GreenMov: A Fiware Based Interoperable Solution to Reduce the Environmental Impact of Mobility

Benoit Couraud[1,2(✉)], Mehdi Nafkha[1], Franck Dechavanne[1], Azeddine El Youssfi[1], and Paulo Moura[1]

[1] Université Côte d'Azur, IMREDD, Nice, France
benoit.couraud@glasgow.ac.uk, m.nafkha@cotedazurfrance.fr
[2] James Watt School of Engineering, University of Glasgow, Glasgow, UK

Abstract. The recent advancements in the Internet of Things (IoT) have facilitated the deployment of numerous applications that enhance urban intelligence by improving the monitoring and control of city assets, such as lighting systems, traffic signals, and public transportation. However, these applications are often tailored to the specific needs of individual cities, limiting their replicability in other locations primarily due to the lack of interoperability in communication signals between assets. Despite various initiatives to establish standards for interoperability, real-world implementations frequently fall short of achieving fully interoperable systems that can be universally replicated, largely due to the absence of comprehensive solutions and implementation examples. This paper presents a fully interoperable use case for Green Mobility solutions in a smart city, utilizing air quality, traffic, and noise intensity data to provide transport recommendations to end-users. The implementation employs the NGSI-LD standard, Fiware data storage tools, and developed artificial intelligence-based algorithms to predict the transport situation for the following day and offer relevant traffic recommendations. This work has resulted in the development of several data models and standardized forecast algorithms with accuracy exceeding 75% on the noise and traffic datasets at our disposal, thereby enabling the potential for replication in other locations.

Keywords: Data models · forecasting services · green mobility · Fiware

1 Introduction

The deployment of IoT devices has enabled a variety of new use cases across sectors such as health, agriculture, energy, and industry [10]. Among these, mobility stands out as particularly significant due to its profound impact on daily life, economic growth, social inclusion, and the environment [8]. Consequently, numerous projects have emerged to enhance urban mobility by reducing environmental impacts, minimizing time spent in transit, and decreasing pollution (air

Published by Springer Nature Switzerland AG 2025
G. Rey et al. (Eds.): IFIPIoT 2024, IFIP AICT 737, pp. 261–278, 2025.
https://doi.org/10.1007/978-3-031-81900-1_16

quality, noise, environment). IoT can address these challenges by monitoring air quality, enabling smart traffic management, and providing transport recommendations to end-users. Start-ups, companies, and municipalities have developed solutions aimed at mitigating the environmental and social impacts of transportation. However, these initiatives often remain as proof-of-concepts and are not easily replicable in other locations, primarily due to interoperability issues between solutions and differing local infrastructures [11]. The variability in sensor technologies necessitates customization to ensure compatibility with existing systems, and the lack of standardized data formats for communication between sensors and controllers further complicates integration. To tackle this interoperability challenge, the Fiware EU initiative advocates for the standardization of data storage through smart data models [4] that encompass various smart city domains, including energy, environment, and mobility. A data model provides a conceptual representation of the data structures required by a database or information system, defining how data is connected, stored, and processed. This blueprint facilitates the building of databases and enhances data management and analysis [7]. Adoption of standardized data models ensures interoperability between data-consuming services, such as decision-making tools, thereby promoting broader and more effective implementation of smart city solutions [9,12].

However, the adoption of a smart data model necessitates specific requirements and architectural considerations [6]. In this paper, we propose utilizing Fiware smart data models to design an end-to-end solution aimed at reducing the environmental impact of mobility in the city of Nice. This work is part of the GreenMov European project [1], which seeks to leverage Fiware Smart Data Models for green mobility solutions. By building on existing data models for mobility, this project extends these models and incorporates Artificial Intelligence (AI)-based services that utilize air quality, noise, and mobility data to provide relevant mobility recommendations to end-users and city councils for managing daily transport needs. The primary outcomes of this paper include a replicable architecture for green mobility solutions in urban environments, an extension of smart data models to facilitate green mobility services, and an example of an AI service designed to forecast the impact of mobility on noise annoyance.

Section 2 outlines the green mobility use case addressed in this work. Section 3 details the end-to-end architecture from sensors to end-users, while Sect. 4 describes one of the AI-based services developed for this use case.

2 Case Study Description: Green Mobility in Nice

Nice, the main city on the French Riviera, is significantly influenced by its transportation infrastructure, which is vital for both its bustling tourism industry and daily urban life. Meanwhile, the Nice Côte d'Azur Metropolis monitors air and noise pollution in specific areas along with traffic congestion. The Metropolis aims to enhance its existing data by collecting, consolidating, standardizing,

and making it interoperable to allow a seamless integration of future use cases and services. The objective of Nice Metropolis and of the initiative described in this work is to improve residents' quality of life by reducing pollution, particularly from road traffic, through various strategies such as alternative routes, improved public transport options, and other solutions. The resulting solution could be replicated in other regions due to its standardized and high-quality data format, potentially becoming the standard for green mobility projects.

In the GreenMov project, the Nice Côte d'Azur Metropolis wants to leverage its sensors infrastructure to improve the quality of transport, and reduce its impact on the environment. The objectives are multifold:

- Reduce the impact of transport on the environment (air quality, climate change, noise)
- Reduce the traffic congestion in Nice
- Provide recommendations to citizens and city administration.

To accomplish these goals, the following use case is proposed: By monitoring air quality, noise levels, weather conditions, and traffic intensity, the GreenMov project aims to predict environmental pollution and traffic conditions. This enables the project to offer recommendations to both the city and end-users, such as promoting increased use of public transportation or encouraging the use of shared bikes over personal vehicles. Two scenarios have been outlined to illustrate this use case more effectively.

