Green mobility data models and services for smart ecosystems

D5.3 Pilot deployment and validation v2

Document Identification	
Contractual Delivery Date	31/08/2023
Actual Delivery Date	31/08/2023
Responsible Beneficiary	IMREDD
Contributing Beneficiaries	ATOS, HOPU, IMEC, FF, AIV
Dissemination Level	PU
Version	1.0
Total Number of Pages:	86

Keywords
Experimentation, Green mobility services, Smart City, Standardization

This document is issued within the frame and for the purpose of the GreenMov project. This project has received funding from the European Union's Innovation and Networks Executive Agency – Connecting Europe Facility (CEF) under Grant AGREEMENT No INEA/CEF/ICT/A2020/2373380 Action No: 2020-EU-IA-0281. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission.

(*) Dissemination level. -PU: Public, fully open, e.g., web; CO: Confidential, restricted under conditions set out in Model Grant Agreement; CI: Classified, Int = Internal Working Document, information as referred to in Commission Decision 2001/844/EC.

Document Information

Related Activity	Activity 5	Document Reference	D5.3
Related Deliverable(s)	D5.1, D5.2	Dissemination Level (*)	PU

List of Contributors	
Name	**Partner**
Benoit Couraud	IMREDD
Mehdi Nafkha	IMREDD
Filip Gosselé	IMEC
Ismail Kutlu	IMEC
Juan Antonio Martínez Navarro	MURCIA
Christie Jaux	EGIS
Ignacio Sevillano	ATOS
Gert Degreef	IMEC

Document History			
Version	**Date**	**Change editors**	**Changes**
0.1	29/06/2023	IMREDD	First version of the deliverable, ToC, work distribution
0.2	09/07/2023	IMREDD	Contributions in progress
0.3	19/07/2023	IMREDD/IMEC/ATOS/HOPU/EGIS	Contributions
0.6	19/8/2023	DV/IMEC/HOPU	Contributions
0.8	27/08/2023	IMEC	Contributions
0.9	28/08/2023	IMREDD	Contributions / Formatting
0.9	28/08/2023	ATOS	Quality Check

Document History			
Version	**Date**	**Change editors**	**Changes**
1.0	31/08/2023	ATOS	FINAL VERSION TO BE SUBMITTED

Quality Control		
Role	**Who (Partner short name)**	**Approval Date**
Reviewer1	IMEC	31/07/2023
Reviewer 2	ATOS	21/08/2023
Quality manager	ATOS	29/08/2023
Project Coordinator	ATOS	31/08/2023

Table of Contents

List of Tables

List of Figures

List of Acronyms

Abbreviation / acronym	Description
AI	Artificial Intelligence
AIV	Agentschap Informatie Vlaanderen
API	Application Programming Interface
AKS	Azure Kubernetes Service
AQI	Air Quality Index
AQC	Air Quality Calculation
AQF	Air Quality Forecasting
AQ	Air Quality
CB	Context Broker
CURL	Client for URL
DCAT-AP	Data CATalogue Application Profile
Dx.y	Deliverable number y belonging to Activity x
EC	European Commission
EGM	Easy Global Market
GBFS	General Bikeshare Feed Specification
GHG	Green House Gas
GTFS	General Transit Feed Specification
IoT	Internet of Things
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
LAeq	A-weighted equivalent sound level
LD	Linked Data

Abbreviation / acronym	Description
LDES	Linked Data Event Stream
ML	Machine Learning
NCA	Nice Côte d’Azur
NGSI	Next Generation Service Interfaces
NiFi	NiagaraFiles (Apache Software Foundation)
OSLO	Open Standards for Linked Organisations
URL	Uniform Resource Locator
UX	User Experience
YAML	YAML Ain't Markup Language

Executive Summary

This document presents and details the deployment steps of GreenMov 's use cases. It includes a description of the software and infrastructure architectures, and details how the storage and the services have been deployed in each of the three use cases, which are located in Nice, Flanders and in Murcia/Molina de Segura. This document also includes a description of the validation plans for all use cases.

Leveraging the knowledge from the first deployment of data models, storage solutions and services designed within GreenMov, this document provides an updated version of all the outputs from GreenMov. It explains how each of these solutions have been implemented in each of the use cases, and therefore how it can be replicated. In addition, this deliverable proposes an updated and final value for the KPIs of activity 5 and of the validation of each use case and their associated services. Indeed, this deliverable includes also a functional and technical validation plan for each service as well as a replication process.

As a summary, this document allows the reader to understand GreenMov 's services and most importantly their architecture and interdependency, but also the underlying technology behind the implementation and the validation plan of each city use case.

1 Introduction

Green mobility services are core of the GreenMov project, and their deployment is specific to the use cases. Therefore, 3 use cases were set up in GreenMov to test and validate these services and the data models and the data storage architecture that enable these services. These three use cases ensure a replication of such services at a larger/different scale.

This deliverable compiles results from Activities 2 “Smart Data Models for green mobility”, 3 “Smart services for green mobility”, 4 “Architecture for Context Broker enhancement in concurrent data intensive scenarios as mobility” and 5 “Pilots deployment” and explains how each of these have been implemented during the demonstration proposed in Activity 5’s use cases. This document details the services, the required infrastructure and software architecture, as well as each use case specificities. The deployment process is also described as well as the validation procedure established by the pilots’ leaders.

This document is complementary to activity 3 deliverable “D3.2 Green Mobility Services Development” [1] and the activity 4 deliverable “D4.2 GreenMov Reference Architecture and guidelines” [2] and “D4.3 GreenMov Source Selection Building Blocks” [3] and is the second version of the “D5.2 Pilot deployment and validation v1”. The deliverables mentioned above propose the reference documents for the services and architectures, whereas this deliverable focusses on their implementation within the use cases.

In Section 2 of the document, we describe the use cases and the KPI’s adopted. In Section 3, we provide the general architecture along with use cases’ specificities. In Section 4, we summarize the data models that are used in the use cases. In Section 5, we detail the services implementation, describing the process followed for services deployment. In section 6, we present the deployment process, including the technical and the non-technical requirements, such as the involvement of stakeholders. Finally, in section 7, we describe the validation plan of the use cases implementation, as well as the final version of the Key Performance Indicators. In Section VIII, we conclude this deliverable by describing open datasets that were published in the European open data portal.

1.1 Purpose of the document

This document describes the data models, services and GreenMov architecture deployment processes in the three pilots. It also includes feedbacks from the deployment of the solutions, as well as a validation of the other Activities results such as Activity 3.

1.2 Relation to other project work

This is the third deliverable for the Activity 5, which is in the centre of the project deployment, Consequently, there has been a natural relation with deliverable D5.1 [4] Requirements for the data sets and mobility services and D5.2 Pilot deployment and validation v1. Similarly, as it describes how data models were implemented, it links directly to deliverable D2.1[5] and D2.2 [6] Extended Smart Data Models. Also, this deliverable explains how services and storage infrastructures were deployed. Therefore, it relates directly to deliverables D3.1 [7], D3.2 [1] on Green Mobility Services Definition and development, but also D4.1 [8] and D4.2 [2] on GreenMov Reference Architecture and guidelines.

1.3 Structure of the document

This document is structured in 2 major categories:

- **Chapter 2, 3, 4 and 5** present the three pilots, the architecture, the data models used and their services respectively.
- **Chapter 6, 7 and 8** present the deployment process as well as the validation of such deployment. Finally, the last section presents the Metadata validation tool, used to validate the datasets created within GreenMov.

1.4 Glossary adopted in this document

- Linked data. It is structured data, which is interlinked with other data, so it becomes more useful through semantic queries.
- Orion. Software solution for context information management compliant with the NGSIv2 specification, created by Telefónica, FIWARE Foundation and other entities, available as a Generic enabler of the FIWARE platform [9].
- Orion-LD. Software solution for context information management compliant with the NGSI-LD specification, created by Telefónica, FIWARE Foundation and other entities, available as a Generic enabler of the FIWARE platform [10].
- Scorpio. Software solution for context information management compliant with the NGSI-LD specification, created by NEC and other entities, available as a Generic enabler of the FIWARE platform [11].
- Stellio. Software solution for context information management compliant with the NGSI-LD specification, created by EGM and other entities, available as a Generic enabler of the FIWARE platform [12].
- PostgreSQL. PostgreSQL is a free and open-source object-relational database management system known for its reliability, scalability, and extensive features for handling complex data.
- QuantumLeap. QuantumLeap is an open-source software framework for building scalable and efficient Internet of Things (IoT) data infrastructures. It provides a unified interface for managing and querying large amounts of time-series data from different IoT devices and platforms.
- MongoDB. MongoDB is a cross-platform document-oriented NoSQL database program that uses JSON-like documents with optional schemas.
- Mintaka. Mintaka is a software solution that utilizes the NGSI-LD temporal retrieval API, which is implemented on top of the Orion-LD Context Broker as its underlying database.
- Cygnus. Cygnus is a flexible and scalable data collection and forwarding agent that collects data from various sources, including IoT devices, and forwards it to other systems. It is part of the FIWARE open-source platform and is often used in combination with the Orion Context Broker for managing IoT data.
- Kubernetes. Kubernetes is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications.
- Gravitee. Gravitee.io is an open-source API management platform that provides developers with tools for creating, managing, and securing APIs. It includes features such as API documentation, analytics, rate limiting, and access control, making it a popular choice for organizations looking to build and manage APIs at scale.

- TimescaleDB. TimescaleDB is a high-performance, scalable, and open-source time-series database that is designed to handle large volumes of time-stamped data with ease. It is built on top of PostgreSQL, and provides features such as automated data retention policies, real-time analytics, and native support for SQL.
- CrateDB. CrateDB is a distributed database management system that combines a SQL interface with a highly searchable document-oriented data store.
- MapBox. Mapbox is a platform that provides custom online maps for websites and applications.
- Draco. Draco is a is an easy to use, powerful, and reliable system to process and distribute data.
- Kafka. Kafka is a distributed streaming platform developed by Apache that allows for the processing and analysis of large volumes of data in real-time. It is widely used for building real-time data pipelines, event-driven architectures, and streaming applications.
- Node-RED. Node-RED is a visual programming, flow-based, low-code development tool initially created by IBM with the purpose of connecting hardware devices, APIs, and online services for Internet of Things applications. It offers a web browser-based flow editor to build JavaScript functions within the system.

2 Use Cases

This section briefly presents the three use cases adopted in the cities Nice, Murcia and Molina and the Flanders region and the KPIs associated.

2.1 Nice

2.1.1 Description

The use case in the city of Nice aims to reduce the impact of road traffic on urban pollution. This is achieved through noise and air pollution prediction models, predictions of the impact of traffic on the environment, and the availability of bikes and public transport. The main benefits of the use case are:

- Support the decisions of urban transport managers (in particular by allocating more public transport or increase the availability of shared bikes).
- Respond to citizen behaviour by proposing alternative green mobility services.
- Quantification of greenhouse gas (GHG) and noise emissions from traffic in the territory.

2.1.2 KPIs

Table 1: Main identified KPI's for Nice.

Indicator name	Description	Quantification range	Target	Frequency
Number of recommendations per day	Refers to the number of recommendations provided by the services implemented, A recommendation can either be a short-term recommendation to change transportation for public transport, or to not change anything.	Highest is best.	24	Everyday
Accuracy of forecast services	Accuracy of the services forecasting models.	[0-100%] 100% is best	75%	Everyday
Number of stakeholders receiving recommendation per day	Refers to the stakeholders of the project addressed by the implemented solution.	Highest is best.	2	Everyday

2.2 Murcia/Molina

2.2.1 Description

The municipalities of Murcia and Molina de Segura have similar goals in terms of mobility due to their proximity and common role for the citizens. As part of GreenMov, by pooling efforts and data sources, they are developing a use case based on AQI calculations, traffic environmental impact and shared bikes availability to assess and predict the impact of road traffic on urban air quality.

By providing a set of green services Murcia and Molina de Segura cities want to promote public transport and rental bike stations to mitigate the impacts of traffic on air quality. To this aim, they provide citizens with the necessary information in the hope of convincing them to abandon the use of cars and search for green alternatives. We also report to the public administration the knowledge to provide intelligent mobility, a critical point for Molina de Segura and Murcia, two relevant cities in the south of Spain that share a lifestyle – companies, universities, shopping centres, etc.

So, in this context, GreenMov takes an essential role in the city necessities to obtain green services for mobility. To delimit a bit, the pilot focuses on Espinardo, a shared university area that also hosts a lot of companies and commerce. Thus, the Campus of Espinardo is a hot spot in terms of mobility and a place with a combination of different public means of transportation (buses, shared bikes, tram, …).

2.2.2 KPIs

Table 2: Main identified KPI's for Murcia/Molina.

Indicator name	Description	Quantification range	Target	Frequency
Number of Green Decisions taken by local authorities.	Refers to the number of decisions taken by the city based on the recommendations provided by the service implemented.	Highest is best.	1	During the entire project
Number of Availability Recommendations per Day.	Refers to the number of recommendations of the bike's availability sent to the stakeholders every day.	Highest is best	4	Every 6 hours
Number of Forecasted Values per Day	Number of Forecasted Values per Day.	Number of Forecasted Values per Day	Number of Forecasted Values per Day	Number of Forecasted Values per Day

2.3 Flanders

2.3.1 Description

By providing forecasts of the availability of shared bikes, Flanders wants to improve the combined use of shared bikes and public transportation. To support this use case, a new OSLO data model “Passenger Transport Hubs” was developed based on the “OSLO Hoppin Points” model (see D2.1 [5] and D2.3 [13]). This use case focuses on Passenger Transport Hubs with trains and “Blue-Bikes” (shared bikes).

The Blue-bike service publishes already real-time availability on its website and via API. In GreenMov the OSLO semantics were applied and the Blue-Bike data were made available as linked data in the LDES format.

To enable citizens to make a data-driven decision to use a shared bike in combination with their train trip a web-app was developed. This app combines the Bike Forecasting services, the LDES-NGSI-LD bike availability service and the real time train schedule service.

Users have been asked about their appreciation of the forecasting service.

If the availability forecast works well for Blue-Bikes, it could also work for other bike share services from other providers. And the Blue-Bike data published in LDES format could be made available to other route-planners and applications in a cost-efficient manner for data owner and app-developer.

2.3.2 KPIs

Table 3: Main identified KPI’s for Flanders.

Indicator name	Description	Quantification range	Target	Frequency
End users quantified satisfaction	Refers to the experience of the end users requesting to know the number of bikes available in the short-term future. The experience is quantified by assessing the accuracy of bikes availability forecasts in the stations targeted within the project.	[0-100%] 100% is best	75%	Every request

3 Architectures

3.1 General Architecture

Activity 4 has determined a generic architecture that use cases can implement as a basis for their implementations. Figure 1shows such an architecture that adopts a multi-layer structure. First, at the bottom, we find the sensors that provide the raw data. In parallel of the sensors, we can find as well third-party services, cloud services that provide relevant data for the use cases. Then, in the second layer, we find Aggregation platforms, such as proprietary platforms that directly gather data from the sensors, maybe using their own data models.

