Design and Implementation of a Scalable Crowdsensing Platform for Geospatial Data of Tinnitus Patients

Robin Kraft[1], Ferdinand Birk[1], Manfred Reichert[1], Aniruddha Deshpande[2],
Winfried Schlee[3], Berthold Langguth[3], Harald Baumeister[4], Thomas Probst[5], Myra Spiliopoulou[6], Rüdiger Pryss[1]
[1]*Institute of Databases and Information Systems, Ulm University, Germany*
[2]*Department of Speech-Language-Hearing Sciences, Hofstra University, New York, USA*
[3]*Clinic and Policlinic for Psychiatry and Psychotherapy, University of Regensburg, Germany*
[4]*Department of Clinical Psychology and Psychotherapy, Ulm University, Germany*
[5]*Department for Psychotherapy and Biopsychosocial Health, Danube University Krems, Austria*
[6]*Department of Technical and Business Information Systems, Otto-von-Guericke-University Magdeburg, Germany*
{*robin.kraft, ferdinand.birk, manfred.reichert, ruediger.pryss,*}*@uni-ulm.de*
aniruddha.deshpande@hofstra.edu, winfried.schlee@gmail.com, berthold.langguth@medbo.de
harald.baumeister@uni-ulm.de, thomas.probst@donau-uni.ac.at, myra@ovgu.de

***Abstract*—Smart devices and low-powered sensors are becoming increasingly ubiquitous and nowadays almost all of these devices are connected, which is a promising foundation for crowdsensing of data related to various environmental phenomena. Resulting data is especially meaningful when it is related to time and location. Interestingly, many existing approaches built their solution on monolithic backends that process data on a per-request basis. However, for many scenarios, such technical setting is not suitable for managing data requests of a large crowd. For example, when dealing with millions of data points, still many challenges arise for modern smartphones if calculations or advanced visualization features must be accomplished directly on the smartphone. Therefore, the work at hand proposes an architectural design for managing geospatial data of tinnitus patients, which combines a cloud-native approach with Big Data concepts used in the Internet of Things. The presented architectural design shall serve as a generic foundation to implement (1) a scalable backend for a platform that covers the aforementioned crowdsensing requirements as well as to provide (2) a sophisticated stream processing concept to calculate and pre-aggregate incoming measurement data of tinnitus patients. Following this, this paper presents a visualization feature to provide users with a comprehensive overview of noise levels in their environment based on noise measurements. This shall help tinnitus or hearing-impaired patients to avoid locations with a burdensome sound level.**

***Keywords*-mHealth; crowdsensing; tinnitus; geospatial data; architectural design**

I. Introduction

When utilizing mobile crowdsensing for environmental phenomena like (urban) noise pollution, beneficial insights for health-conscious people can be revealed. This is especially true for humans with chronic diseases like tinnitus [5], [6]. Noise exposure is one example that affects tinnitus patients in their daily life. Being able to tell which places are above a certain threshold in terms of the average loudness, would make it possible to avoid these places and therefore unhealthy noise exposure. Smartphones as well as stationary sensors in public places can be used to collect noise measurements and store them to a central backend, which calculates averages and delivers the aggregated data back to the smartphones of the users. Then, for example, the received data can be used to highlight places with an unhealthy noise exposure by proper visualization features. However, in order to provide high scalability and the ability to store, aggregate, and deliver geospatial data in an efficient way, conventional approaches using monolithic backends and classical relational databases are not suitable. Taking such scenarios into account, we conceived an architectural design for a crowdsensing platform that utilizes a cloud-native approach, combined with Big Data and stream processing concepts. The architectural design and the implemented prototype show that still mHealth crowdsensing scenarios exist for which no proper technical settings can be found. On the other, the work at hand shows that geospatial crowdsensing data can be valuable for tinnitus patients in their daily life. Note that a general treatment does not exist for tinnitus patients and they crave for any valuable method to better cope with their symptoms. Therefore, the work at hand aims to utilize the power of the crowd to create helpful noise level maps of a region. To conclude, the contributions of this work are:

- By using geospatial mobile crowdsensing data of tinnitus patients, a visualization feature can be established that shows noise exposure regions on a map. Patients can use this map to avoid burdensome places.
- An architectural design is proposed that enables the addressed scenario in an efficient manner. In particular, the proposed *mobile context* enables efficient data calculations for the visualization feature.
- The architectural design shall serve as a foundation for

other related mHealth scenarios.