2.1 Scenarios

Option 1. Antoine plans to drive to work from Cannes to Nice next Friday morning at 9 AM. On Thursday evening, he receives a notification from the Nice-Traffic App recommending that he use public transportation or bikes instead. The message indicates that air quality in Nice will be degraded on Friday morning and suggests using the public transportation or bikes available at that specific time, provided by the city of Nice.

Option 2. The city of Nice assesses Friday's Air Quality Index forecast alongside traffic predictions. Based on the forecasted conditions, additional electric buses and bicycles are deployed to accommodate commuters driving to Nice, and targeted messages are sent to regular Friday commuters.

Thus, this use case necessitates establishing an IoT infrastructure to monitor air quality, noise levels, and traffic patterns; ensuring data storage in an interoperable format using Fiware NGSI-LD (Next Generation Service Interface) data models standardized by ETSI; implementing AI-based forecasting services for traffic, noise, and air quality predictions; and offering recommendations based on anticipated future conditions.

An overview of the GreenMov's use case in Nice is depicted in Fig. 1, while Fig. 2 illustrates the deployment locations of all sensors.

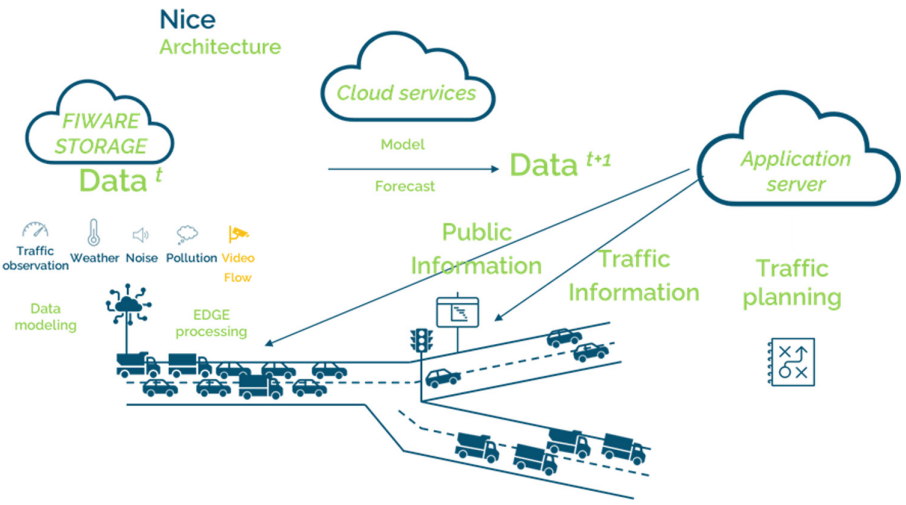

Fig. 1. General Architecture of GreenMov's Nice use case.

2.2 Data Sets

The data sets from Nice to be used in this use case implementation are available on the European Data Portal as well as various European, national, and local city portals. The use case requires data to evaluate air quality at specific locations, noise pollution, traffic activity, and real-time availability of alternative transportation solutions, such as buses, trams, or shared bikes. Consequently, six main categories of datasets are needed: air quality measurements, weather measurements, noise measurements, traffic measurements, public transportation availability, and bike availability. To enhance accuracy and coverage, multiple data sources were utilized for some of these datasets.

The primary Fiware NGSI-LD data models characterizing the collected data are: Traffic Flow Observed, Vehicle Emission Label, Air Quality Observed, Air Quality Monitoring, Weather Observed, Noise Level Observed, Noise Pollution, Public Transportation, and Bikes Availability.

The implementation of this use case also led to the development of new data models for green mobility, such as Noise Pollution, Traffic Environmental Impact, and several forecasting models specifying data forecasted for specific future times (e.g., Air Quality Forecast, Air Pollution Forecast), as well as Bikes and Public Transport Availability.

3 Interoperable IoT Architecture

Fiware and the GreenMov project have established a generic architecture that can serve as a foundational framework for various implementations. Figure 3 illustrates this architecture, which adopts a multi-layer structure.

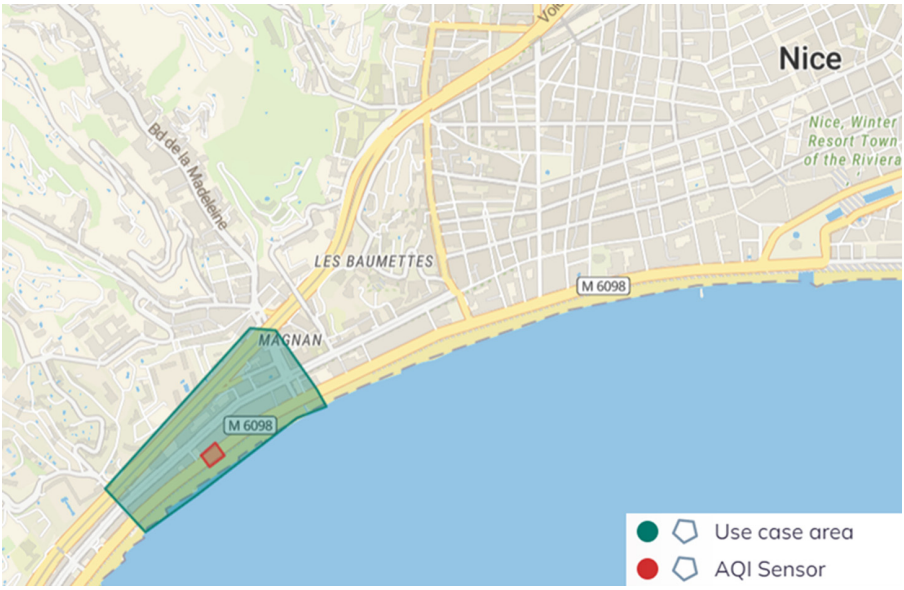

Fig. 2. GreenMov's Nice use case area.