The third layer consists of the core solutions that allow data routing and storage. Indeed, the third layer defines the context broker, that ensures consistency of the data received with existing datamodels, and that then stores the data in relevant databases. These databases can either be real-time databases, such as MongoDB based databases, or historical database, to store timeseries values, such as 20mplement 20 databases. API tools for such historical databases exist and include QuantumLeap (mostly for NGSI-v2), Cygnus, Draco, or Mintaka. The real-time brokers proposed in the generic architecture are OrionLD and Scorpio. The deployment can use container technology based on docker, or Kubernetes. These technologies are defined in Section 6.

Next, the application layer includes all the data analysis aspects, mostly based on historical data. These application layer includes dashboards, data processing, etc...

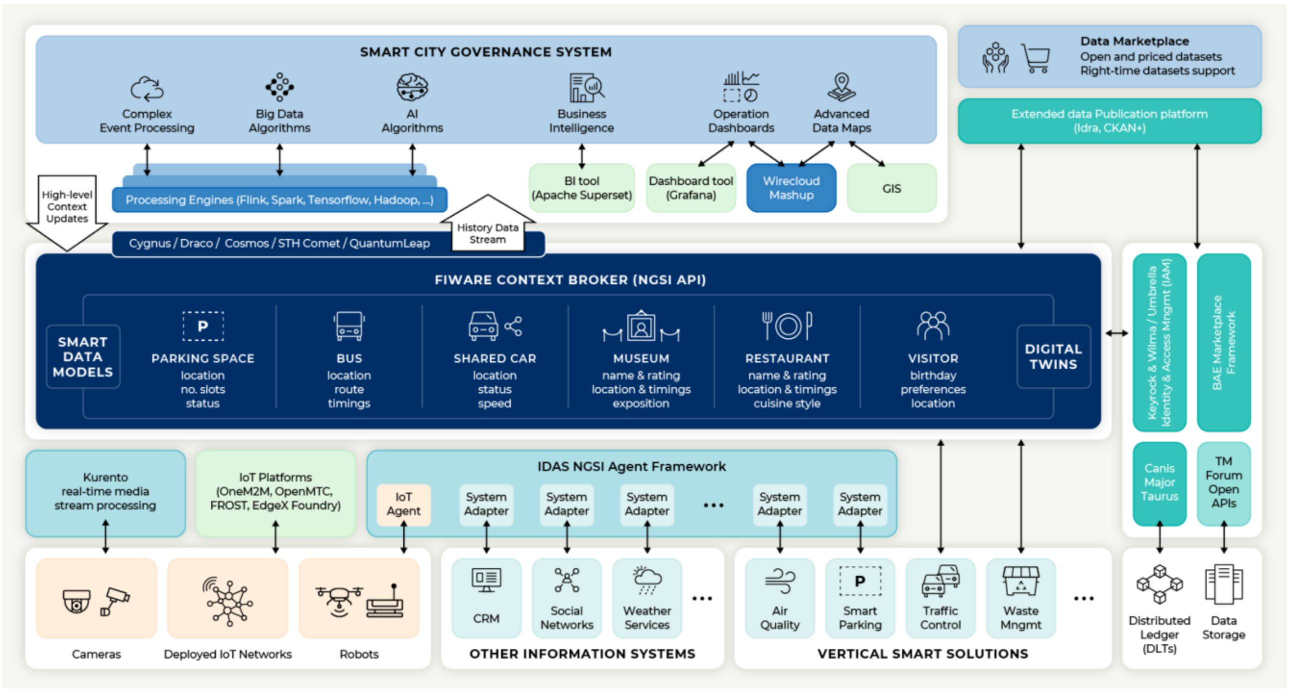

SMART CITY GOVERNANCE SYSTEM
Complex Event Processing
Big Data Algorithms
AI Algorithms
Business Intelligence
Operation Dashboards
Advanced Data Maps
Data Marketplace
Open and priced datasets
Right-time datasets support
Extended data Publication platform (Idra, CKAN+)
High-level Context Updates
Processing Engines (Flink, Spark, Tensorflow, Hadoop, ...)
BI tool (Apache Superset)
Dashboard tool (Grafana)
Wirecloud Mashup
GIS
History Data Stream
Cygnus / Draco / Cosmos / STH Comet / QuantumLeap
FIWARE CONTEXT BROKER (NGSI API)
SMART DATA MODELS
PARKING SPACE location no. slots status
BUS location route timings
SHARED CAR location status speed
MUSEUM name & rating location & timings exposition
RESTAURANT name & rating location & timings cuisine style
VISITOR birthday preferences location
DIGITAL TWINS
Keyrock & Wilma / Umbrella Identity & Access Mngmt (IAM)
BAE Marketplace Framework
Canis Major Taurus
TM Forum Open APIs
Kurento real-time media stream processing
IoT Platforms (OneM2M, OpenMTC, FROST, EdgeX Foundry)
IDAS NGSI Agent Framework
IoT Agent
System Adapter
Cameras
Deployed IoT Networks
Robots
CRM
Social Networks
Weather Services
OTHER INFORMATION SYSTEMS
Air Quality
Smart Parking
Traffic Control
Waste Mngmt
VERTICAL SMART SOLUTIONS
Distributed Ledger (DLTs)
Data Storage

Figure 1 : Generic Architecture [3].

Finally, in parallel of these layers, we can find sub-services such as securing of access to data, using Keycloack and Keyrock identity and access management. Also, the addition of a LDES adapter can help in the supply of time series data by speeding up the process of data retrieval.

Document name:	D5.3 Pilot deployment and validation					**Page:**	20 of 86
Reference:	D5.3	**Dissemination:**	PU	**Version:**	1.0	Status:	Final

Based on this generic architecture was proposed within Activity 4, the use cases defined in activity 5 have implemented their own specificities. The rest of this section describe these specificities.

3.2 Use Cases Architectures

In this subsection, we present the specific architectures as implemented within all use cases.

3.2.1 Nice

The architecture of Nice follows the architecture proposed in Figure 1, and is detailed in Figure 2.

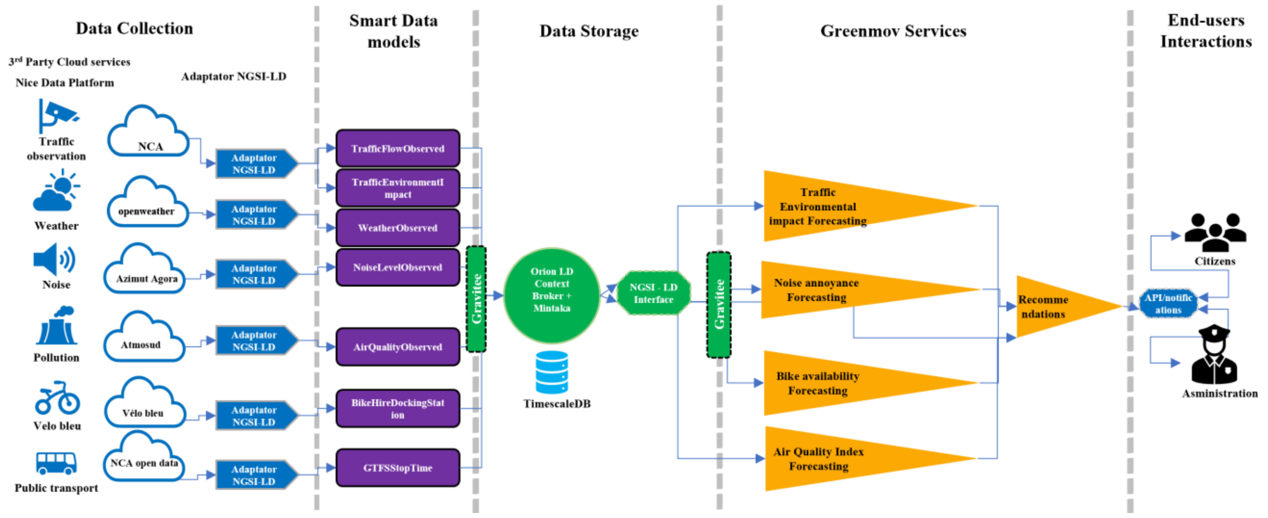


Figure 2 : Nice use case architecture.

The first layer that is detailed in the left-hand side of Figure 2, represents the data collection layer along with the adapters that allow a conversion towards NGSI-LD format. In the case of Nice, several sensors are used, such as air quality monitoring, weather, noise, bikes availability, transport availability, and the traffic monitoring. Once each of this data has been translated into the right data model, it is sent to the storage layer, with the context broker and the storage technology. In the case of Nice, the context broker is OrionLD, that is used along with Mintaka and storage of data achieved through PostgreSQL technology using TimescaleDB, which is an efficient timeseries data storage solution. On top on the APIs available in the generic architecture (21mpleme and OrionLD), the Nice use case has implemented an API manager, named Gravitee, that allows several things:

- Route renaming/routing, which allows clients (such as the services) to use different addresses than the standard context broker addresses to access the data. In the case of Nice, data can be accessed using: https://tip-imredd.unice.fr/data/imredd/nice/bikestation/entities?api-key=32a5519c-fde2-4df4-bcaf-bf29cbbe1989&type=https://smartdatamodels.org/dataModel.OSLO/ResourceReport&limit=1000 instead of an address beginning by: https://orion-ld:1026/ngsi-ld/v1/.
- 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 second 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.

3.2.1.1 Software

The Nice use case requires several software components that are described in the sections below.

3.2.1.2 Data collection

The first software components relate to the collection of data for the use cases. These components were implemented in Node-RED and achieve the following tasks:

- Traffic observation: at the time of this deliverable writing, the software component that collects and store the traffic intensity retrieves an updated csv file at every time interval from the Nice metropole. 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 send 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 send it to the context broker.
- Traffic: a nodered flow retrieves the Nice Côte d'Azur metropole traffic data on a daily basis, convert it 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.

3.2.1.3 Data storage

For data storage, the standard orion-ld docker image was used to replicate the storage architecture. This includes TimescaleDB, and Mintaka. On top of that, other components were added, such as Gravitee as an API manager to provide a dataset catalogue, re-routing (to replace the standard Orion-LD URLs with custom URLs), and a secured access to data through API keys.

NGINX component was also added to trigger different services depending on the route of the URL.

3.2.1.4 Services implementation

For the services implementation, several software components were designed and developed. In all cases, these services were implemented in dockerized Python images. Figure 3 displays a generic architecture for most of the forecasting related services. Services include an API to receive requests for the service's output. This is done through kserve component and FastAPI. Then, most services implement an AI model that requires training which depends on the library used, but mostly scikit learn "fit" function. The model is then used to compute a forecast or a set of forecasts that are used by a "formula computation" component that aims to compute a processed information from the forecasts. As an example, air quality index forecasting as aims to forecast particles concentrations in the near future, that are then used to compute the future air quality index.

The services run on a docker image, and AI models training are executed on a regular basis (daily). Whereas another sub-service called MLFlow runs in parallel to allow the service operator to monitor the ML components as shown in the Figure 4 for the example of the AirQuality Forecasting, Finally, all services include http requests components to request last data from the storage facility's context broker, which are then used to generate the short term forecasting.

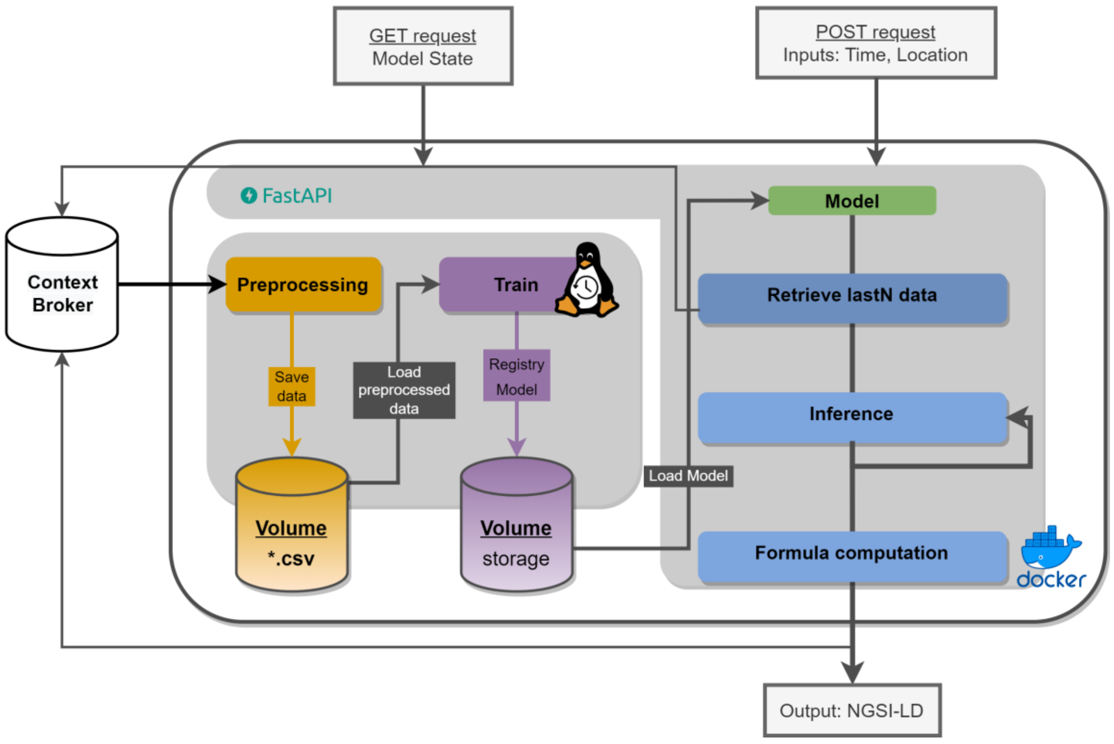
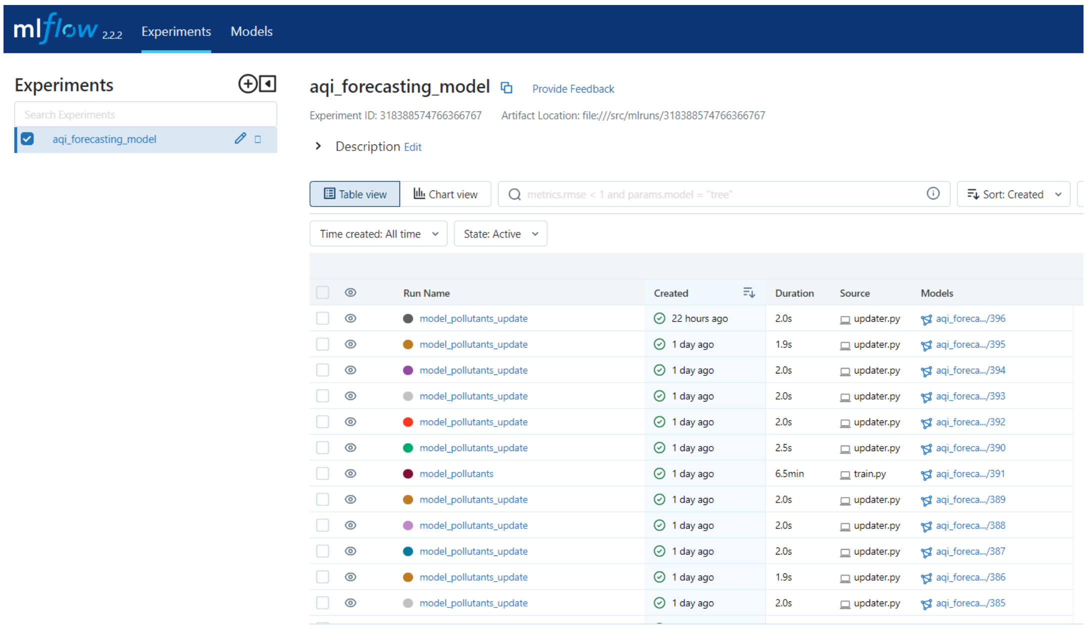


Figure 3 : Services Generic architecture [5].



aqi_forecasting_model

Experiment ID: 318388574766366767 Artifact Location: file:///src/mlruns/318388574766366767

Run Name	Created	Duration	Source	Models
model_pollutants_update	22 hours ago	2.0s	updater.py	aqi_foreca.../396
model_pollutants_update	1 day ago	1.9s	updater.py	aqi_foreca.../395
model_pollutants_update	1 day ago	2.0s	updater.py	aqi_foreca.../394
model_pollutants_update	1 day ago	2.0s	updater.py	aqi_foreca.../393
model_pollutants_update	1 day ago	2.0s	updater.py	aqi_foreca.../392
model_pollutants_update	1 day ago	2.5s	updater.py	aqi_foreca.../390
model_pollutants	1 day ago	6.5min	train.py	aqi_foreca.../391
model_pollutants_update	1 day ago	2.0s	updater.py	aqi_foreca.../389
model_pollutants_update	1 day ago	2.0s	updater.py	aqi_foreca.../388
model_pollutants_update	1 day ago	2.0s	updater.py	aqi_foreca.../387
model_pollutants_update	1 day ago	1.9s	updater.py	aqi_foreca.../386
model_pollutants_update	1 day ago	2.0s	updater.py	aqi_foreca.../385

Figure 4 : ML Flow monitoring interface for AQI forecasting.