The remainder of this paper is organized as follows. In Section II, relevant related work is presented, while Section III discusses required background information. The architectural design is presented in Section IV. Its peculiarity, the measurement context, is then discussed in Section V. Insights and impressions into the implementation are presented in Section VI, whereas Section VII concludes the paper with a summary and an outlook.

II. Related Work

Mobile crowdsensing platforms for urban sensing and crowdsensing-based noise maps have been already considered. The authors of [1] apply a participatory sensing approach using an Android application and an urban sensing platform to enhance the spatial and temporal data resolution of noise pollution in cities. The mobile application leverages the smartphones' microphones and GPS-sensors to perform location-related noise measurements and sends this data to an open urban sensing platform. Users can then access this data and generate real-time noise maps and data graphs. [2], in turn, uses crowdsensing with Android devices for a noise monitoring platform and acoustic urban planning in smart cities. A web-based visualization application is able to suggest certain noise reduction interventions to city planners and helps them to meet European laws and regulations. Both platforms, however, offer no visualization of the measured noise data directly on the mobile application and focus on web-based approaches, where scalability, data quota, and energy consumption are not a priority. Crowdsensing architectures that deal with efficiency and context-awareness have been also proposed in the past [3]. The aggregation of sound measurements in the mHealth context, in turn, was not addressed by these approaches. Crowdsensing platforms that leverage their opportunities for chronic disorders, especially for the tinnitus disease, have already been proven their feasibility [4], [5], [6]. However, to the best of our knowledge, none of the existing tinnitus approaches consider geospatial data in combination with crowdsensing as we do in the work at hand.

III. Background Information

Mobile crowdsensing is a paradigm that becomes more and more prevalent in the mHealth context [4], [7]. In combination with Ecological Momentary Assessments (EMA) [8], it has shown to be able to reveal new medical insights [5], [6]. As an example, we developed TrackYourTinnitus (TYT), a mobile crowdsensing platform for the tinnitus disease [4]. Tinnitus can be described as the phantom perception of a sound. The symptoms, in turn, are subjective and vary over time. Hence, TYT was realized to monitor and evaluate the variability of symptoms over time based on EMA and mobile crowdsensing. Importantly, tinnitus is a chronic disorder with a high economic burden; $10 - 15\%$ of the population worldwide suffer from tinnitus. As a general treatment does not exist, patients crave for any valuable experiences and methods to better cope with their symptoms in daily life. One counter measurement constitutes the avoidance of loud locations as it is reported that often after a concert, patients suddenly get a tinnitus or their already existing symptoms get worse. Therefore, this projects aims to exploit the power of the crowd to create noise level maps of a region. Users shall be enabled to measure the current sound level using their smartphones and mobile crowdsensing techniques. If such noise levels maps can be established with a high reliability, then tinnitus patients may use the map to avoid loud locations.

Designing a crowdsensing platform that is able to collect, process, store, and deliver data for these types of applications comes with several challenges. First, the platform should be able to handle many concurrent requests for both incoming and outgoing measurement data without a loss of performance. Therefore, it should be scalable and, in the best case, also elastic. Second, the platform should be able to efficiently process geospatial data, with potentially millions of data points. In general, computations on geospatial data are a complex endeavour due to costly operations on fine-grained coordinates in order to aggregate geographically and hierarchically related data. As a consequence, it is important to choose an efficient concept for indexing and aggregation of geospatial data. Third, the platform should be conceived in a generic way so that it is able to handle different kinds of geospatial crowdsensing data (e.g.; noise pollution, air pollution, or traffic information).