At the base, the first layer comprises sensors that provide raw data. Parallel to the sensors, there are third-party and cloud services that supply relevant data for use cases, such as weather forecasts. The second layer consists of aggregation platforms, including proprietary platforms that directly gather data from the sensors, potentially using their own data models.

The third layer is composed of core Fiware solutions for data routing and storage. This layer includes the context broker, which ensures data consistency with existing data models and stores the data in appropriate databases. These databases can be real-time, such as those based on MongoDB, or historical, storing time-series data, such as PostgreSQL databases. API tools for historical databases include QuantumLeap (primarily for NGSI-v2), Cygnus, Draco, and Mintaka. The real-time brokers proposed in the generic architecture are OrionLD and Scorpio. Deployment can utilize container technology such as Docker or Kubernetes.

The application layer, next, encompasses all data analysis aspects, primarily based on historical data. This layer includes dashboards, data processing, and other analytical tools.

Finally, in parallel of these layers, we can find sub-services such as securing data access, using Keycloack and Keyrock identity and access management. Also, the addition of a LDES (Linked Data Event Stream) adapter can help in the supply of time series data by speeding up the process of data retrieval. Based on this generic architecture, the use cases described in Sect. 2 implemented their own specificities. The rest of this section describes these specificities.

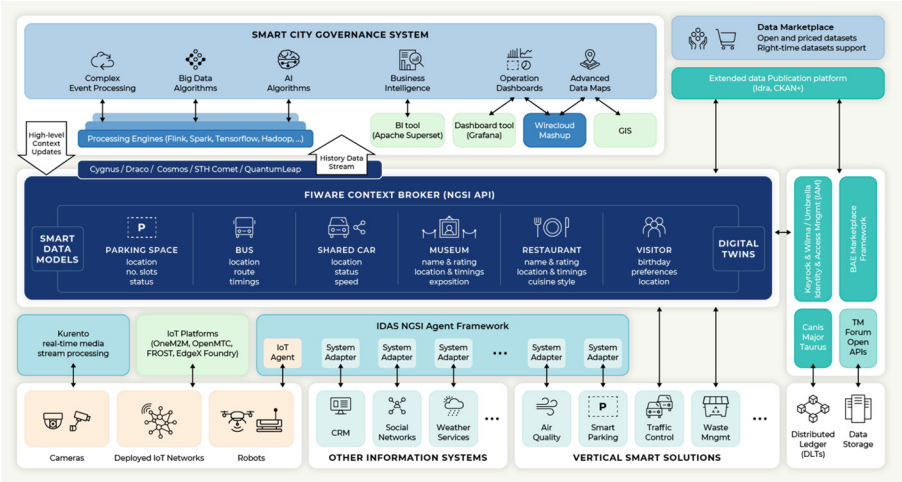

Fig. 3. General Architecture of Fiware implementation, from [3].

3.1 Adaptation to the Implementation of GreenMov Use Case in Nice

The architecture of Nice follows the architecture proposed in Fig. 4.

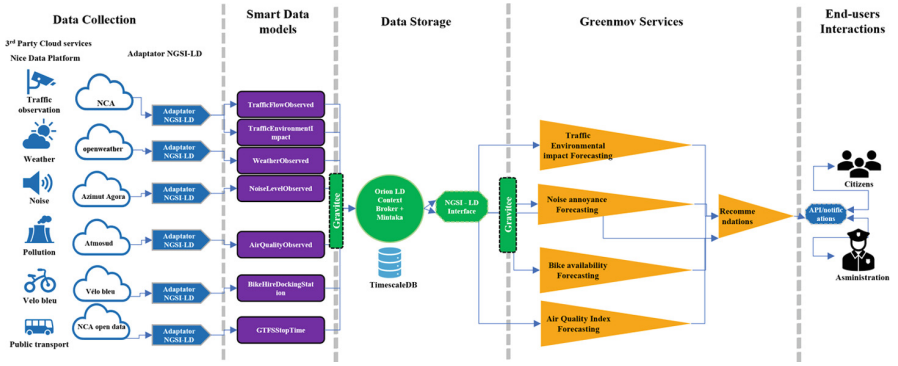

Fig. 4. Data Architecture of GreenMov's implementation in Nice.

The first layer, detailed on the left-hand side of Fig. 4, represents the data collection layer along with adapters that enable conversion to the NGSI-LD format. In the case of Nice, several sensors are utilized, including those for air quality monitoring, weather, noise, bike availability, transport availability, and traffic monitoring. Once this data has been translated into the appropriate data model, it is sent to the storage layer, which incorporates the Fiware context

broker and storage technology. For the Nice use case, the context broker is Orion-LD, used in conjunction with Mintaka, and data storage is managed through PostgreSQL using TimescaleDB, an efficient time-series data storage solution. In addition to the APIs available in the generic architecture (Mintaka and Orion-LD), we implemented an API manager, named Gravitee, which provides several functionalities:

- Route renaming/routing, which allows clients (such as the services) to use different addresses than the standard context broker addresses to access the data.
- Also, it is worth noting that another benefit of Gravitee is the fact that the presence of one specific word (such as "temporal") in the URL of request sent by a third party can route the request to Mintaka instead of OrionLD. Everything is made transparent to the end-user who only need to know the URLs for data posting or retrieval.
- The last main advantage of Gravitee is the control of access to data. First, Gravitee implements a catalogue of available datasets to which any user can subscribe. But on top of this, it allows the data owners to grant access to some of the data, through the use of an API-Key specific to the end-users or to the application defined to access the dataset.