3.2.1.5 Front end implementation

In the updated scenario, front-end services are available to end-users in Nice. This means that 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 provided through specific requests. In the Nice use case, the front-end component consists of both the interface for end-users and the API component. The interface for end-users is the WebApp that is displayed in Figure 5. 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 advice to use bikes and public transportation.

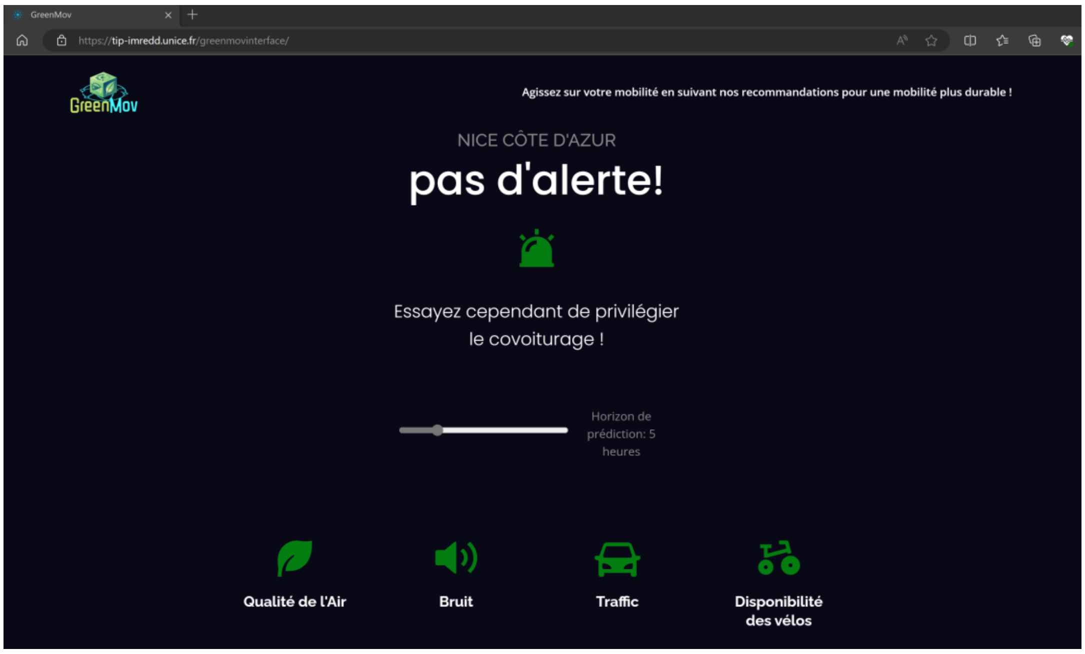

GreenMov
https://tip-imredd.unice.fr/greenmovinterface/
GreenMov
Agissez sur votre mobilité en suivant nos recommandations pour une mobilité plus durable !
NICE CÔTE D'AZUR
pas d'alerte!
Essayez cependant de privilégier le covoiturage !
Horizon de prédiction: 5 heures
Qualité de l'Air
Bruit
Traffic
Disponibilité des vélos

Figure 5 : Nice Use case front end

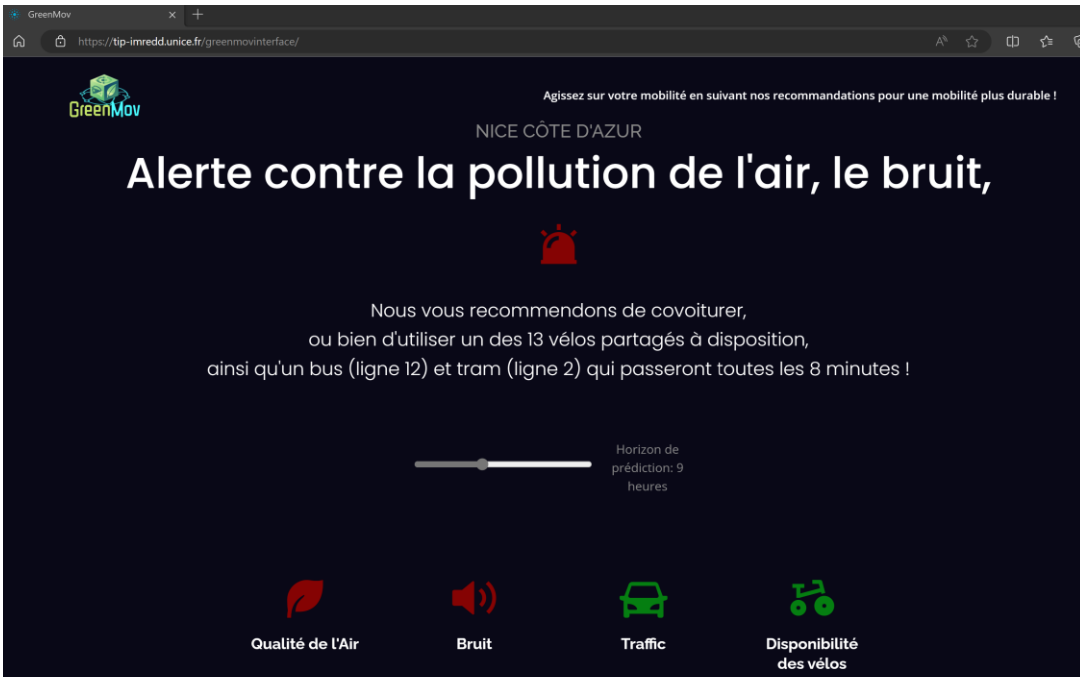

GreenMov
https://tip-imredd.unice.fr/greenmovinterface/
GreenMov
Agissez sur votre mobilité en suivant nos recommandations pour une mobilité plus durable !
NICE CÔTE D'AZUR
Alerte contre la pollution de l'air, le bruit,
Nous vous recommendons de covoiturer, ou bien d'utiliser un des 13 vélos partagés à disposition, ainsi qu'un bus (ligne 12) et tram (ligne 2) qui passeront toutes les 8 minutes !
Horizon de prédiction: 9 heures
Qualité de l'Air
Bruit
Traffic
Disponibilité des vélos

Figure 6 : Nice use case front end with an alert due to air quality and noise annoyance

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 7 shows these different visualisations proposed to the end-users, along with a map on which users can find where the bikes are available.

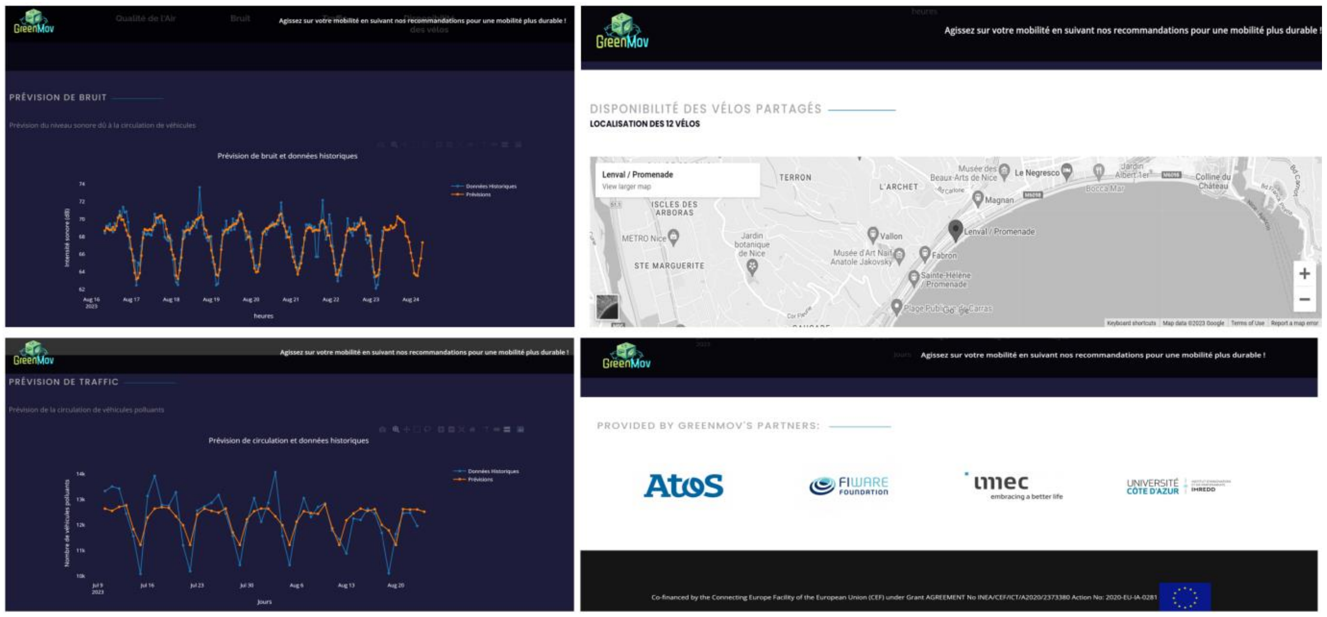

PRÉVISION DE BRUIT
Prévision de bruit et données historiques
PRÉVISION DE TRAFFIC
Prévision de circulation et données historiques
DISPONIBILITÉ DES VÉLOS PARTAGÉS
LOCALISATION DES 12 VÉLOS
Agissez sur votre mobilité en suivant nos recommandations pour une mobilité plus durable !
PROVIDED BY GREENMOV'S PARTNERS:
Atos
FIWARE FOUNDATION
imec
UNIVERSITÉ CÔTE D'AZUR

Figure 7 : Rest of Front End for Nice use case

3.2.1.6 Infrastructure

For the Nice use case, all the software components and storage facility were implemented in IMREDD's owned servers located in IMREDD's own data centre. This is depicted in Figure 8 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.

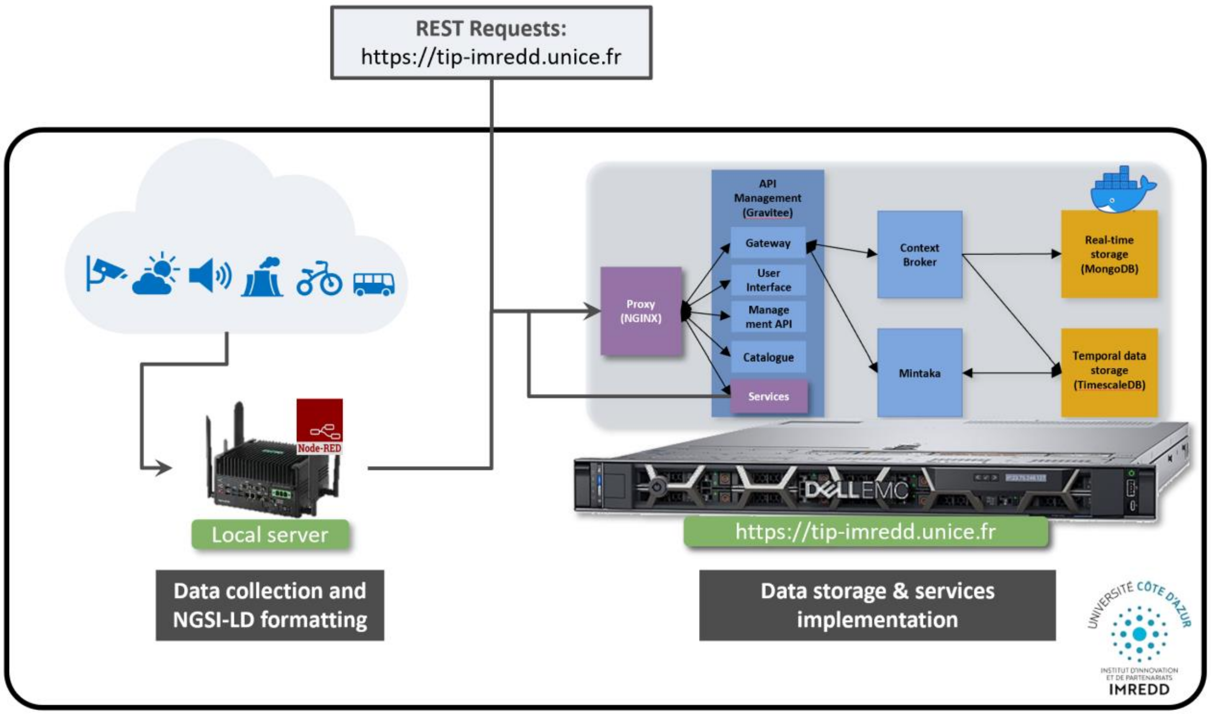

REST Requests:
https://tip-imredd.unice.fr
Node-RED
Local server
Data collection and NGSI-LD formatting
Proxy (NGINX)
API Management (Gravitee)
Gateway
User Interface
Management API
Catalogue
Services
Context Broker
Mintaka
Real-time storage (MongoDB)
Temporal data storage (TimescaleDB)
DELL EMC
https://tip-imredd.unice.fr
Data storage & services implementation
UNIVERSITÉ CÔTE D'AZUR
INSTITUT D'INNOVATION ET DE PARTENARIATS
IMREDD

Figure 8 : Overall infrastructure for Nice Use Case.