IV. Technical Approach

First of all, Table I shows the core functions that were identified for the mobile crowdsensing platform enabling a noise level map for tinnitus patients. During the requirements analysis, we identified that each function should be bounded to a context. Following this, we can develop respective microservices that can be flexibly changed if new context requirements occur. Therefore, each function in Table I is mapped to one of the five identified bounded contexts *User Identity*, *Social*, *Measurements*, *Incentives*, *Communication*, and *Sensors*. Although a bounded context is technically represented through a microservice, it can become necessary to use more than one microservice for a bounded context. Therefore, the platform supports different patterns to technically represent a bounded context through microservices.

In order to enable a scalable and efficient processing of concurrent noise measurement requests, we focus on a *cloud-native* approach. A *cloud-native application (CNA)* is intentionally designed to be operated in the cloud. Therefore, such application is by design distributed, elastic, and horizontally scalable. Furthermore, it is composed of microservices with a minimum of isolated states [9]. In our prototypical implementation, we use different microservices

#	Function	Bounded Context
1.1	Let users register and authenticate with the backend.	User Identity
1.2	Let users change their password and provide lost-password recovery.	User Identity
1.3	Let users deactivate and delete their user account.	User Identity
2.1	Let users maintain a *User Profile* with personal information.	Social
2.2	Let users join groups and start, follow and contribute to discussions.	Social
2.3	Provide geospatial relations of groups and discussions.	Social
2.4	Trigger a notification to the user on new contributions in subscribed discussions or subscribed areas of interest.	Social
3.1	Collect measurements provided by smartphones and other IoT-devices and streamline them as a common input stream.	Measurements
3.2	Aggregate the measurements to provide min-, max-, and average values within certain geospatial areas and time-based windows.	Measurements
3.3	Allow geospatial request filtering by specifying the area of interest and time windows.	Measurements
3.4	Allow access to single stored measurements with a pagination like limitation for the number of results.	Measurements
3.5	Provide an API that returns the results in a common geospatial format to allow straightforward visualization features with commonly used frontend technologies.	Measurements
4.1	Track user contributions for authorization of additional functionality and to provide a feature that users can evaluate their progress.	Incentives
4.2	Maintain awards and streaks for certain achievements that motivate users to continue in contributing measurements.	Incentives
5.1	Inform users about certain events via email.	Communication
5.2	Inform users about certain events via push-notifications.	Communication
5.3	Let the user define preferences for the type of events he or she likes to be informed.	Communication
6.1	Manage meta-information about statically deployed sensors.	Sensors

Table I: Core business capabilities of the platform mapped to bounded contexts

with *Docker*[1] as container technology and *Kubernetes*[2] as orchestration system. We combine the cloud-native approach with a *stream processing* in order to enable decoupled processing of incoming geospatial data. Stream processing is a programming paradigm, for which data is continuously read from an unbounded dataset (*stream*) in an asynchronous, non-blocking manner [10]. We use the *Kafka Streams* library as stream processing implementation due to its compatibility to *Apache Kafka*[3]. Apache Kafka is a publish and subscribe messaging system that allows for publishing generic types of (stream) data. Thereby, services can publish certain events to *Kafka Topics* and every interested service can act as a consumer that polls messages from such topics [10]. In order to tackle the challenges of an efficient storage, processing, and hierarchical aggregation of geospatial data, we make use of the *Discrete Global Grid Systems (DGGS)*. "A DGGS is a spatial reference system that uses a hierarchical tessellation of cells to partition and address the globe", as defined by the Open Geospatial Consortium (OGC) [11]. The system represents a series of discrete global grids, where each grid encompasses an increasing number of cells with respect to its predecessor grid and therefore having a finer resolution. DGGSs come with the advantage that they cover the whole spherical surface of the earth and can therefore be used to partition data collected anywhere on the planet. We further use *Uber*'s *Hexagonal Hierarchical Spatial Index* (H3)[4] as a DGGS implementation, which allows to represent the same data efficiently in differently sized buckets, which, in turn, is important in order to aggregate (and visualize) data on different scales. Note that H3 allows us to determine in which bucket a specific geospatial location has to be placed and, inversely, to calculate the boundary of each bucket if we know its index.