Data Collection. Several components were developed to collect the required data for the use case. These components were implemented in Node-RED and achieve the following tasks:

- Traffic observation: a component that collects and stores the traffic intensity directly from Nice metropolis measurements at specific locations. The software component implemented in Node-RED selects the last data, converts it into NGSI-LD format, and sends it to the context broker of the storage facility.
- Weather: a Node-RED flow leverages the European meteostat JSON API to retrieve weather forecast from Nice, converts it into NGSI-LD, and sends it to the context broker.
- Noise: For noise, using the API from Nice Côte d'Azur metropole, a nodered flow retrieves the noise data every day, extracts it, converts it into NGSI-LD, and sends it to the context broker.
- Air quality: Similar to the noise data, air quality is retrieved daily by a nodered flow that extracts the air quality information from the selected sensor, converts it into NGSI-LD, and sends it to the context broker.
- Bike availability: a Node-RED flow extracts bikes availability data available in the core of the velobleu stations website. From the webpage code, the nodered flow accesses the bikes availability data per stations, extracts the required bikes stations, converts the data into NGSI-LD, and sends it to the context broker.
- Public transport: a Node-RED flow retrieves the Nice Côte d'Azur metropole public transport GTFS data on a daily basis, convert it into NGSI-LD for the selected lines, and sends it to the context broker.

Data Storage. For data storage, the standard Orion-LD Docker image was employed to replicate the storage architecture, incorporating TimescaleDB and Mintaka. Additionally, several other components were integrated, including Gravitee as an API manager to offer a dataset catalog, enable re-routing (replacing standard Orion-LD URLs with custom URLs), and secure data access through API keys. An NGINX component was also added to serve as a proxy, directing different services based on the URL route.

Services. For the implementation of the services, several software components were designed and developed. These services were deployed in a container to facilitate replication in future use case locations. The services encompass the forecasting of air quality, traffic intensity, bike availability, and noise annoyance, as well as providing recommendations to citizens and the city council.

Figure 5 presents a generic architecture for most of the forecasting-related services. Each forecasting service includes an API to receive requests for the service's output, which is managed through kserve or FastAPI components. Most services incorporate an AI model that requires training, typically using the scikit-learn *fit* function. The trained model is then used to compute forecasts, which are processed by a *formula computation* component to generate actionable information. For instance, the air quality index forecasting service predicts future particle concentrations, which are then used by a computation service to calculate the future air quality index.

The services run on a Docker image, and the AI models are trained on a regular schedule (daily or weekly). An additional sub-service, MLFlow, operates in parallel to enable the service operator to monitor the machine learning components. Lastly, all services include HTTP client request components to retrieve the latest data from the storage facility's context broker, which is then used to generate short-term forecasts.

Front End Implementation. Front-end services are available to end-users of Nice. An interface was designed to facilitate user interaction with the services. Users can access and utilize the traffic recommendation services directly through the interface. Additionally, individual services outputs can still be retrieved through specific requests. Therefore, the interface for end-users is the WebApp that is displayed in Fig. 6. End users can use the slide in the middle of the interface to select the time for which they want a traffic recommendation. It goes up to one day ahead, although we could extend it to several days. Figure 6 shows an example of the interface when there is an event of high air quality pollution and high noise annoyance. In this case, a recommendation follows the logic of the traffic recommendation service and results in advices to use bikes and public transportation depending on the weather forecast.

The rest of the interface is constituted of graphs showing the accuracy of the forecasts along with the shape of future noise annoyance or traffic evolutions. Figure 6 shows these different visualisations proposed to the end-users, along with a map on which users can find where the bikes are available.

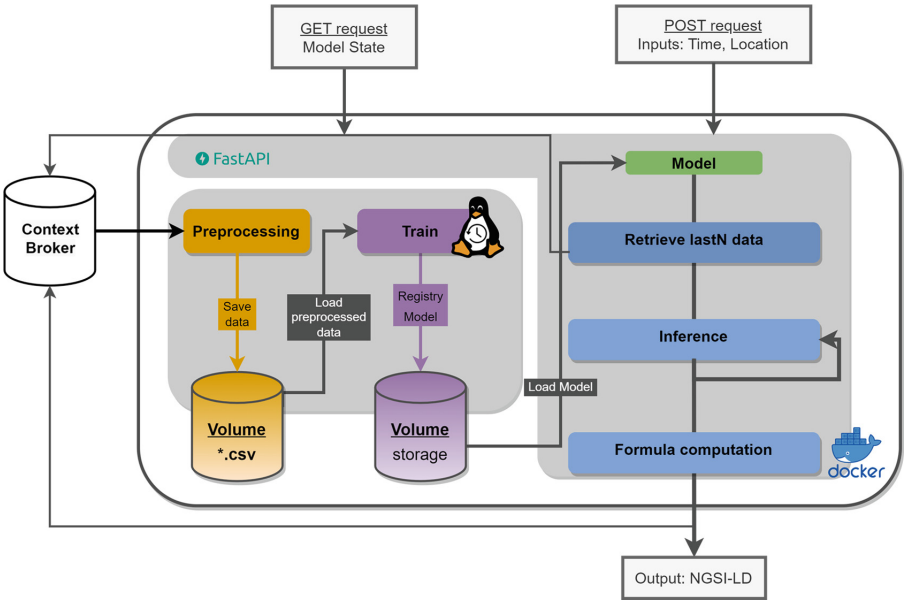

Fig. 5. General Architecture of Fiware & AI based Services.

Fig. 6. Nice Use case front end.