3.2.2 Murcia/Molina

The architecture of Murcia/Molina follows the architecture proposed in Nice, with the particularity that it uses context brokers federation to centralise data from both cities, as is detailed in Figure 9 below:

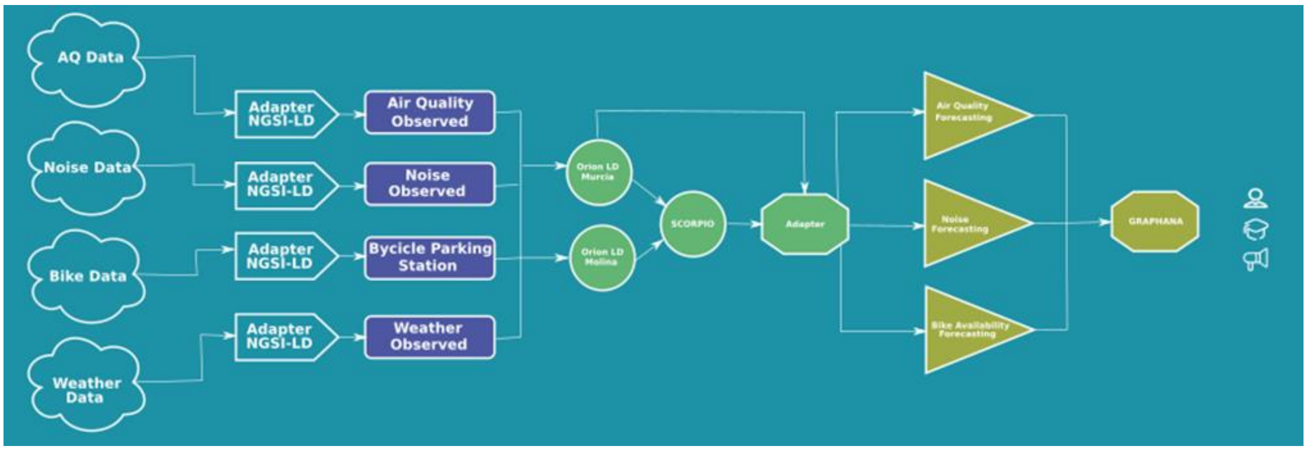


Figure 9 : Architecture Murcia / Molina Use Case.

The first layer is called the data collection layer, including the adapters that convert the data into NGSI-LD format.
In the case of Murcia/Molina, several sensors are used, such as air quality monitoring, noise and bikes stations. Once each of this data has been translated into the right instance of the corresponding data model, it is sent to the corresponding context broker and the storage technology. Then, Orion-LD brokers from Murcia and Molina are federated into a Scorpio broker hosted by HOPU.

In the case of Murcia, the bike data is only consumed by the cities brokers and is not federated into the Scorpio broker for technical reasons. The reason for this is that according to OSLO data model, a new entity is created at each timestamp. This consumes a lot of memory and lead to a Scorpio fail when the memory of the server is full. For this reason, we opted only for federated noise and air quality data, which are more robust in this aspect. This federated structure feeds the green services, which can be accessed by the different actors and stakeholders via dashboard and graphics in Grafana.

3.2.2.1 Software

The software integrated in the architecture of this use case includes all the green services that is deployed, namely Air Quality Index Calculation, Air Quality Forecasting, Traffic Impact Calculation, Bike Availability Forecasting and Noise Annoyance Forecasting, described in detail in Section 5. In addition, the software includes the NGSI adaptors and a service used to check and correct the missing values from Air Quality Sensors.

3.2.2.2 Data collection

The principal software components are the *services* from the GreenMov suite. The Murcia/Molina use case implements:

- Air Quality: the air quality data is collected from the IoT devices manufactured by HOP Ubiquitous. The software used to collect the data store the information of the sensor in a Context Broker (v2) and then transform it in NGSI-LD in the federation process, using the AirQualityObserved smart data model.

- Weather: this data is obtained from the AEMET reference stations, that can be accessed via API. This data is persisted in a CrateDB component.
- Noise: the noise data is collected from the IoT devices manufactured by HOP Ubiquitous. The software used to collect the data store the information of the sensor in a Context Broker (v2) and then transform it in NGSI-LD in the federation process, using the NoiseLevelObserved smart data model.
- Bike Availability: the bike availability data is collected from the MuyBicistations and stored in a Context Broker (v2) and then transform it in NGSI-LD in the federation process, using the NoiseLevelObserved smart data model.

3.2.2.3 Data storage

For data storage, a docker image was used to replicate the architecture. This includes CrateDB, and QuantumLeap.

3.2.2.4 Services implementation

The software integrated in the architecture of this use case include all the green services that are deployed, namely Air Quality Index Calculation, Air Quality Forecasting, Bike Availability Forecasting and Noise Annoyance Forecasting, described in detail in Section 5. In addition, the software includes the NGSI adaptors, and a service used to check and correct the missing values from Air Quality Sensors.

For the services implementation, several software components were designed and developed. In all cases, these services were implemented in dockerized Python images, using Flask, FastAPI and MLFlow. Then, most services implement an AI model that requires training which depends on the library used, but mostly scikit learn "fit" function.

The services run on a docker image, and AI models training are executed on a regular basis (daily). Finally, all services include http requests components to request last data from the storage facility's context broker, which are then used to generate the short-term forecasting.

3.2.2.5 Front end implementation

For Front end implementation, we used the open tool Grafana. Grafana is a powerful front-end tool for data visualization and monitoring. With its intuitive and user-friendly interface, Grafana allows users to create dynamic dashboards, charts, and graphs to represent complex data in a visually appealing manner. Its extensive library of plugins and integrations with various data sources enables seamless data retrieval and analysis. Grafana empowers developers and data analysts to effortlessly explore and present data-driven insights, as well as provide friendly interfaces for end users.

The frontend interface to be implemented has different functionalities. At this moment, it is 28mplementted over the devices installed in Molina de Segura and is extended with Murcia bike stations and the green services. The first dashboard shows the location of the different sensors over the use case perimeter.

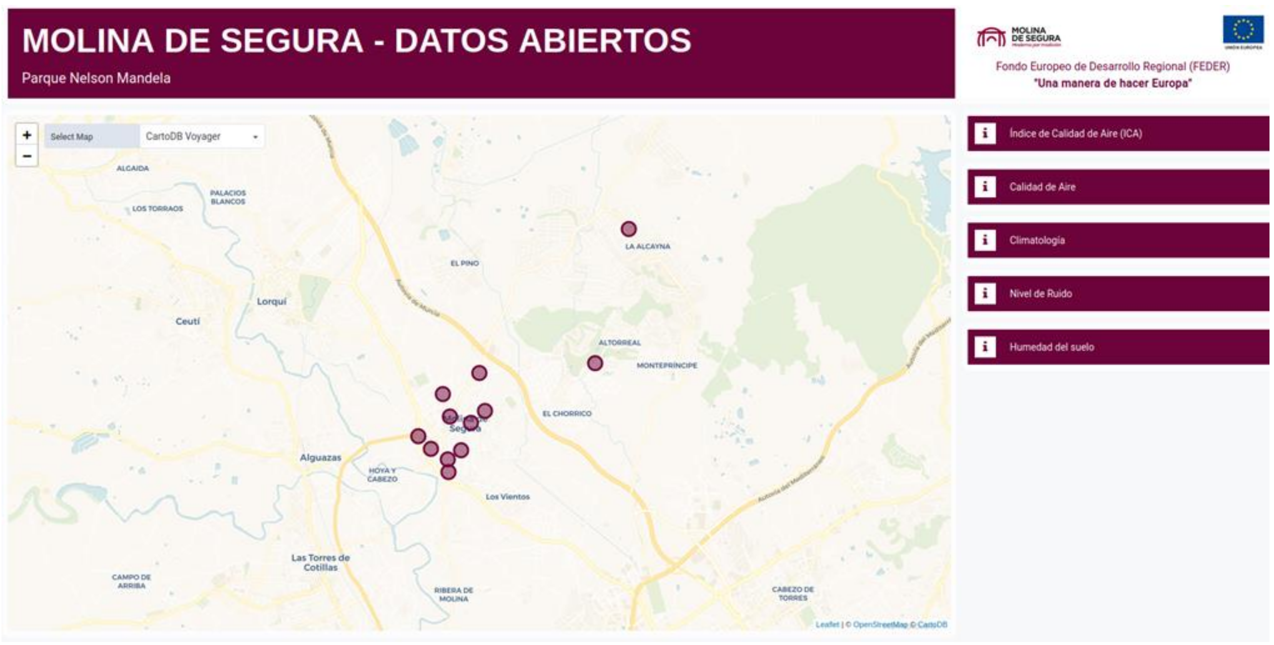
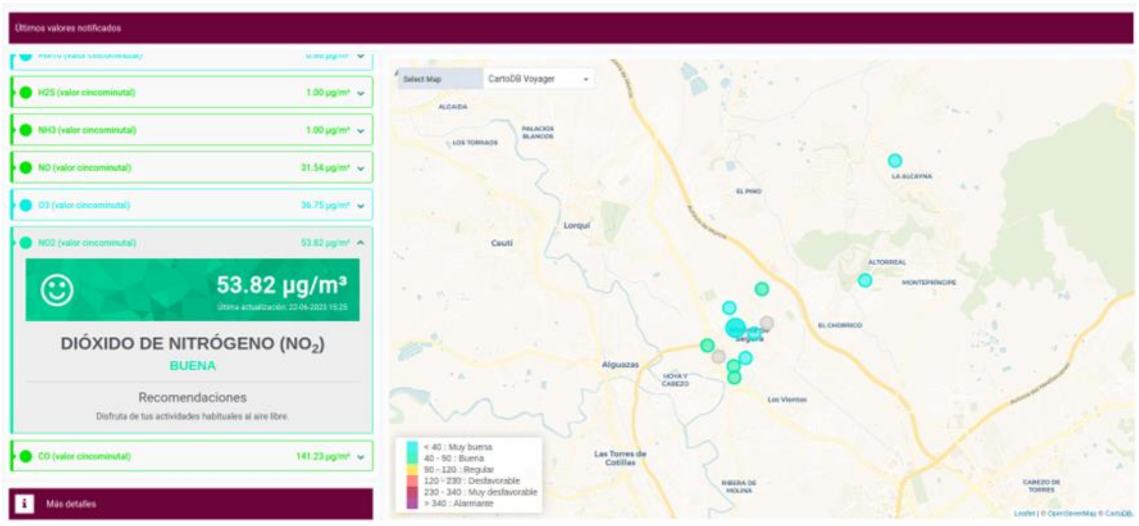


Figure 10 : Murcia/Molina dashboard with sensors locations.

Next, it is possible to inspect each sensor and the different pollutants. The color of the sensor is selected in function of the AQI calculated by the corresponding service.


Últimos valores notificados

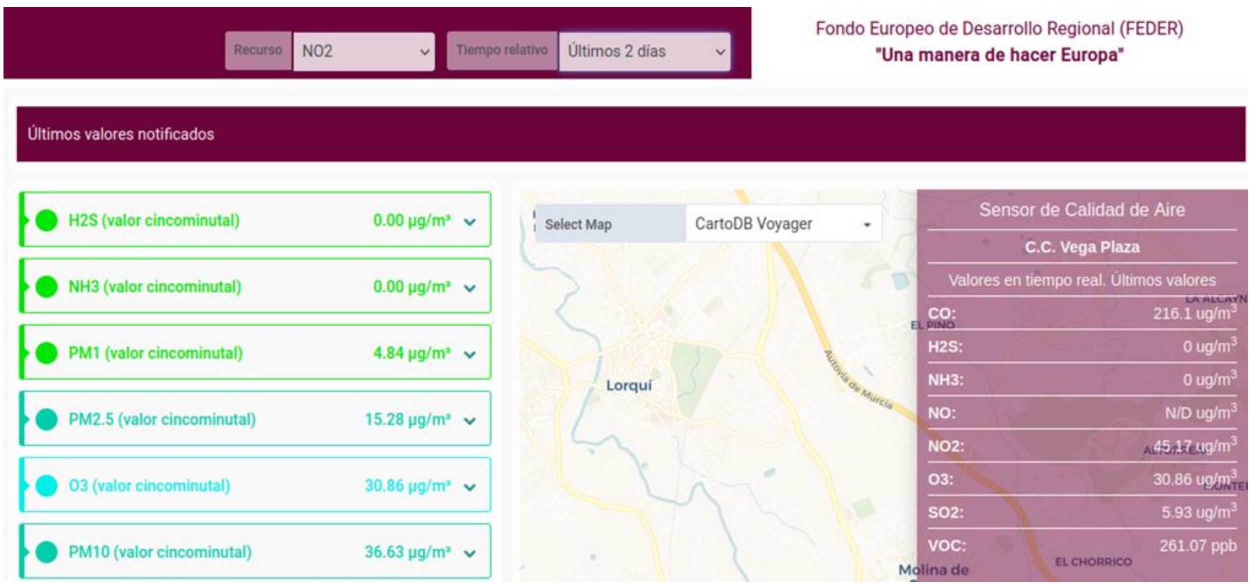
H2S (valor cincominutal)	1.00 µg/m³
NH3 (valor cincominutal)	1.00 µg/m³
NO (valor cincominutal)	31.54 µg/m³
O3 (valor cincominutal)	36.75 µg/m³
NO2 (valor cincominutal)	53.82 µg/m³
CO (valor cincominutal)	141.23 µg/m³

53.82 µg/m³
Última actualización: 22-06-2023 15:25
DIÓXIDO DE NITRÓGENO (NO_2)
BUENA
Recomendaciones
Disfruta de tus actividades habituales al aire libre.
Más detalles

🟦	< 40 : Muy buena
🟩	40 - 90 : Buena
🟨	90 - 120 : Regular
🟥	120 - 230 : Desfavorable
🟥	230 - 340 : Muy desfavorable
🟪	> 340 : Alarmante

Figure 11 : Example of a sensor inspection on the Murcia/Molina dashboard.

Last, when the user inspects the data, it is possible to select the time period for which the data is shown. This allows to select past data and future data, as well as the forecasting services allows to perform this.


Recurso NO2 Tiempo relativo Últimos 2 días

Fondo Europeo de Desarrollo Regional (FEDER)
"Una manera de hacer Europa"

Últimos valores notificados

H2S (valor cincominutal)	0.00 µg/m³
NH3 (valor cincominutal)	0.00 µg/m³
PM1 (valor cincominutal)	4.84 µg/m³
PM2.5 (valor cincominutal)	15.28 µg/m³
O3 (valor cincominutal)	30.86 µg/m³
PM10 (valor cincominutal)	36.63 µg/m³

Select Map: CartoDB Voyager

Sensor de Calidad de Aire
C.C. Vega Plaza
Valores en tiempo real. Últimos valores

CO:	216.1 ug/m³
H2S:	0 ug/m³
NH3:	0 ug/m³
NO:	N/D ug/m³
NO2:	45.17 ug/m³
O3:	30.86 ug/m³
SO2:	5.93 ug/m³
VOC:	261.07 ppb

Figure 12 : Overview of the dashboard functionalities.

3.2.2.6 Infrastructure

The infrastructure is based on a platform with a context broker that federates the context brokers of the two cities. This master context broker is hosted by HOPU in an Azure machine, enabling enough computation resources and the scalability of the services.

