single call to our platform for every value here would have significantly impaired achievable performance. Through message coalescing, i.e. sending a configurable amount of values for the same sensor in one single message, we managed to reduce the number of (costly) HTTP calls as well as the load on the relational database. Thus, through this approach we were able to significantly improve import performance.

The second pattern, in contrast, primarily emanates from the frequent polling of larger sets of live data like in the case of luftdaten.info. Here, every single iteration results in new data for a high number of sensors with comparably few (1-3) values each. Even though we were also able to reduce the number of HTTP calls through message coalescing here, our API-design still requires several operations on the relational database before values can be posted to the time-series database. The achievable performance in this import pattern is thus constrained by the performance (and scalability) of the relational database. The load, however, arises only once per polling period of every data source. As the main focus of our platform is to *serve* data in a scalable way, we deemed this limitation acceptable for the moment. Later developments might nonetheless also explore possible optimizations in this regard.

5 Data Consumption

For consuming the open environmental data hosted by OpenSense.network, we offer two different consumption interfaces, targeting different audiences: First, we offer a web-friendly and highly scalable API with extensive query capabilities and second, we implemented a map-based user interface that allows for manual exploration and plotting of data.

5.1 Web API

The web API is the data consumption interface that provides the functionality of OpenSense.network we consider most valuable and important: In response to a simple HTTP call, it delivers data in a size-efficient and web-friendly way based on extensive query capabilities. In particular, it allows filtering on the basis of geospatial parameters (bounding-box, point-and-distance, polygon), measurands, covered timespan, value boundaries (min/max), and license-related properties. To reduce implementation complexity on the side of upstream developers and, thus, to increase accessibility, query parameters are codified as URL parameters. The following example demonstrates the simplicity of the API with a query requesting all sensor values within a 300m area for one day:

```
https://www.opensense.network/beta/api/v1.0/values?
 refPoint=52,13&maxDistance=300&minTimestamp=2018-01-30
 &maxTimestamp=2018-01-31
```

Alternatively, users can also query the available sensors according to measurands and usage rights, among other parameters. The following query retrieves the first 20 temperature sensors (measurandId=1) that grant the redistribution of data.

```
https://www.opensense.network/beta/api/v1.0/sensors?
 measurandId=1&maxSensors=20&allowsRedistribution=true
```

Complete queries can thus be shared as convenient and straightforward HTTP links. To facilitate onward processing in well-established web development frameworks, data is provided as an easily parsable single json object containing the metadata for all sensors and the respective values matching the query. Finally, we also provide an API explorer allowing interested developers to easily explore available query parameters and construct queries.[8]

As Catlett et al. [3] point out, *open* does not always mean *usable*. In this regard, OpenSense.network's web API closes a long-existing gap preventing the practical utilization of open environmental data in a multitude of use cases: Instead of having to download, laboriously preprocess and filter *open* datasets provided by various providers, developers can now find and query such data in a uniform way and retrieve it in a *usable* format compatible with modern software development tools. OpenSense.network's API interface significantly facilitates data consumption and thereby paves the way for innovative new applications and services to be built on top. For instance, websites can now easily embed weather data, society-oriented developers can build on the basis of historical particulate matter data and the platform can develop into a well-accepted data foundation for researchers.

5.2 Map Interface

To demonstrate and test the capabilities of our web API and to facilitate the ad-hoc discovery and exploration of data available on OpenSense.network, we

[8] Available at https://www.opensense.network/apidocs/

also implemented a rudimentary visual map interface.[9] Upon loading, the interface collects all sensors available on the platform and lets the user to select a measurand (e.g. PM10) to explore. Respective sensors are then shown on the map in dynamically generated clusters which can be zoomed into until single sensors become visible. After selecting one of these individual sensors, the user is prompted to specify a time period for plotting. Respective data is then fetched from the web API and presented in a human-conceivable graph.

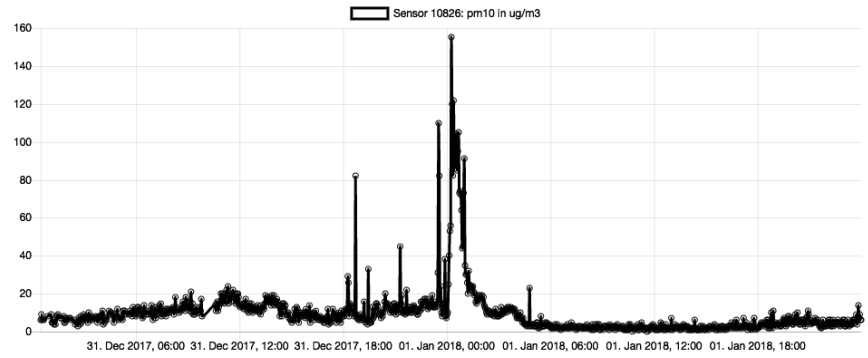

Fig. 3. Exemplary Plot for Particulate Matter around New Year's Eve 2017/2018 in Berlin