Infrastructure. All the software components and storage facility were implemented in the University Côte d'Azur's owned servers located in IMREDD building's own data centre. This is depicted in Fig. 7 that highlights the two main components of the Nice use case architecture: on the right hand side, the main server that is used for data storage and services implementation, whereas the server on the left hand side is the one responsible for data collection, NGSI-LD formatting and sending data to the storage facility.

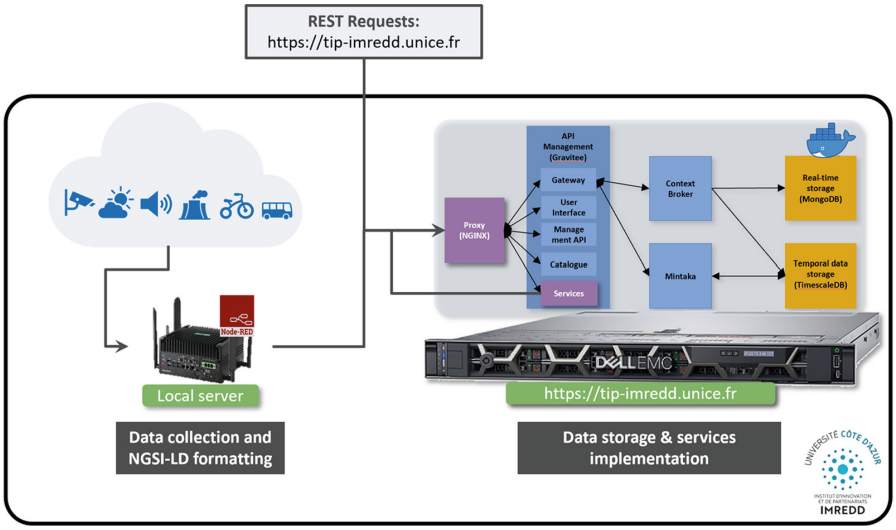

Fig. 7. Infrastructure of the Server implementation.

4 Forecasting Services

Services have been described shortly in the previous section. However, this section proposes to detail one specific forecasting service (noise annoyance forecasting) to showcase what can be done with city data and state of the art libraries. The same applies to other forecasting services, such as air quality, traffic, and bikes availability, although they might require other inputs such as weather forecasts. Noise annoyance forecasting is a service that predicts the level of noise disturbance in a specific location within a specified time frame. The aim of this service is to provide information about future noise intensity so it can be used to compute its expected impact. This noise intensity forecast is then sent to the noise annoyance calculation service for prediction of future noise annoyance, which will then be sent to the traffic recommendation generation service, as depicted in Fig. 8.

The rest of this section describes the process that was followed to provide accurate forecasts.

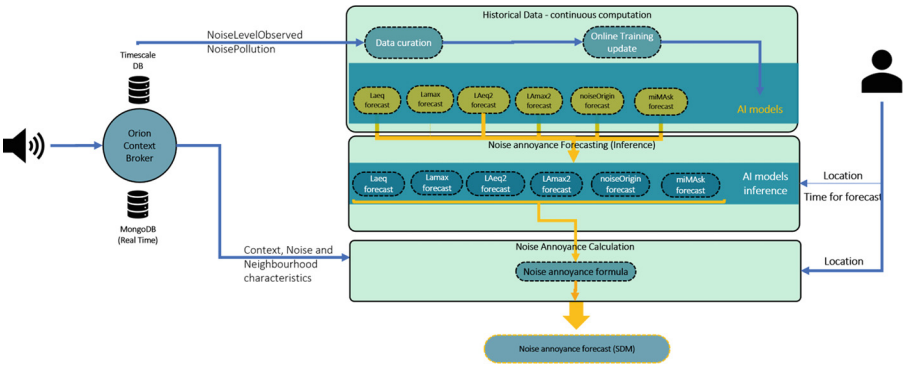

Fig. 8. Overview of the whole noise annoyance forecasting services.

4.1 Data Analysis

Historical Data. The historical data sets involved in the training of the AI model and the implementation of the noise annoyance forecasting service play a crucial role in its accuracy. In the case of Nice use case, the data set includes the noise levels in a specific location (Sensor with the following coordinates: 43°40'56.4"N 7°13'58.1"E) from 01/01/2020 until 2022. This data provides a comprehensive view of the noise situation in the area over a two-year period and can be used to train the AI model to make accurate predictions about future noise levels.

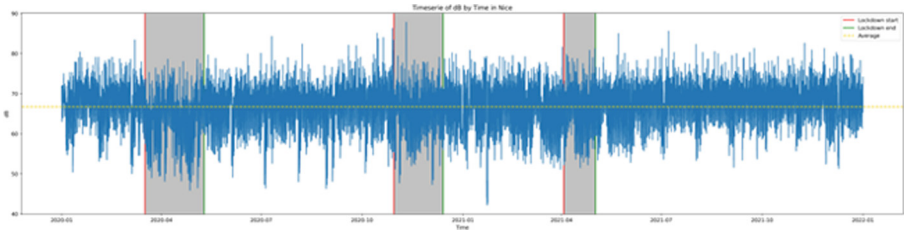

Fig. 9. Initial Noise dataset used in the use case of Nice.

Figure 9 shows the records used in the project, where Covid lockdown periods were highlighted in order to show that not all the available data can be used in the study. It shows that the training dataset of the forecasting models might require to be adjusted and benchmarked in order to avoid negative impacts from period of time that will never happen again.

Preprocessing. Before the data can be used in the calculation of noise annoyance, it is important to perform a thorough cleaning and pre-processing process

to ensure accuracy and reliability of results. This process involves reviewing the data for any errors, discrepancies, or missing information, and correcting or removing these issues as necessary.