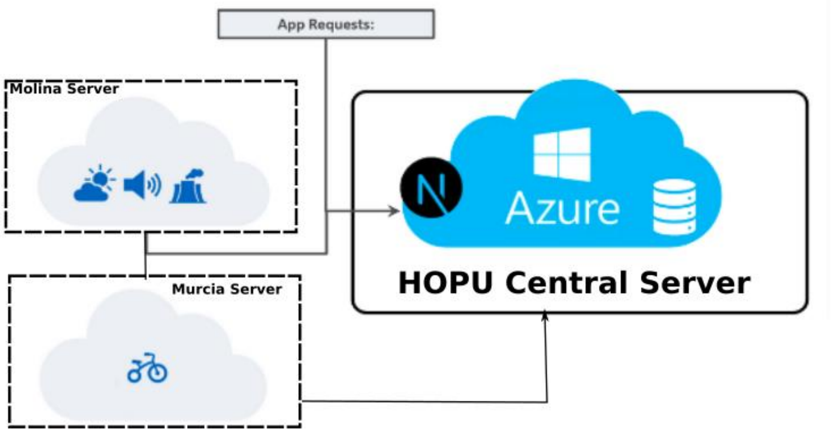


Figure 13 : Infrastructure in Murcia/Molina use case.

3.2.3 Flanders

In the figure below the architecture and the flows for the scenario 'forecast bike availability upon train arrival' are shown.

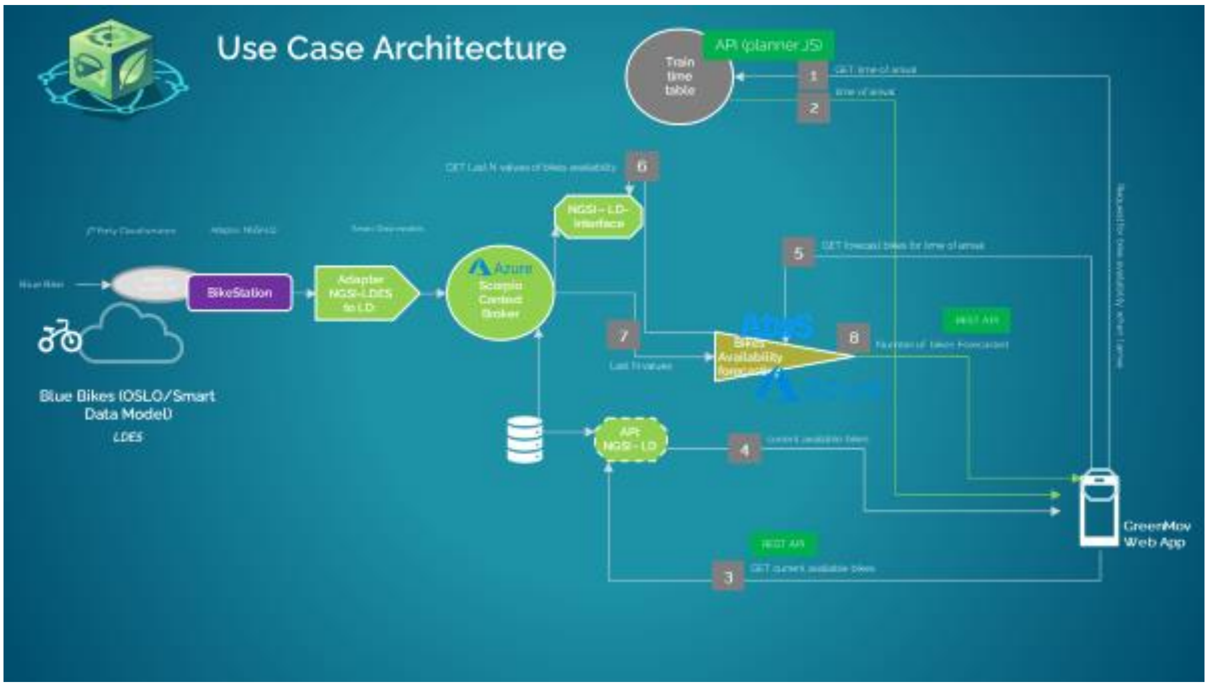


Figure 14 : Architecture of Flanders' Use Case.

3.2.3.1 Software

The software integrated in the Flanders architecture use case include the LDES replicator (LDES to NGSI-LD), Redis, Orion-LD Broker which is deployed in a distributed way. It consists of the following components: atContextServer, Config server, Entity Manager, Eureka server, Gateway, History Manager, Query Manager, Registry Manager, Registry Subscription Manager and Subscription Manager. The Orion-LD context broker as defined in Section 1.4 has a dependency on Kafka and TimescaleDB so these software components are also used. The other software components include the NGSI-LDES component and the bike forecast model developed as a service in Activity 3.

The GreenMov app is a web-based tool that helps users plan their train and bike journeys by checking the availability of Blue-bikes at their destination station. The app uses three main APIs – Planner.js for train availability, Atos/Scorpio for the Blue-bike current data, and the data-model api for the bike forecast – to fetch train and bike availability data and provide users with real-time predicted information on their options.

The app is built with NextJS, ReactJS, Mapbox, DaisyUI library, and it is designed to be user-friendly and intuitive. The NextJS framework provides server-side rendering and static generation capabilities, allowing the app to load fast. The ReactJS library provides a component-based approach for building user interfaces, allowing the app to be modular and scalable. The Mapbox platform provides mapping and location data, allowing the app to display train and bike stations on a map and provide directions.

3.2.3.2 Infrastructure

The LDES Server is hosted by IMEC IDLab on their own infrastructure and it contains the Blue-bike data open to public. It can be used by several applications.

All the other components such as the Orion context broker, the LDES-replicator etc are hosted by IMEC on an Azure Kubernetes Service (AKS) cluster in Azure cloud. This provides better container orchestration, enough computation and resource allocation, and scalability of the services. On the same AKS cluster, the Blue-bike web frontend, Bike forecast service AI model and MongoDB are hosted as well.

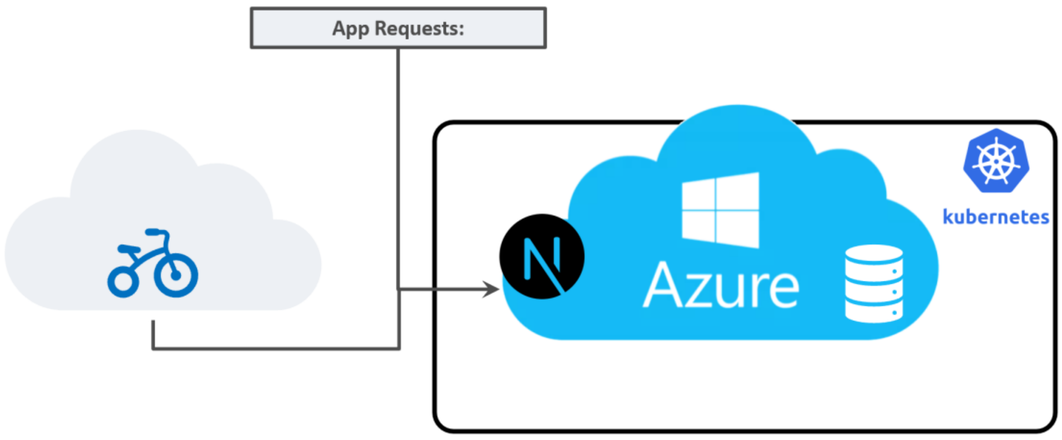


Figure 15 : Infrastructure in Flanders use case [1].

4 Data Models

This section describes the updated list of the identified data models. Some of the data models from "D2.1 Extended Smart Data Models v1" [5] and "D2.2 Extended Smart Data Models" [6] were either changed, extended or removed from the project due to the evolutions of the services deployed in the different use cases.

The list of data models used without changes:

Table 4: List of data models used without changes.

Data Model	Description	Associated service(s)
AirQualityMonitoring	Used for the AQI calculations.	Air Quality Index Calculation.
WeatherObserved	An observation of weather conditions at a certain place and time used for the AQI calculations.	Air Quality Index Calculation.
WeatherForecast	A forecast of weather conditions for a certain place and time used to predict the AQI.	Air Quality Index Forecasting.
GTFSStopTime	Used in the traffic recommendation service to retrieve public transportations plan.	Traffic recommendation, sub-service: Alternative transport availability forecasting.

Some data models were updated during the project with new attributes (See D2.2 [6]). The list of the extended data models is:

Table 5: List of updated data models.

Data Model	Description	Associated service(s)
AirQualityObserved	An observation of AQ conditions used for the AQI calculations.	Air Quality Index Calculation.
BikeHireDockingStation	A description of bike hiring station at a certain place and time, used for generating alternative transportations recommendations.	Bikes' availability forecasting, Traffic recommendations generation, sub-service: Alternative transport availability forecasting.

Data Model	Description	Associated service(s)
NoiseLevelObserved	An observation of noise levels at a certain place and time, used to calculate the noise pollution.	Noise annoyance calculation.
TrafficFlowObserved	An observation of traffic flow conditions used to calculate traffic impact.	Traffic environmental impact calculation, Traffic recommendations generation.

New data models within the Smart Data Models framework were introduced during the analysis of the use cases. The list of the new data models is:

Table 6: List of new data models.

Data Model	Description	Associated service(s)
AirQualityForecast	A forecast of AQ conditions used for the AQI forecast service.	Air Quality Forecasting.
NoisePollution	An observation of noise levels at a certain place and time, used to calculate the noise pollution.	Noise annoyance calculation.
NoisePollutionForecast	A forecast of noise pollution at certain place and time, used for noise annoyance forecasting service.	Noise Annoyance Forecasting.
BicycleParkingStation	An FIWARE/OSLO data model describing the actual state of a Bicycle Parking Station, used to forecast the future state of the station.	Bikes availability forecasting, sub-service: Alternative transport availability forecasting.
BicycleParkingStationForecast	An FIWARE/OSLO data model forecasting the state of a Bicycle station at a certain place and time.	Bikes' availability forecasting, Traffic recommendations generation, sub-service: Alternative transport availability forecasting.
TrafficEnvironmentImpact	Aims to calculate the environmental impact of the current traffic flow observed at a given location.	Traffic environmental impact calculations, Traffic forecasting

Data Model	Description	Associated service(s)
TrafficEnvironmentImpactForecast	Aims to calculate what is the environmental impact of the current traffic observed at a given time and location.	Traffic environmental impact calculations, Traffic forecasting
ResourceReport	Resource Report Schema meeting Passenger Transport Hubs AP Schema specifications.	Bikes availability forecasting
ResourceReportForecast	A forecast of the resource Report Schema for Passenger Transport Hubs.	Bikes availability forecasting

- Some of the data models mentioned in previous deliverables (D5.2 [[18], D5.1 [4], D2.1 [5] and D2.2 [6]] were finally not used due to changes in some services architecture and structure. The list of the data models not used are:
    - ParkingSpot, which was not used because only traffic intensity, bikes and public transportation information are used in the use cases. Parking Spots could have been used to recommend to people to park their car and use public transportation, however, at use case's locations, there was no exhaustive monitoring of the available parking spot, which highlights one of the main limitations of development of interoperability solutions in real implementation: there is still and always missing information or IoT deployment.
    - VehicleEmissionsLabel: this smart data model was replaced by the TrafficEnvironmentImpact data model, which is more exhaustive and directly includes VehicleEmissionsLabel.
    - PublicTransportation: Likewise, this smart data model was dropped out because the GTFSStopTime data models already includes all the information needed.
    - GBFS station_status and GBFS station_information: these smart data model waere left out because their information was no longer needed as they were all included in the BicycleParkingStation smart data model.

5 Services

This section describes the services and sub-services adopted in the project along with the implementation and deployment specificity of each service depending on its use case:

5.1 Air Quality Index

As it is explained in D3.1 [7] and D3.2 [1], basic schema is applied, that is, forecasting service followed by calculation service to provide easy-readable information to users. A certain prediction of a given set of pollutants is output by AQF service. Next, AQC service processes this prediction to get the corresponding air quality index. The whole process is related to a time-location. Both AQF and AQC services works in an hourly based manner.

5.1.1 Air Quality Index Calculation

As it is defined in D3.2 [1], this service calculates Air Quality Index and Air Quality Level, based on the level of pollutants, as dictated by European normative. It receives the predictions from the Air Quality Forecasting service and outputs an AirQualityForecast instance with the corresponding Air Quality Index and Air Quality Level.

The architecture of the service is detailed in the Figure 16.

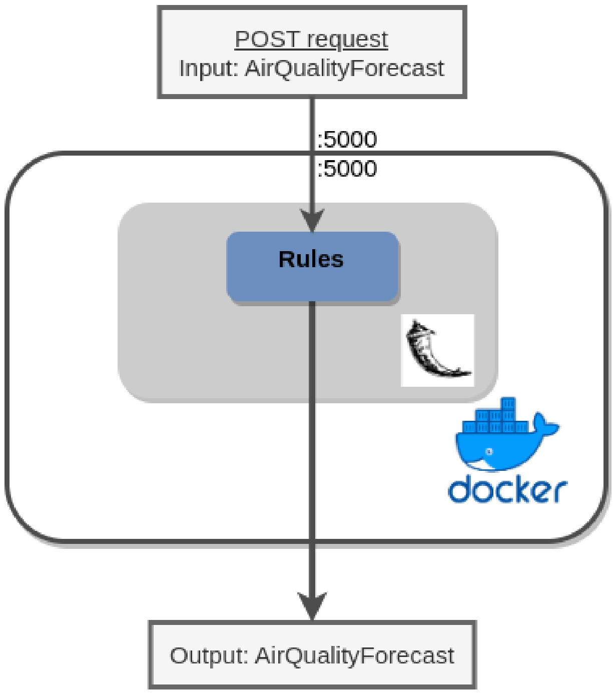


Figure 16 : Air Quality Index Calculation service architecture [1].

5.1.2 Air Quality Index Forecasting

As it is defined in D3.2 [1], this service provides an hourly prediction of a given set of pollutants. Having received a request containing a time stamp and a geographical location for the forecast, it outputs the corresponding pollutant levels.

The architecture of the service is detailed in the Figure 17.