As vividly illustrated by figure 3, the user interface allows even laypersons to get enlightening insights from the data provided through OpenSense.network. It shows the historical development of particulate matter density as captured by a luftdaten.info sensor on a major street in Berlin around New Year's Eve 2017/2018 and clearly depicts the contamination caused by the extensive use of fireworks around midnight. Without the provision of historical luftdaten.info data through OpenSense.network, gaining such insights would entail manually identifying the respective sensor, downloading two .csv files, parsing their project-specific data fields and generating a graph. As demonstrated by our rudimentary map interface, the availability of respective data via our API thus allows for the easy implementation of innovative usages of existing open environmental data. These would otherwise have been hindered by excessive efforts in data retrieval and preprocessing.

6 Related Work

Throughout past years, the relevance of highly dispersed environmental data and its value for the provision of novel, innovative applications and services has been

[9] The map is available via https://www.opensense.network/

recognized by governments, in the commercial sector and by several community-driven initiatives. In the following, we briefly introduce some of these and discuss them in relation to OpenSense.network.

In particular, large efforts have been put into a multitude of governmentally driven open data repositories like the ones of the EEA (European Environmental Agency)[10] or by DWD already discussed above. These repositories are typically dedicated to the provision of official, government-collected data and do not allow for the integration of different kinds of data sources, including community-driven projects. Besides, their capabilities with regard to web-friendly data provision and discoverability are highly diverse.

Commercially driven platforms particularly include Netatmo's Weathermap[11], which is fed by hundreds of thousands of privately maintained Netatmo weather stations worldwide. Even though Netatmo provides a highly functional and freely usable map interface, it, however, does not provide the raw data publicly. Instead, the proprietary use of customer-provided data as the basis for additional commercial offers is an integral part of Netatmo's business model. Consequently, the Netatmo platform does not serve the goals pursued herein. Weather Underground[12] essentially follows a comparable approach, using amongst other things privately maintained weather stations by ambient weather.[13] Other commercial offerings such as OpenWeatherMap[14], also provide environmental data but represent weather forecasting APIs instead of providing raw data. They thus serve different goals than OpenSense.network.

As a non-commercial, community-driven project, openSenseMap also warrants a mention besides the luftdaten.info project already discussed above. This project and its public platform primarily aim at the collection of data from DIY created "senseBoxes"[15] offered to interested individuals. Information gathered by these senseBoxes includes air quality, temperature, humidity and light intensity. In comparison to OpenSense.network, the openSenseMap API exhibits several limitations, especially with regard to the availability of larger amounts of historical data (i.e. one month as a maximum time frame for latest measurements for a sensor; 2500 as a maximum count of values in one request). Given the project's primary goal of raising interest in DIY sensing and related environmental aspects in general, these limitations seem acceptable, but they stand in stark contrast to the goals pursued herein. As the data is licensed under "ODC Public Domain Dedication and License", we might, however, pursue efforts to include openSenseMaps' data in OpenSense.network in the future. OpenSense.network, in contrast, targets a much broader and large-scale monitoring of environmental data. We focus on the key scientific questions of handling and collecting large

[10] https://www.eea.europa.eu/data-and-maps
[11] https://weathermap.netatmo.com
[12] http://api.wunderground.com/api/
[13] https://www.ambientweather.com
[14] https://openweathermap.org/API
[15] http://sensebox.github.io/en/

datasets as well as the provision of web-friendly accessibility, offering a long lasting and independent delivery of IoT data.

7 Conclusion, Limitations, and Future Work

In this paper, we presented OpenSense.network, a scalable platform that provides open environmental data in a web-friendly way. OpenSense.network addresses several challenges currently hindering the development of innovative applications on top of such data. In particular, it delivers data originating from multiple sources in a single API, freeing developers from the need to collect data from different organizations themselves. Furthermore, it provides data in a uniform, well-structured, and web-friendly json format instead of the semi-structured formats typically used in existing repositories, and makes it accessible through a powerful query interface. To demonstrate the platform's capabilities and to provide easy discoverability of the hosted data, we also implemented a rudimentary map interface using our own API.

The whole platform is designed with scalability in mind from ground up, which allows it to serve huge amounts of time-series data. We achieved this through careful data modelling and, in particular through a system architecture of polyglot persistence, combining a relational database with geospatial capabilities (PostgreSQL together with PostGIS) for sensor metadata and a highly scalable NoSQL database (Cassandra) for time-series of measured data. In addition, we also implemented the front-facing API in an easily scalable manner.