4.2 Model Design

The development of the noise forecasting service is essential to ensure its accuracy and effectiveness. The AI model must efficiently analyze historical traffic datasets, account for various factors influencing noise intensity, and predict future noise levels. To achieve this, the service must undertake several key steps, including data preparation, model selection, and model training. The appropriate AI model must handle the complexity of the data and the numerous factors affecting noise. This involves evaluating multiple AI models and selecting the one that delivers the most accurate predictions. In the case of traffic forecasting, models such as K-Nearest Neighbors (KNN), Decision Tree, and Random Forest were benchmarked. KNN outperformed the others, achieving an accuracy above 85%, which met the requirements of the GreenMov project. Consequently, further custom models were not explored.

Regarding features, a correlation study identified the following inputs for the model: type of day (weekday or weekend), hour of the day, and noise levels at various lags (1 h, 2 h, 3 h, 24 h, 72 h, and 168 h prior). Weather forecasts were found to be uncorrelated with noise values.

The model outputs a single noise intensity value for the next hour. This output can then be used iteratively to generate noise forecasts for subsequent hours by feeding the predicted noise intensity at h+1 as an input to predict the noise intensity at h+2.

In line with Green AI principles and to minimize data and computing power requirements, we conducted a study to analyze the necessity of large datasets. For our use case, the benchmarking results, as illustrated in Fig. 10, indicate that the size of the training dataset impacts forecast accuracy. A training dataset of two weeks provides sufficient accuracy for the GreenMov project's forecast accuracy requirements (>75% r^2).

Results. Figure 11 displays the r^2 scores for the different AI models used, and for different sizes of training dataset, with a maximum r^2 accuracy of 89%, and an average accuracy of 80.3%. These figures were obtained using a train/test split of 70%. Figures below show graphically the comparison between real measurements and predictions. We can see in Fig. 11 that there is a significant gap between forecast and measurements for the day before the last day, but these are due to a measurement error, given the fact that the recorded values were considerably low compared to all previous historic data.

4.3 Model Deployment

Once the model was designed, it was deployed within our infrastructure. This subsection outlines the process followed for service deployment.

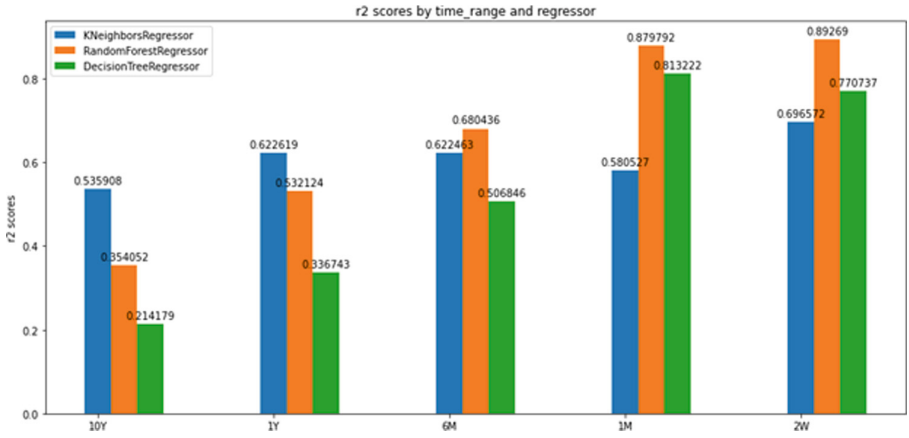

Fig. 10. Results of AI models benchmark for the use case of Nice.

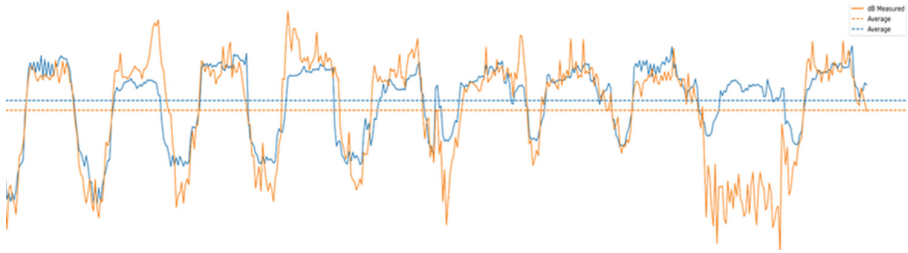

Fig. 11. Forecast (blue) vs Ground Truth (orange) for one week in 2022. (Color figure online)

The workflow leading to the final model is illustrated in Fig. 12. Initially, data was gathered for analysis to determine the noise forecasting model. This data was then cleaned (as detailed in the pre-processing section) and augmented with complementary information, such as data from the past few hours and the type of day.

During the model creation phase, various models, features, and training requirements were evaluated to identify the optimal combination. A separate model was designed for each location. Models tested included K-Nearest Neighbors (KNN), Decision Tree, Random Forest, and XGBoost. The selected features (inputs) included the type of day and previous noise levels (from the last hours up to 168 h ago). Training could use data spanning several years or be restricted to a few weeks or months to avoid capturing seasonal specificities, such as those from summer, when forecasting for winter. Consequently, a benchmark of training dataset sizes was conducted to determine the optimal size for the training dataset. This benchmarking led to a specific combination of model, training size, and features used in the final noise forecasting service.

In the rollout phase, we leveraged the architecture, inputs, and training requirements to provide a real-time solution that forecasts noise levels at the required times. The steps for computing a forecast are as follows:

- First, from the requested location of the forecast, the model that corresponds to the location is selected.
- Then, the last N data are requested to the context broker in order to retrieve the noise intensity at the previous times (last hour, last 2 h, last 24 h, ...).
- The noise level of the next hour is predicted.
- Using this forecast as an input of the next model's inference, the model is used to compute the noise intensity forecast in the next 2 h.
- This new forecast is used as an input to forecast the noise level in 3 h.
- And this process is repeated until the time of the requested forecast is reached. The result of the final forecast (for the requested time) is then formatted as NGSI-LD and fed back to the user.