The overall architecture of the platform is shown in Fig. 1. The central *Measurement API Services* handle incoming measurements from mobile devices and forward them to the stream processing with Apache Kafka. Internet of Things (IoT) sensors can contribute their data directly to Kafka via the Message Queuing Telemetry Transport (MQTT) protocol. The other services consume raw or transformed measurement data through these services. This process is described in detail in Section V. The *Authentication Services* handle user authentication and user authorization for all other services that are access-restricted. Sensor as well as social, incentive, and communication data are stored independently and managed through their respective services.

V. Measurement Context

This section discusses the measurement context. The latter is of utmost important and key factor to create a noise level map. To be more precise, mainly the question emerges in what way noise measurements should be represented and aggregated in an efficient manner. First of all, Fig. 2 shows the elaborated document-based (NoSQL) data model for noise measurements used for the *Measurement Context*. Measurements and aggregations are modeled as GeoJSON [12] *Simple Features* using *Type*, *Geometry*, and

[1]https://www.docker.com/
[2]https://kubernetes.io/
[3]https://kafka.apache.org/
[4]https://eng.uber.com/h3/

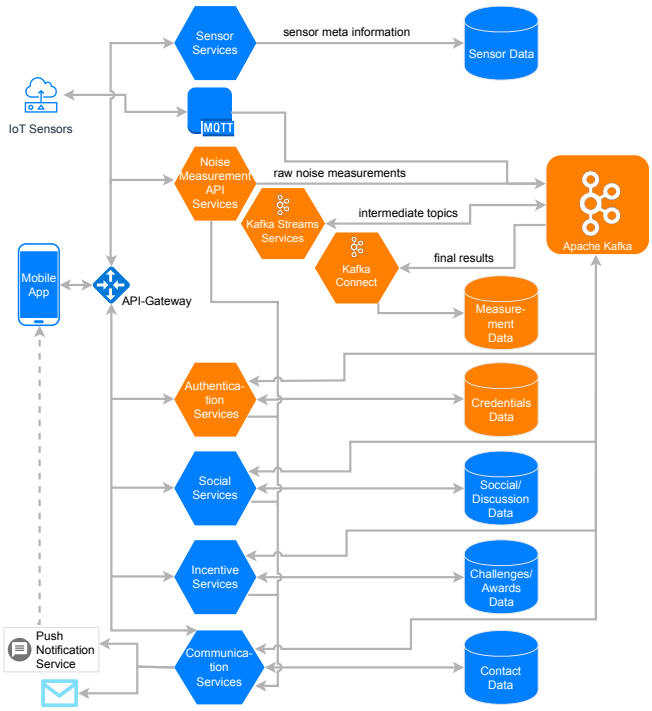


Figure 1: Architecture of the crowdsensing platform. The components depicted in orange are part of the proof-of-concept implementation

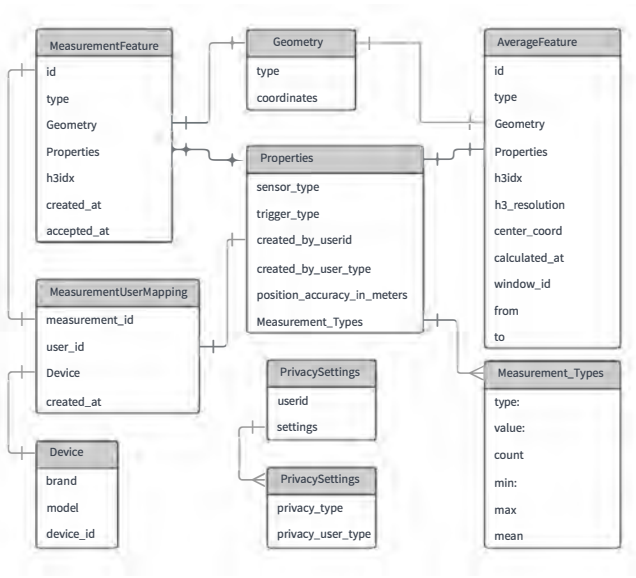


Figure 2: Data Model for Noise Measurements

Property attributes for better compatibility with different geo-libraries and data-storage in MongoDB[5]. Note that the latter inherently supports indexing of GeoJSON structures. Each noise measurement is assigned with an unique *id*, and additionally geo-indexed using the H3 library. Measurement payload is stored in *Properties*, containing – amongst other attributes – the type sensor and trigger of the measurement and one or multiple *Measurement_Types* (e.g., different noise measurements like LAeq, LCPeak and TWA [13]). One such type can be either related to a single measurement, in which case only type and value are of relevance, or it can be related to an *AverageFeature*, for which minimum, maximum, mean, and count values are stored. *AverageFeature* is used to store aggregations for a specific geographical area (i.e., a hexagon), and over a specific time span. For privacy reasons, any user-related data is stored separately from the measurement data.