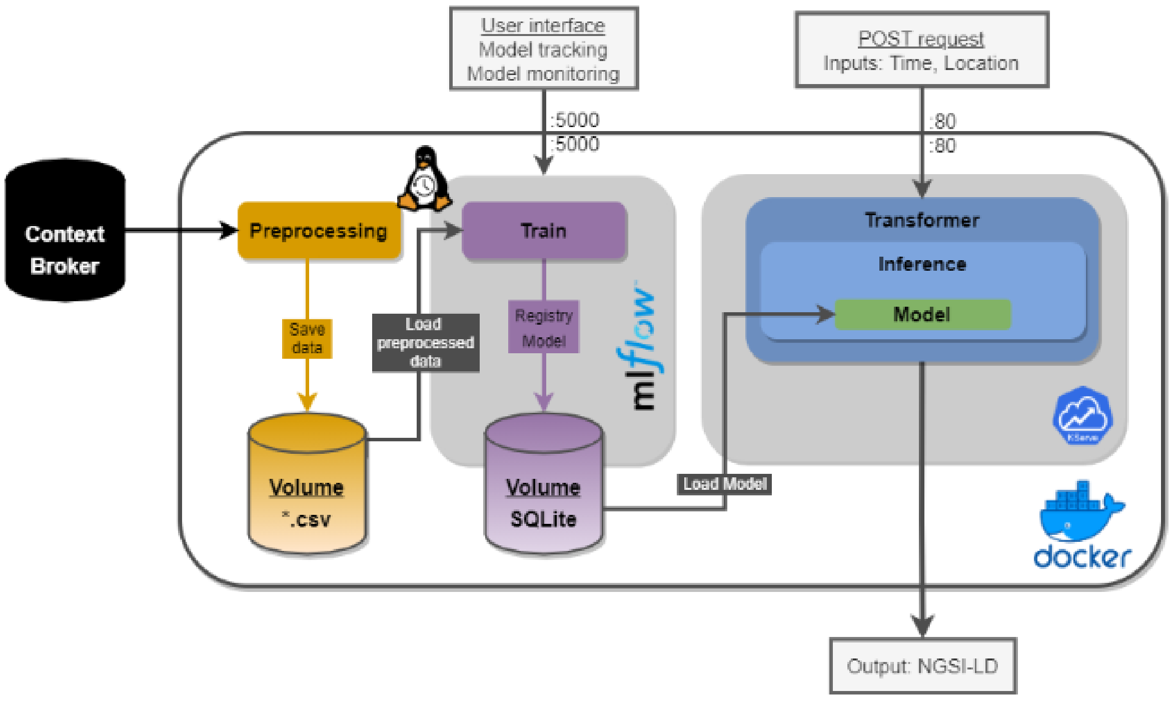


Figure 17 : Architecture AQF service 0.

5.1.3 Use Case specificity

5.1.3.1 Nice use case specificity

The Air Quality Index (AQI) forecasting service for the city of Nice use a combination of IoT infrastructures and machine learning techniques to provide accurate and real-time forecasts of AQI levels within the city. The AQ sensor belonging to the south of France air quality observatory ATMOSUD is located inside the use case area as seen in the Figure below.

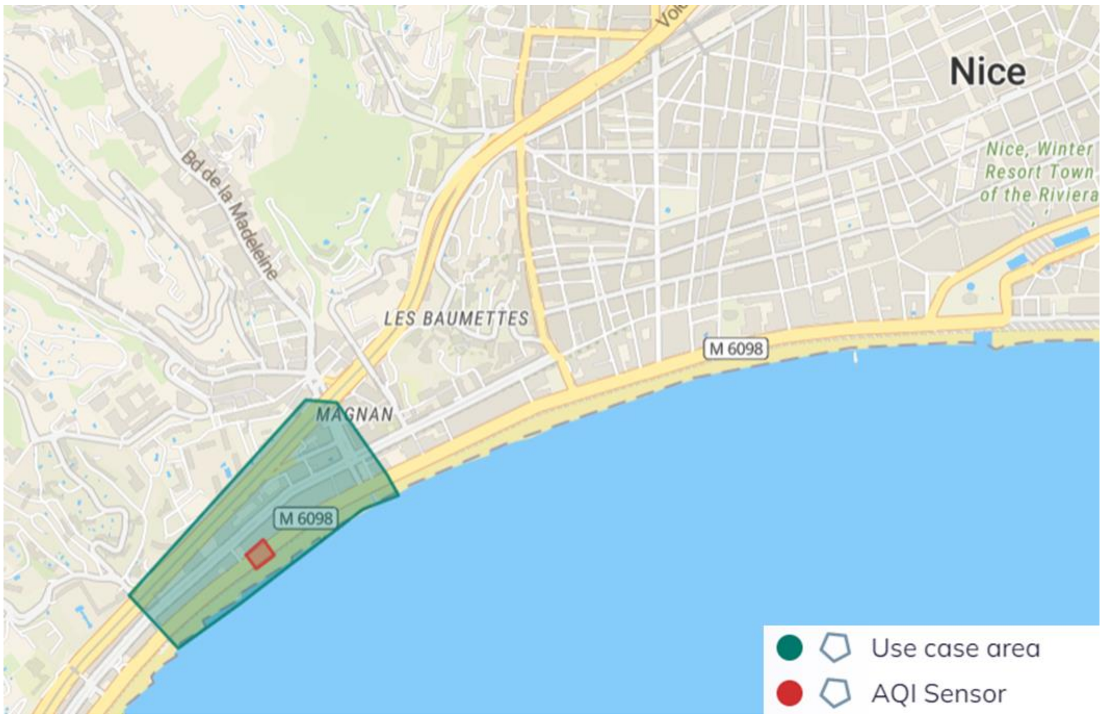
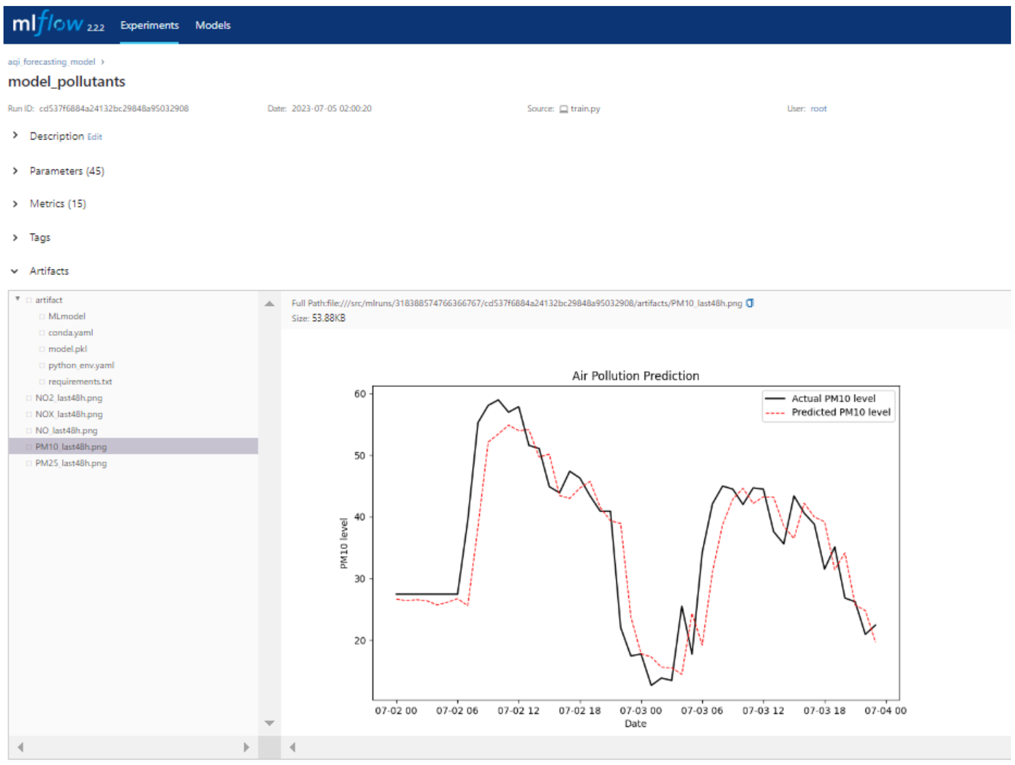


Figure 18 : Nice use case AQI sensor.

The service makes use of 3rd party cloud services and databases to access and process historical AQI data for the past 2 years.

The implementation process starts by collecting the historical AQI levels as well as weather parameters. The data is then pre-processed and analysed to identify patterns and trends. Various forecasting models are tested and evaluated using benchmarking techniques to select the most accurate and efficient one. The service is then encapsulated into a Docker image, which enables easy deployment and scalability. Additionally, the service allows for customization by integrating it with the Nice city's infrastructure.

To ensure effective monitoring of the service, a sub-service called MLFlow is developed to run in parallel. MLFlow serves as a monitoring and tracking tool for the service operator. Its primary function is to collect and store data related to the performance and execution of the ML components as shown in the Figure 19.


mlflow 2.2.2 Experiments Models

aqi_forecasting_model ›

model_pollutants

Run ID: cd537f6884a24132bc29848a95032908 Date: 2023-07-05 02:00:20 Source: train.py User: root

› Description Edit

› Parameters (45)

› Metrics (15)

› Tags

⌄ Artifacts

- artifact
  - MLmodel
  - conda.yaml
  - model.pkl
  - python_env.yaml
  - requirements.txt
- NO2_last48h.png
- NOX_last48h.png
- NO_last48h.png
- PM10_last48h.png
- PM25_last48h.png

Full Path:file:///src/mlruns/318388574766366767/cd537f6884a24132bc29848a95032908/artifacts/PM10_last48h.png

Size: 53.88KB

Figure 19 : MLFlow AQI index monitoring interface.

5.1.3.2 Murcia/Molina use case specificity

As well as in Nice, the Air Quality Index (AQI) forecasting service uses a combination of IoT infrastructures (composed by Smart Spots in the city of Molina) and machine learning techniques to provide accurate and real-time forecasts of AQI levels within the city, and it is also applied to the air quality station in San Basilio. The Figure below shows the location of the Air Quality Devices in Molina de Segura.

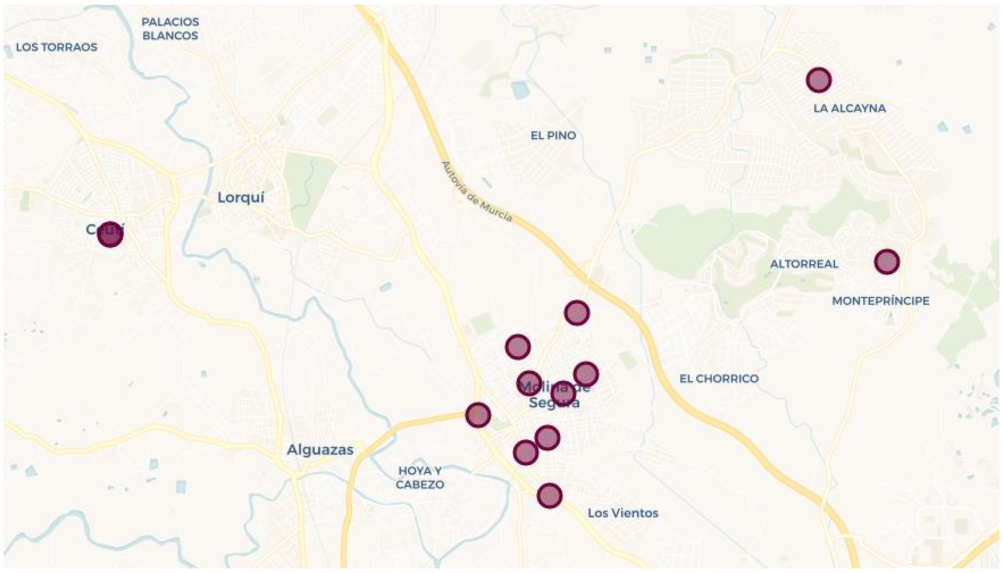


Figure 20 : Molina Air Quality Sensors.

5.1.3.3 Flanders use case Specificity

Air quality services are not used in the Flanders use case.

5.2 Traffic Environmental Impact services

The traffic environmental impact service aims to provide a forecast of short-term future impact of the traffic on the environment.

This service can be decomposed into several components, as depicted in Figure 21. First, on the left side, the data is collected by the use cases collection software solutions and is then formatted into a NGSI-LD data model to provide the traffic intensity per type of vehicles' emissions. This data is then stored into the use case database. The traffic environmental impact service is then composed of a Machine Learning component, that requires frequent training from recently collected data, and that provides forecasts for short term future traffic intensity, and finally a calculation component that computes the actual particles emissions from the categorised traffic intensity forecast.

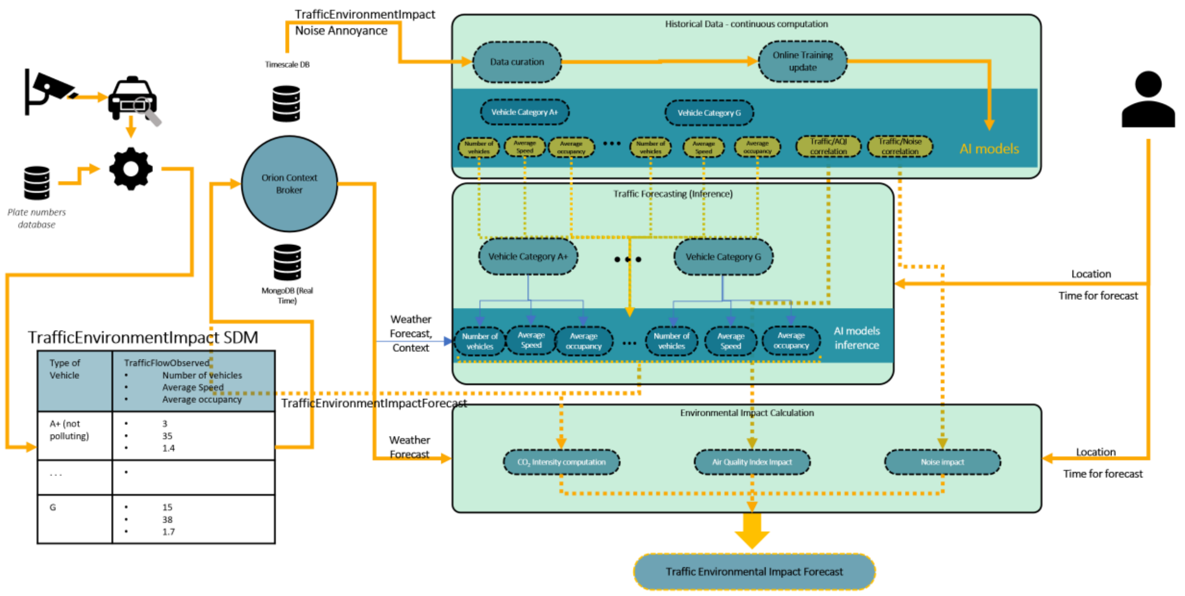


Figure 21 : Traffic environmental impact forecasting service [1].

5.2.1 Traffic forecasting

The service provides a prediction of the traffic flow for a given location and at a specific time T based on historical traffic data. Setting T equal to 0 means the current traffic flow. This service includes an API sub-service to provide forecasts on requests. According to Figure 22 from D3.2 [1], The service runs in a docker container that can be started using docker-compose in any computer or server, as it was done for Nice use case.