Finally, at the end of each day, the daily data is retrieved from the context broker, and the model is retrained with the updated training dataset. This updated dataset comprises the previous training dataset with the data from the last day added and the oldest day's data removed. The updated model parameters are then stored using joblib for use in the following day's forecasts.

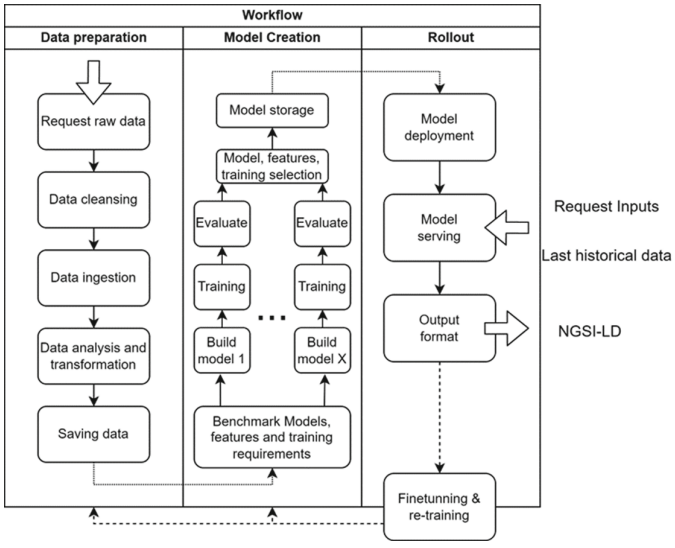

Fig. 12. Workflow for Noise forecasting service for each location.

In the next subsection all the technical aspects and selected technologies to accomplish the functions described are further discussed.

4.4 Noise Annoyance Calculation

The noise annoyance calculation service aims to provide a comprehensive assessment of the noise environment in a specific area, identifying when and where noise levels and sources are unacceptable. Key parameters include the type of noise source, which helps pinpoint contributors to noise pollution such as traffic, industrial activities, or construction sites. Another critical parameter is the average age of residents in the area, as different age groups exhibit varying sensitivities to noise. This information was collected for the locations of the sensors used in the Nice use case.

The noise intensity level is also essential, with the service calculating noise annoyance using various models, including the A-weighted equivalent continuous sound pressure level (LAeq). This calculation offers an assessment of the noise impact on residents, which can significantly affect their health, well-being, and quality of life.

To summarize, data requirements to compute the noise annoyance in an area involve the following: Area: The area of the calculation of the annoyance; Period of time: The measurement time; Type of area (residential, industrial, commercial); Noise level; Noise level classification; Average age level in the area; Dominant noise source in the area.

In order to make the calculations, values are aggregated from the data sets and a mathematical formula uses these values for the calculation of the noise annoyance index. This formula is inspired from the noise disturbance calculation tool developed within IKCEST - International Knowledge Centre for Engineering Sciences and Technology under the Auspices of UNESCO [2]. The formula used states that the annoyance level is the sum of the *Noise level value*, the *average age level value*, the *Noise source value* and the *Type of area value*. It requires to convert all the qualitative data (type of noise source) into quantitative data, using tables as shown in Fig. 13 and Fig. 14 for the output conversion. The outputs from the calculation are shown in the Figure below.

Noise sources (based on the area)	Value
Industrial and construction	2
Road and air traffic	2
Enetrtainement and commercial	1.5
Domestic	1

Average age level	Value
0-35	1
35-50	1.5
50	2

Noise level	Value
40-50 dB	1
50-60 dB	2
60-65 dB	3
65-70 dB	4
70-80 dB	5
> 80 dB	6

Type of area	Value
Residential	2
Commercial	1.5
Mix	1.5
Industrial	1

Fig. 13. Noise annoyance calculation parameters translation.

Using the values and the outputs from the Figures above, we can make a calculation example: For an industrial area with average age level of 40 and a noise level of 55 db the noise annoyance calculation is: $2 + 1.5 + 2 = 5.5$ Which corresponds to a moderate noise annoyance.

Noise annoyance	Value (from to)
Very calm	0-1
Calm	1-2
Good	2-3
Acceptable	3-4
Medium	4-5
Moderate	5-6
Annoying	6-7
Very annoying	7-8
Unsupportable	8-9
Dangerous	9-10
Very Dangerous	over 10

Enviromental impact	Value
Good	2
Medium	5
Moderate	6
Unhealthy	7
Dangerous	8
Extremely dangerous	Over 9

Fig. 14. Noise annoyance calculation outputs.

Finally, the internal architecture of the noise annoyance computation component is proposed in Fig. 15. A docker container enables an easy replication of the service. The architecture includes the application of the formula proposed in the previous section along with a local database to store the parameters of the formula.

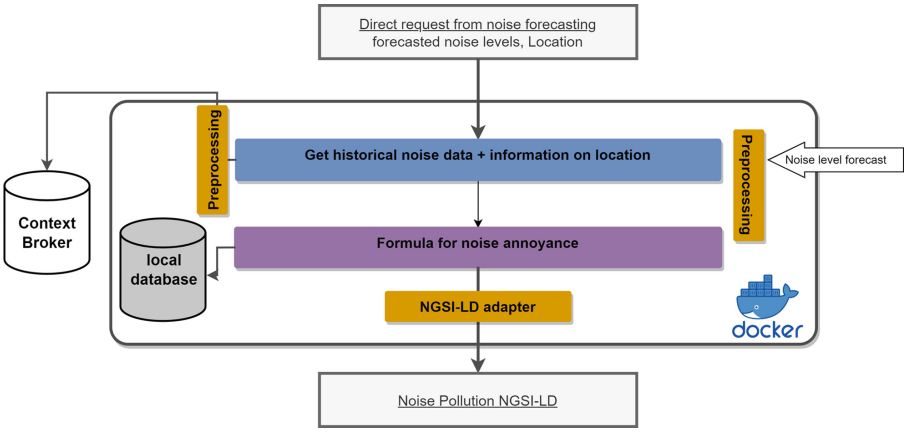

Fig. 15. Noise annoyance calculation architecture.