As first datasets to be provided by OpenSense.network, we imported open environmental data from two particularly valuable and comparably diverse data sources: The German national weather service (DWD) and the community-driven luftdaten.info. During our importing efforts we identified several challenges, particularly regarding the inconsistent use of station IDs and the unexpected change of respective metadata in existing repositories. Also, we highlighted the drawbacks resulting from established data provision practices in matters of data retrieval.

Current limitations of our platform particularly include the limited scalability of the relational database holding sensor metadata, which especially emerged during the import of additional values for a high number of different sensors (see section 4.3). Another major limitation regards the lack of support for non-stationary sensors, which might play an important role for various interesting data sources during further development. To address these limitations, we will in the future concentrate on scalability-related optimizations in the handling and processing of metadata and examine the actual scalability limitations of our backend's relational part in more detail. The support of non-stationary sensors in line with our scalability goals, in turn, would in all likelihood require a shift of the geospatial functionality from the relational to the time-series part of our backend. A geospatial extension to Cassandra will therefore also be an important focus of our future research activities.

Additionally, we plan on increasing the capabilities of our query interface further, especially through the inclusion of server-side aggregations, allowing data consumers to specify different aggregation levels and functions (e.g. daily maximum, hourly average) in order to limit the amount of single data points that need to be transferred and processed. Finally, we hope to refine existing importers and implement additional ones – and we invite interested parties to join us in this endeavor, striving towards an evermore increasing comprehensiveness of open environmental data available through a single, scalable, and web-friendly platform.

8 Acknowledgements

We are deeply indebted to our student assistant Gereon Dusella. His implementation efforts for the visual map interface and our helpful discussions concerning the continuous development of the overall prototype are greatly appreciated.

References

1. Botts, M., Robin, A.: Opengis sensor model language (sensorml) implementation specification. OpenGIS Implementation Specification OGC **7** (2007)
2. Castell, N., Dauge, F.R., Schneider, P., Vogt, M., Lerner, U., Fishbain, B., Broday, D., Bartonova, A.: Can commercial low-cost sensor platforms contribute to air quality monitoring and exposure estimates? Environment International **99**, 293 – 302 (2017). DOI 10.1016/j.envint.2016.12.007
3. Catlett, C., Malik, T., Goldstein, B., Giuffrida, J., Shao, Y., Panella, A., Eder, D., van Zanten, E., Mitchum, R., Thaler, S., et al.: Plenario: An open data discovery and exploration platform for urban science. IEEE Data Engineering Bulletin **37**(4), 27–42 (2014)
4. Christin, D., Reinhardt, A., Kanhere, S.S., Hollick, M.: A survey on privacy in mobile participatory sensing applications. Journal of Systems and Software **84**(11), 1928 – 1946 (2011). DOI 10.1016/j.jss.2011.06.073
5. Datastax Documentation: Getting started with time series data modeling. URL www.academy.datastax.com/resources/getting-started-time-series-data-modeling. Accessed on 15.05.2018
6. Kuhlenkamp, J., Klems, M., Röss, O.: Benchmarking scalability and elasticity of distributed database systems. pp. 1219–1230. VLDB Endowment (2014). DOI 10.14778/2732977.2732995
7. Leighton, B., Cox, S.J.D., Car, N.J., Stenson, M.P., Vleeshouwer, J., Hodge, J.: A best of both worlds approach to complex, efficient, time series data delivery. In: Environmental Software Systems. Infrastructures, Services and Applications, pp. 371–379. Cham (2015). DOI 10.1007/978-3-319-15994-2_37
8. Liang, S., Huang, C.Y., Khalafbeigi, T., et al.: Ogc sensorthings api-part 1: Sensing. OGC Implementation Standard (2016). URL http://docs.opengeospatial.org/is/15-078r6/15-078r6.html. Accessed on 15.05.2018
9. Sadalage, P.J., Fowler, M.: NoSQL distilled: a brief guide to the emerging world of polyglot persistence. Pearson Education (2012)

10. Schäfer, M., Strohmeier, M., Lenders, V., Martinovic, I., Wilhelm, M.: Bringing up opensky: A large-scale ads-b sensor network for research. In: Proceedings of the 13th International Symposium on Information Processing in Sensor Networks, pp. 83–94 (2014). DOI 10.1109/IPSN.2014.6846743
11. Shadbolt, N., O'Hara, K., Berners-Lee, T., Gibbins, N., Glaser, H., Hall, W., Schraefel, M.C.: Linked open government data: Lessons from data.gov.uk. IEEE Intelligent Systems **27**(3), 16–24 (2012). DOI 10.1109/MIS.2012.23
12. Wetz, P., Trinh, T.D., Do, B.L., Anjomshoaa, A., Kiesling, E., Tjoa, A.M.: Towards an environmental information system for semantic stream data. In: Proceedings of the 28th EnviroInfo Conference, pp. 637–644 (2014)