The realized data flow of measurements in the *Measurement Context* is shown in Fig. 3.

Steps 1–4 represent the *Data Ingress Phase*, which are briefly introduced: In Step 1, the mobile application sends its measurement records to an endpoint of the Ingress Service, whereas in Step 2, the records are checked for validity and the endpoint handler attaches timestamps as well as a *userid* (if the user is authenticated). In Step 3, the measurement is published to the Kafka topic *noise-raw-measurements*, while in Step 4, a confirmation message is sent to the mobile application if the measurement is valid. Following this, the *Stream Processing Phase* is accomplished in Apache Kafka in Steps 5–12, which are discussed in the following: In Steps 5–7, *the Preparator* is processing all messages from the *noise-raw-measurements* topic, validates the measurements, and anonymizes them for privacy reasons. Additionally, a H3-index with the resolutions 10 (smallest) and 5 (intermediate) is calculated for the coordinates of the measurements. The indexed and anonymized data is then published to new Kafka topics. In Steps 8–10, the *Aggregator* is taking the data from the previous steps as input and calculates averages for the different resolutions and different time-windows. A time-window is defined by a window-length (e.g., 15 minutes) and a retention-time (i.e., how long the window is refreshed if measurements arrive late). Averages are calculated with respect to the logarithmic scale of decibels and, once again, published to different Kafka topics. In Steps 11–12, in order to allow for an efficient querying of the produced results in partitioned topics in the Kafka Cluster, their data is persisted to a *MongoDB* with Kafka Connect. Finally, data is accessed and prepared for visualization in the *Data Access Phase* in Steps 13–17, which do the following: In Step 13, the mobile applications sends a request to the RESTful API of the *Access Service* for a specified H3 resolution, time-window, and geo-boundary; In Step 14, optionally, authorization is performed for access-restricted routes, while in Step 15, data is loaded from the corresponding databases, making use of MongoDB's geospatial indexes. In Step 16, final aggregations are performed if the requested data is not already pre-aggregated and privacy filters are applied (e.g.,