Following the workflow highlighted in Fig. 15, the noise annoyance calculation service receives the request directly from the noise forecasting service along with the location and the forecast for the future noise levels. Then, the service will request the context broker to get access to noise pollution information for the considered location. This provides information such as the building type, the noise origin, etc. Using this information, the formula described above is used to compute the noise annoyance index, that is then encapsulated into the NoisePollutionForecast smart data model and sent back to the noise forecasting service.

Although noise annoyance is a subjective variable, it can be computed based on local characteristics, such as the building types and source of noise, and based on the noise level. The output of this service is then converted into NGSI-LD format [5], and will then be used to automatically generate traffic recommendations that can help reducing the annoyance from traffic intensity.

5 Conclusion

The recent development of IoT has led to many applications that often use custom data models and fail to bring interoperability in a system that could be replicated at different locations if it was designed in an interoperable way. To address this issue, smart data models have been proposed and supported by the European commission through the NGSI-LD initiative standardized by ETSI. However, implementing an end-to-end use case based on Fiware smart data models still requires adaptation and development to extend data models to meet the use case requirements. In this work, we proposed an architecture to replicate an end-to-end NGSI-LD implementation that we applied to the GreenMov use case in Nice that aimed to provide recommendations to end-users and city council to reduce the environmental impact of mobility. Data from sensors had to be converted into the right NGSI-LD format before being stored in Fiware NGSI-LD compliant storage architecture. Then, several application services were proposed, among which forecast services that leverage NGSI-LD data to train machine learning models. An example was given with a noise annoyance forecasting model that manages to forecast day-ahead noise intensity levels with an accuracy greater than 75%. This use case and work demonstrates the relevance for interoperability solutions in order to provide replicable solutions that can be deployed in any country in the world.

Acknowledgments. This study was funded by the European GreenMov project, and this work was carried out on the technological platform "Smart City Innovation Center" funded with the support of the European Union through the European Regional Development Fund, the Metropolis Nice Côte d'Azur, the Department of Alpes Maritimes, the Region Sud Provence Alpes Côte d'Azur, and the State, particularly within the framework of the Initiative of Excellence of the Future Investments Program.

References

1. Green Mobility data models and services for smart ecosystems. https://green-mov.eu/
2. On-line calculation of noise pollution level (LNP) in impact assessment. https://www.gislite.com/app/sc444
3. Smart Data Models Reference. https://smartdatamodels.org/
4. The Fiware Smart Cities Reference Architecture. https://www.fiware.org/about-us/smart-cities/
5. Guidelines for modelling with NGSI-LD. Technical report, ETSI (2021). https://www.etsi.org/images/files/ETSIWhitePapers/etsi-wp-42-NGSI-LD.pdf

6. Carnevale, L., Galletta, A., Fazio, M., Celesti, A., Villari, M.: Designing a fiware cloud solution for making your travel smoother: the fliware experience. In: 2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC), pp. 392–398 (2018). https://doi.org/10.1109/CIC.2018.00059
7. Cirillo, F., Solmaz, G., Berz, E.L., Bauer, M., Cheng, B., Kovacs, E.: A standard-based open source IoT platform: fiware. IEEE Internet Things Mag. **2**(3), 12–18 (2019). https://doi.org/10.1109/IOTM.0001.1800022
8. Echeverría, L., Gimenez-Nadal, J.I., Molina, J.A.: Green mobility and well-being. Ecol. Econ. **195**, 107368 (2022). https://doi.org/10.1016/j.ecolecon.2022.107368. https://www.sciencedirect.com/science/article/pii/S0921800922000301
9. Pham, V.C., Makino, Y., Tan, Y.: A fiware IoT agent for echonet lite protocol. In: 2020 IEEE 9th Global Conference on Consumer Electronics (GCCE), pp. 235–239 (2020). https://doi.org/10.1109/GCCE50665.2020.9291983
10. Ramson, S.J., Vishnu, S., Shanmugam, M.: Applications of internet of things (IoT) - an overview. In: 2020 5th International Conference on Devices, Circuits and Systems (ICDCS), pp. 92–95 (2020). https://doi.org/10.1109/ICDCS48716.2020.243556
11. Santa, J., Bernal-Escobedo, L., Sanchez-Iborra, R.: On-board unit to connect personal mobility vehicles to the IoT. Procedia Comput. Sci. **175**, 173–180 (2020). https://doi.org/10.1016/j.procs.2020.07.027. https://www.sciencedirect.com/science/article/pii/S1877050920317063. The 17th International Conference on Mobile Systems and Pervasive Computing (MobiSPC),The 15th International Conference on Future Networks and Communications (FNC),The 10th International Conference on Sustainable Energy Information Technology
12. de la Vega, F., García-Martín, J.P., Santos, G.P., Torralba, A.: Implementation of a fiware-based integration platform and a web portal as aids to improve the control of ships navigation in a river. In: 2020 IEEE International Symposium on Systems Engineering (ISSE), pp. 1–3 (2020). https://doi.org/10.1109/ISSE49799.2020.9272242