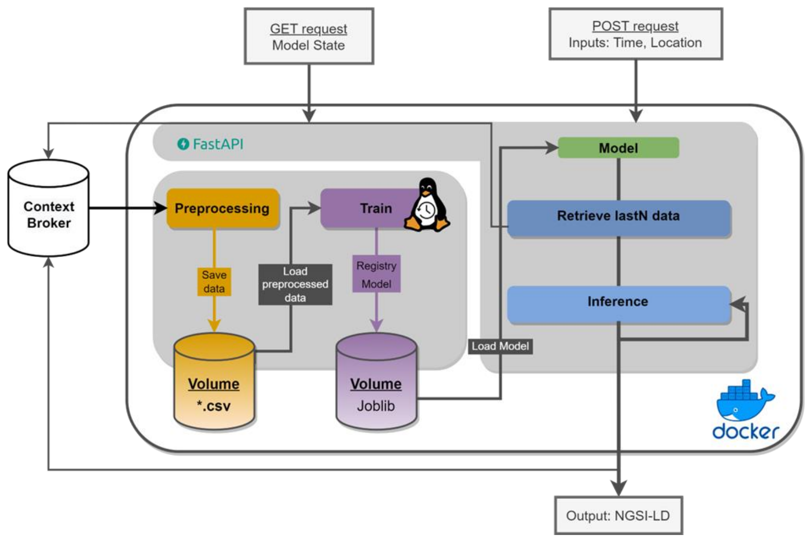


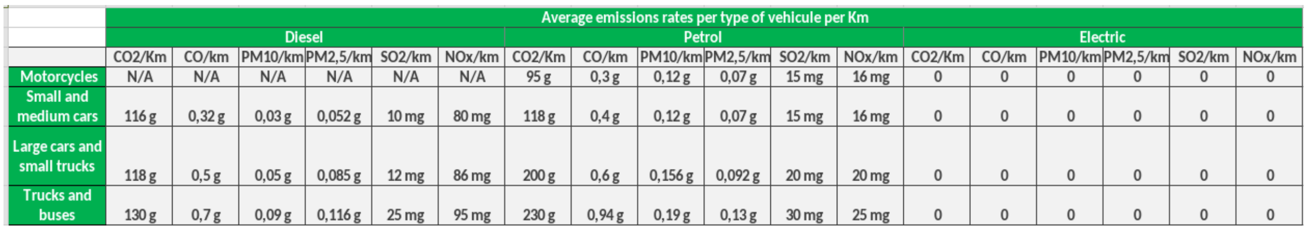
Figure 22 : Architecture of Traffic forecasting service [1].

5.2.2 Traffic Environmental Impact Calculation

The forecast provided by the traffic forecasting service can be used as an input to the traffic environmental impact calculation service that determines the CO2 and particles emissions due to the traffic intensity forecast categorized by vehicles' pollution levels.

The vehicles pollutions level is categorized and specified in the table below for each type:

Table 7: Average emissions rate per type of vehicle.



	Average emissions rates per type of vehicule per Km																	
	Diesel						Petrol						Electric					
	CO2/Km	CO/km	PM10/km	PM2,5/km	SO2/km	NOx/km	CO2/Km	CO/km	PM10/km	PM2,5/km	SO2/km	NOx/km	CO2/Km	CO/km	PM10/km	PM2,5/km	SO2/km	NOx/km
Motorcycles	N/A	N/A	N/A	N/A	N/A	N/A	95 g	0,3 g	0,12 g	0,07 g	15 mg	16 mg	0	0	0	0	0	0
Small and medium cars	116 g	0,32 g	0,03 g	0,052 g	10 mg	80 mg	118 g	0,4 g	0,12 g	0,07 g	15 mg	16 mg	0	0	0	0	0	0
Large cars and small trucks	118 g	0,5 g	0,05 g	0,085 g	12 mg	86 mg	200 g	0,6 g	0,156 g	0,092 g	20 mg	20 mg	0	0	0	0	0	0
Trucks and buses	130 g	0,7 g	0,09 g	0,116 g	25 mg	95 mg	230 g	0,94 g	0,19 g	0,13 g	30 mg	25 mg	0	0	0	0	0	0

This service is implemented through a docker image as seen in Figure 23:

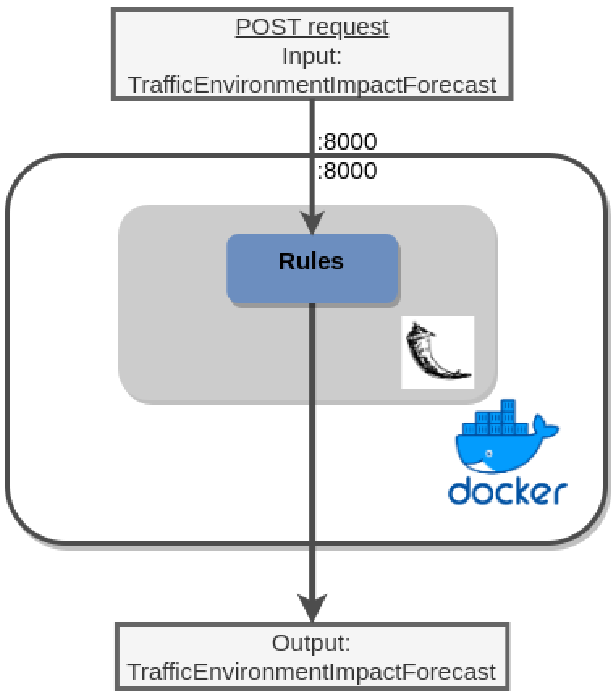

POST request
Input:
TrafficEnvironmentImpactForecast
:8000
:8000
Rules
docker
Output:
TrafficEnvironmentImpactForecast

Figure 23 : Traffic Environmental Impact Calculation service architecture [1].

5.2.3 Use Case specifics

5.2.3.1 Nice Use case Specificity

In Nice, traffic congestion and air pollution are ongoing issues. This service addresses this issue in a specific area situated on the “Promenade des Anglais” and near the “Hopital Lenval”. This zone is located few hundred meters west of the city centre and close to the “Voie Pierre Mathis” which is one of the main traffic axes of the city as shown in the Figure below. Moreover, many tram, bus and bike stations “Velo Bleu” are situated in the same area which makes it the perfect spot to implement this service.

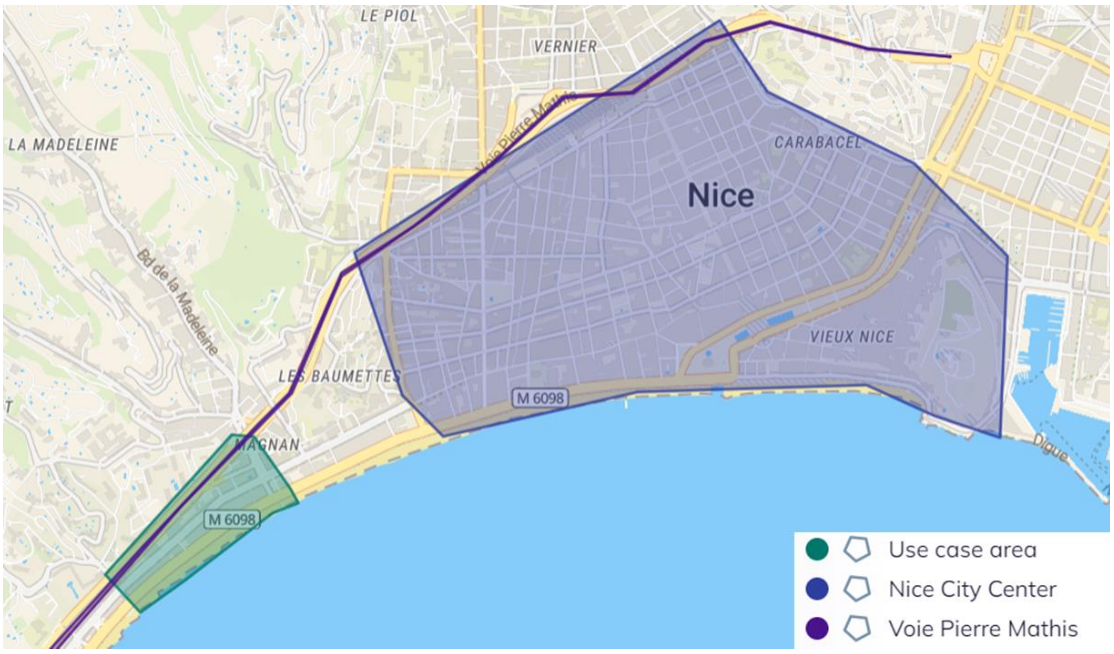


Figure 24 : Nice use case location.

Figure 25 presents an example of forecast for traffic intensity of polluting vehicles in Nice. We see that the forecast accuracy is here above 76% on a specific week of July, which is one of the hardest times to forecast traffic given the affluence of tourists in the city.

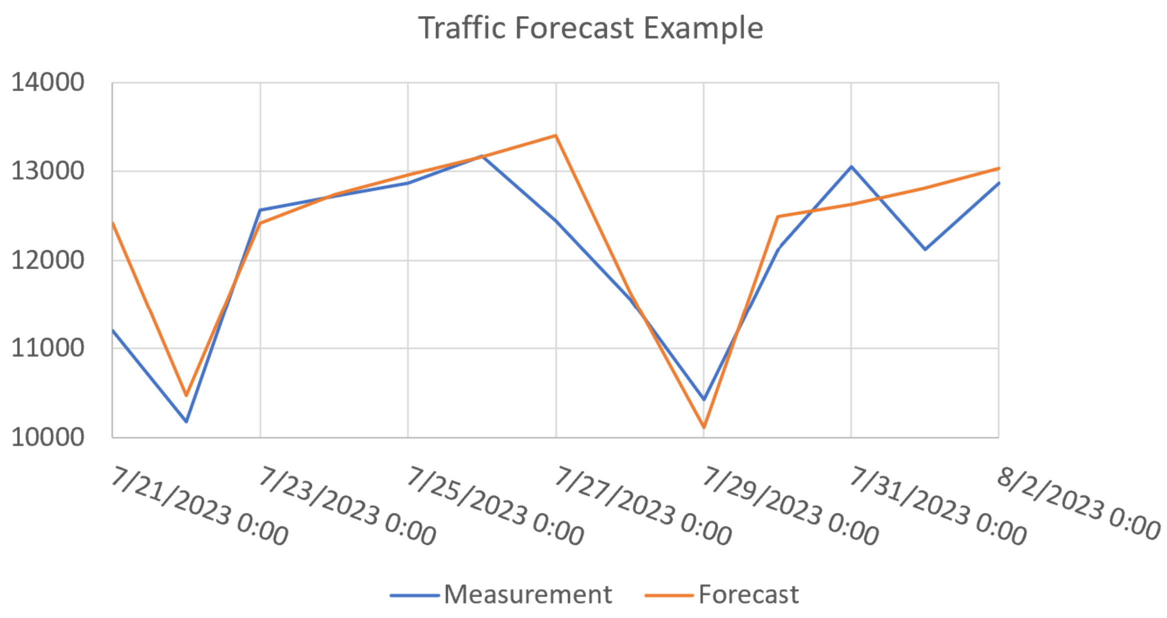


Figure 25 : Forecast example for traffic forecasting service.

5.2.3.2 Murcia/Molina use case Specificity

The Murcia/Molina use case does not use any of the traffic services.

5.2.3.3 Flanders use case Specificity

The traffic environmental impact service is not involved in the Flanders use case.

5.3 Traffic Recommendations Generation

The service provides traffic recommendations at a specific location and given time T. this recommendation generation system uses data from services such as: the traffic environmental impact calculation, the traffic forecasting, the noise annoyance forecasting, the bikes availability and the public transportations alternatives of the involved use case and generates recommendations on the most efficient and eco-friendly mode of transportation to take, these recommendation are published on the front-end interface and sent each hour to the project stakeholders to help make more informed decisions about how to travel.

5.3.1 Use Case specificities

5.3.1.1 Nice Use Case

The implementation of the service begins with processing the outputs of other services, such as noise pollution, air quality, traffic forecast and impact, as well as public transportation options and the availability of bikes nearby, to provide recommendations to users based on these parameters. The initial step is to analyse the public transportation data and other forecasted datasets that are utilized, as well as pre-processing the data. The next step is to conduct benchmarking by testing different algorithms and models to determine the optimal results. This process should consider not only the complexity of the models, but also the amount of data used as input. Lastly, the service is encapsulated into a Docker image through the “dockerization” process.

The flexibility of the Docker image allows for two options for deploying the service. Firstly, the image can be integrated into the Nice city’s infrastructure, which allows for customization to the specificity of our use case. Alternatively, the image can be deployed on a generic cloud for increased scalability.

The diagram in the Figure 26 below provides a detailed representation of how traffic recommendations are generated using data from noise, air quality, weather, and traffic forecasting services. These factors are analysed to determine the best transportation options for individuals, considering whether they should utilize public transport, bikes, car-sharing, or their personal car. By evaluating the environmental conditions, real-time traffic situations, and individual preferences, the system generates personalized suggestions to promote efficient and sustainable mobility choices.