[5]https://www.mongodb.com/

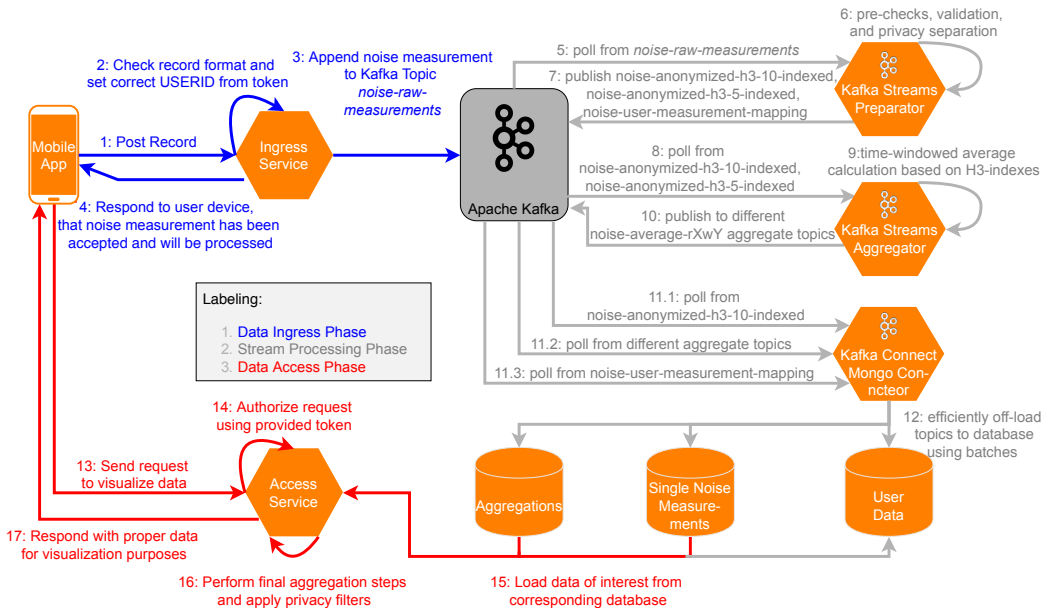


Figure 3: Data Flow of Noise Measurements

to specify that only the user itself can see his or her own raw measurements). Finally, in Step 17, data is sent back to the mobile application for visualization purposes.

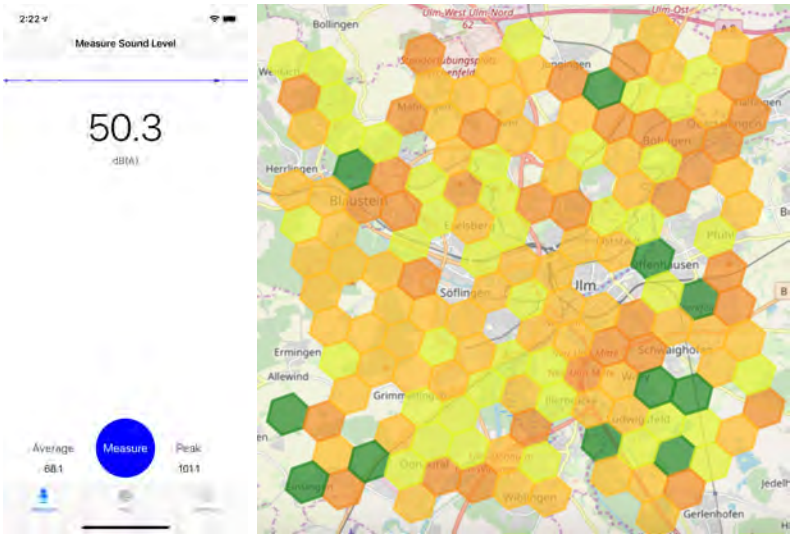
VI. Proof-of-Concept Prototype

We implemented a proof-of-concept prototype, which is briefly discussed in the following. Selected screenshots of the prototype are shown in Fig. 4. The mobile application (see Fig. 4a) shows the current sound level in A-weighted decibels (i.e., dB(A)). Noise measurements can be performed by pressing the *Measure* button. The *Equivalent continuous A-weighted sound level (LAeq)* and the *C-weighted peak sound level (LCpeak)* are (1) then calculated over a time period of 30 seconds, (2) displayed to the user, and (3) finally sent to the backend (either directly or delayed if there is currently no connection to the server). The backend handles these measurements as described in Section V. Both the mobile application and a website (see Fig. 4b) are then able to access aggregations for different zoom levels (i.e., different H3 resolutions). Following this, the mobile app or the website can provide a proper visualization to the user, indicating the noise exposure for a certain area by the use of a color gradient between harmless (green) and harmful (red).

Currently, the proof-of-concept prototype is tested with users in the region of Ulm, Germany. First results indicate that users like the application and see its benefits. However, for the mobile application side, at present, we solely implemented an iOS mobile application. Therefore, an Android application must be developed as well. The reason why we opted only for iOS for the first release is related to the fact that noise measurements are more reliable on iOS with respect to the analysis and interpretation phases. As Android implies several hardware vendors compared to iOS, the evaluation of collected microphone results might differ among different Android vendors and therefore becomes more complex. However, the integration of Android smartphones will be a crucial future step to represent the majority of smartphone users. Moreover, the iOS app will be revised based on the feedback of the users as well as new features will be added. Regarding the latter, for the first version of the app, features like incentives were not realized. Additionally, large-scale performance tests are needed in order to evaluate the scalability results of the platform.

VII. Summary and Outlook

This work presented an approach for a mobile crowd-sensing platform that is able to process noise measurements of crowd users and their smartphones with the goal to establish a noise level map. The calculated map indicates noise exposure for a certain area by the use of a color system and zoom levels. The latter become possible through our conceived measurement context that stores and aggregates noise measurements by the use of a sophisticated stream processing pipeline. Based on a proof-of-concept prototype, first study results indicate that users welcome the overall procedure and usability of the platform. However, we also discussed technical improvements that must be addressed to finally provide a technical solution that can be reliably used in different practical scenarios. Additionally, in order to evaluate the scalability of the system, large-scale performance tests in comparison to state-of-the-art architectures should be performed. In the context of the tinnitus disease, the discussed solution may help patients to avoid burdensome



(a) Mobile application (b) Website

Figure 4: Screenshots of the mobile application and the website showing the noise level map

places based on the noise level map. While using the proof-of-concept prototype in practice, many more useful features could be revealed for tinnitus patients. For example, users want to fill out a tinnitus-related questionnaire when accomplishing a noise measurement. Based on these information, users want to retrospectively relate their recorded noise levels with data of the tinnitus questionnaire to learn more about their daily tinnitus fluctuations. From a technical perspective, the realization of the discussed features show that a complex technical architecture and infrastructure are needed. The resulting solution, in turn, can be utilized in other mHealth context as well. For example, in the context of migraine, weather-related factors could be measured. Altogether, we see mobile crowdsensing still in its infancy for scenarios in the mHealth context. On the other, approaches like shown in this work indicate that mobile crowdsensing can be a promising technology for mHealth scenarios.

References

[1] I. Schweizer, T. Darmstadt, F. Probst, R. Bärtl, T. Darmstadt, M. Mühlhäuser, T. Darmstadt, A. Schulz, and T. Darmstadt, "Noisemap - real-time participatory noise maps," in *In Second International Workshop on Sensing Applications on Mobile Phones*, 2011.

[2] M. Zappatore, A. Longo, and M. A. Bochicchio, "Crowdsensing our Smart Cities: a Platform for Noise Monitoring and Acoustic Urban Planning," *Journal of Communications Software and Systems*, vol. 13, no. 2, p. 53, Jun. 2017.

[3] P. Jayaraman *et al.*, "Cardap: A scalable energy-efficient context aware distributed mobile data analytics platform for the fog," in *East European Conference on Advances in Databases and Information Systems*. Springer, 2014, pp. 192–206.

[4] R. Pryss *et al.*, "Mobile Crowdsensing Services for Tinnitus Assessment and Patient Feedback," in *IEEE Int'l Conf on AI & Mobile Services*. IEEE, 2017, pp. 22–29.

[5] T. Probst *et al.*, "Emotional States as Mediators between Tinnitus Loudness and Tinnitus Distress in Daily Life: Results from the "TrackYourTinnitus" application," *Scientific Reports*, vol. 6, 2016.

[6] W. Schlee, R. Pryss, T. Probst, J. Schobel, A. Bachmeier, M. Reichert, and B. Langguth, "Measuring the moment-to-moment variability of tinnitus: the trackyourtinnitus smart phone app," *Frontiers in Aging Neuroscience*, vol. 8, 2016.

[7] R. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: current state and future challenges," *IEEE Communications Magazine*, vol. 49, no. 11, 2011.

[8] U. Ebner-Priemer and T. Kubiak, "Psychological and psychophysiological ambulatory monitoring," *European Journal of Psychological Assessment*, vol. 23, no. 4, pp. 214–226, 2007.

[9] N. Kratzke and P.-C. Quint, "Understanding cloud-native applications after 10 years of cloud computing - A systematic mapping study," *Journal of Systems and Software*, vol. 126, pp. 1–16, Apr. 2017.

[10] N. Narkhede, G. Shapira, and T. Palino, *Kafka: The Definitive Guide: Real-time Data and Stream Processing at Scale*. " O'Reilly Media, Inc.", 2017.

[11] M. Purss, "Topic 21: Discrete Global Grid Systems Abstract Specification," p. 63, 2017.

[12] RFC 7946 - The GeoJSON Format. [Online]. Available: https://tools.ietf.org/html/rfc7946

[13] V. Novin, S. Givehchi, and H. Hoveidi, "Assessing the occupational noise in workplaces at local levels," 2014.