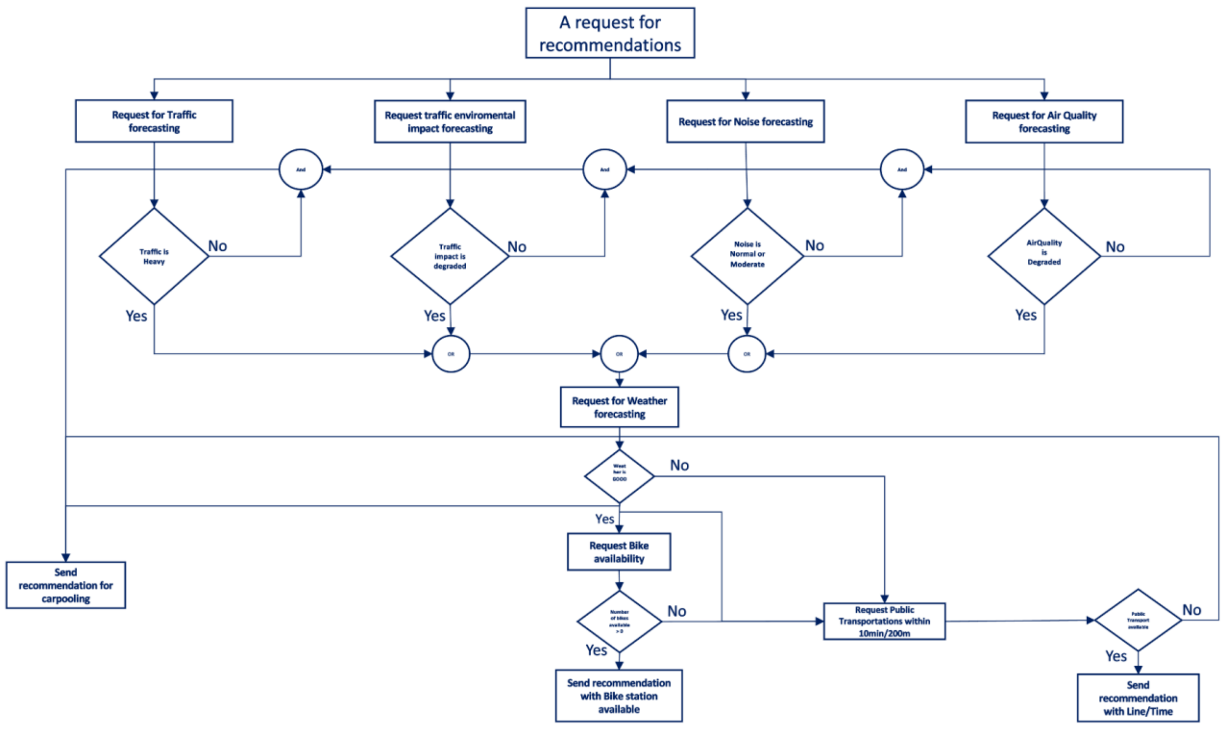
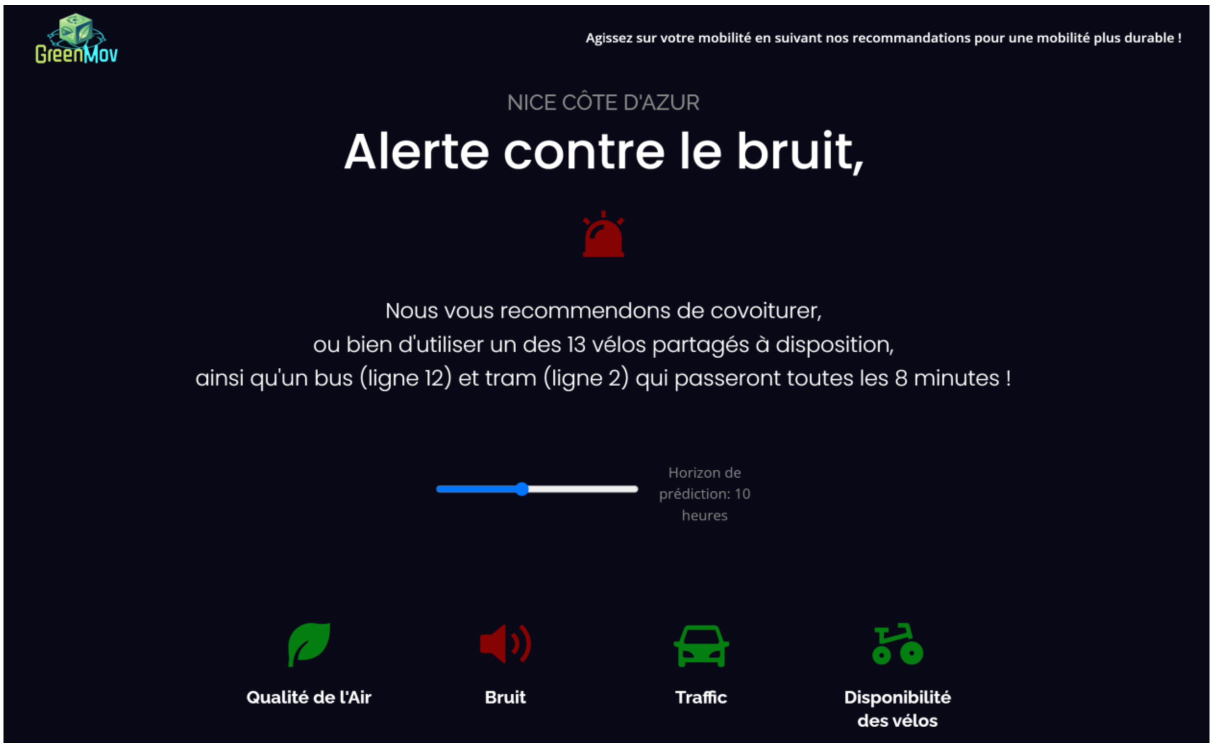

A request for recommendations
Request for Traffic forecasting
Request traffic enviromental impact forecasting
Request for Noise forecasting
Request for Air Quality forecasting
And
And
And
Traffic is Heavy
Traffic impact is degraded
Noise is Normal or Moderate
AirQuality is Degraded
No
No
No
No
Yes
Yes
Yes
Yes
OR
OR
OR
Request for Weather forecasting
Weather is GOOD
No
Yes
Request Bike availability
Number of bikes available > 0
No
Yes
Send recommendation for carpooling
Send recommendation with Bike station available
Request Public Transportations within 10min/200m
Public Transport available
No
Yes
Send recommendation with Line/Time

Figure 26 : Traffic recommendations service flowchart.

This service has been implemented along with an interface which allows an ergonomic interaction with the end-users. Figure 27 represents such an interface in which a recommendation is proposed to end-users, so they change their behaviour because of noise annoyance at the location considered in the use case.



Agissez sur votre mobilité en suivant nos recommandations pour une mobilité plus durable !

NICE CÔTE D'AZUR

Alerte contre le bruit,

Nous vous recommendons de covoiturer,
ou bien d'utiliser un des 13 vélos partagés à disposition,
ainsi qu'un bus (ligne 12) et tram (ligne 2) qui passeront toutes les 8 minutes !

Figure 27. Front End for the traffic recommendation service

5.4 Bikes availability Forecasting Service

The several use cases require a knowledge of the availability of bikes to inform end-users that they can use a bike to reach their destination and reduce the environmental impact of their mobility. Therefore, a bike availability forecasting service was developed by the GreenMov project team to achieve such recommendation. The output of the service consists in the number of bikes available at the requested time, and at the requested location. This requires having a forecasting model for bikes availability at each location considered. This is all explained in detail in D3.2 [1]. As a summary, the architecture of the bikes availability forecasting service is detailed in Figure 28.

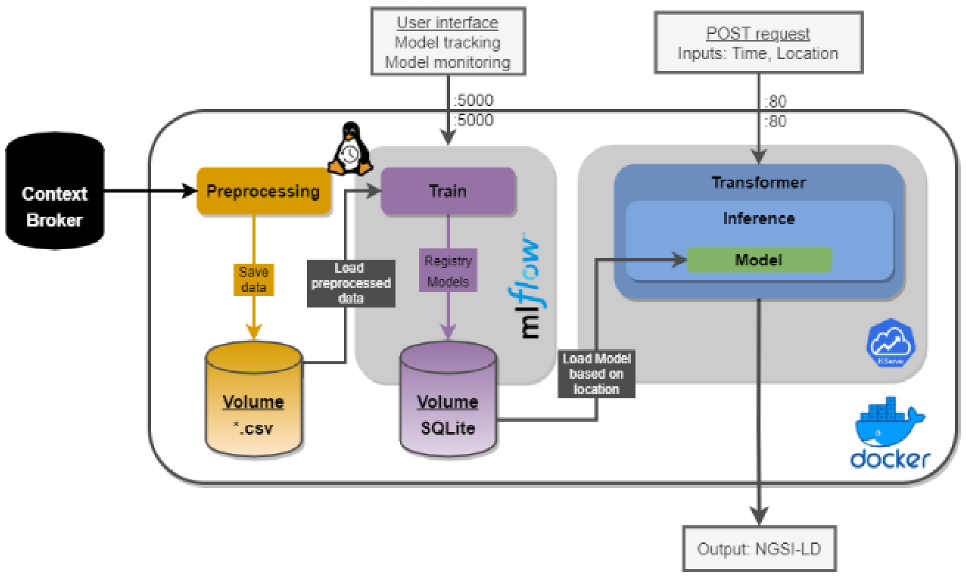


Figure 28 : Architecture Bikes availability forecasting service [1].

Requests from the end-users or from other intermediary services (such as traffic recommendation service) are sent to the bike availability service through an HTTP request, which is handled by the service's API, as shown in Figure 28 in the upper right POST request box. Then, the service directly loads the Machine Learning forecasting model for the requested location. Meanwhile, last availability bikes availability monitored data is retrieved, to make an inference of the bike's availability forecast for the requested time. Details of working principles of this service are provided in D3.2 [1]. As a summary, the forecasting model only provides one forecast for a 30 minutes interval (for example. Can be less depending on the data set time interval). Therefore, to forecast the availability of bikes for the next 3 hours, the model is used 6 times until the time for the request is reached. This is all done automatically, and the end-user directly receives the bikes availability forecast encapsulated in the BikeHireDockingStation smart data model. The API interface with the end-users is done by Kserve. To maintain consistency and ease the deployment, all components have been packaged into a single Docker image. This approach ensures efficient deployment of the bike availability forecasting service.

It is worth noting that these machine learning models for bike availability forecasting are constantly evolving depending on the evolution of bikes availability patterns at each location.

The readers can refer to deliverable D3.2 [1], D3.1 [7] and D4.1 [8] for an exhaustive description.

5.4.1 Use Case specificity

Describes the service implementation/usage in the different use cases.

5.4.1.1 Nice use case specificity

In the Nice use case, 5 "Velo Bleu" stations were selected to forecast bike availability. These stations were chosen based on their proximity to the use case area, as well as their proximity to nearby bus and tram stations. The exact location and name of each station can be seen in the Figure below. The main objective of this service is to provide accurate predictions of bike availability at these five stations, which would feed later the traffic recommendation generation service.

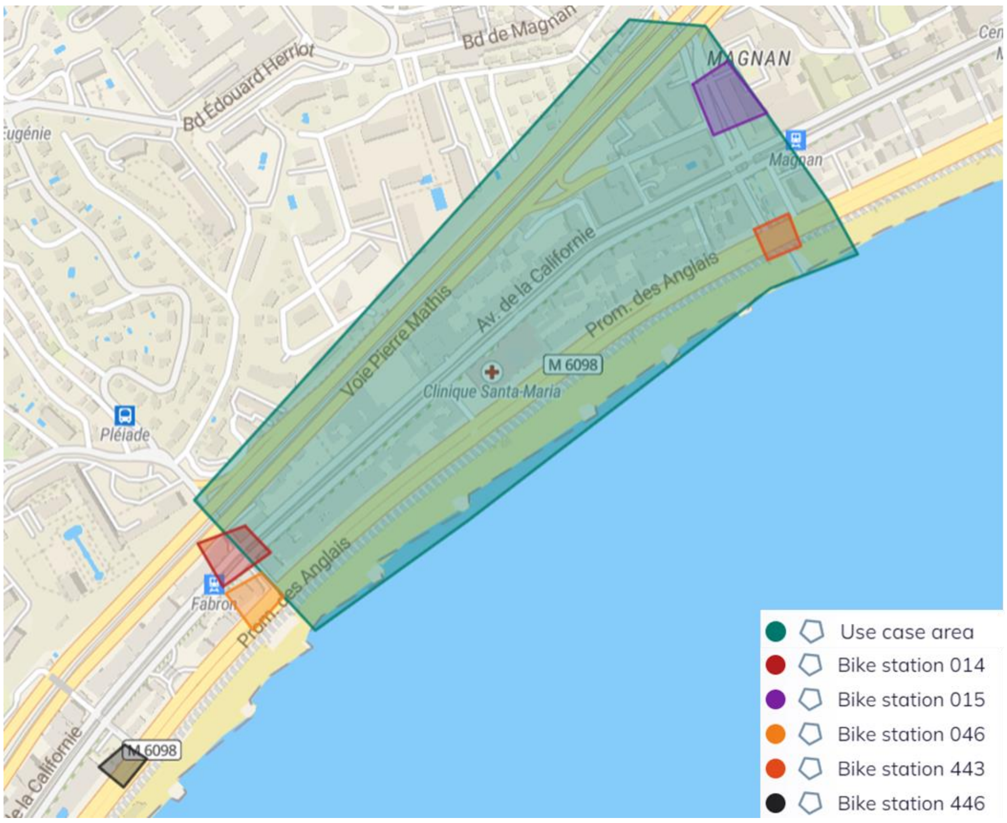
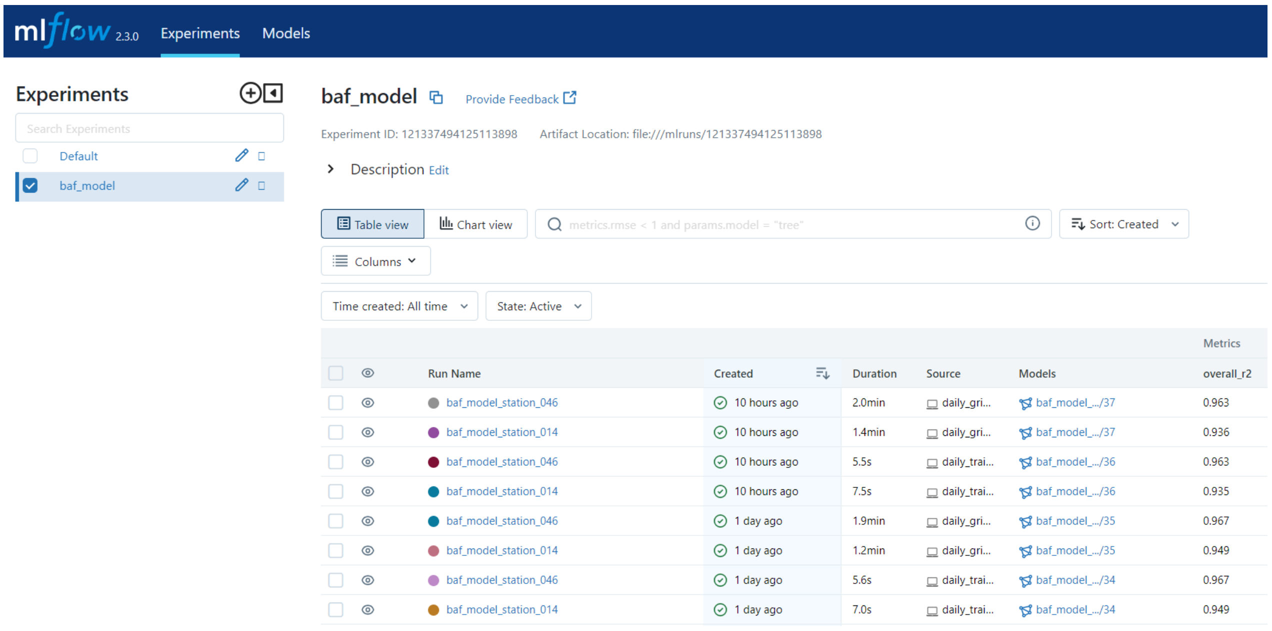


Figure 29 : Nice use case bike stations.

The implementation of the bike forecasting service in the Nice use case begins by utilizing historical data on bike availability over the past few months. The process starts with gathering and analysing this historical data, as well as any additional relevant information. The data is then pre-processed to prepare it for use in the forecasting model. Different algorithms and models are tested and compared through benchmarking to determine the best approach. The complexity of the models and amount of data used as input are also considered in this step. Finally, the service is packaged into a Docker image for deployment. The use of a Docker image allows for easy replication of the service across different environments. This can be particularly useful for testing and development purposes, as well as for creating multiple instances of the service for use by different teams. Furthermore, the image can be manipulated, which allows for rollback to previous versions if necessary. This gives added flexibility and control over the deployment of the services.

Finally, the service can be deployed in two ways, thanks to the flexibility of the Docker image. The first option is to integrate the image into the infrastructure of Nice city, which allows for tailoring the service to the specific needs of the use case. The second option is to deploy the image on a cloud platform, which enables scalability and wider accessibility. Moreover, MLFlow is running in parallel to allow the service operator to monitor the ML components as shown in the Figure 30



baf_model

Experiment ID: 121337494125113898 Artifact Location: file:///mlruns/121337494125113898

Run Name	Created	Duration	Source	Models	overall_r2
baf_model_station_046	10 hours ago	2.0min	daily_gri...	baf_model_.../37	0.963
baf_model_station_014	10 hours ago	1.4min	daily_gri...	baf_model_.../37	0.936
baf_model_station_046	10 hours ago	5.5s	daily_trai...	baf_model_.../36	0.963
baf_model_station_014	10 hours ago	7.5s	daily_trai...	baf_model_.../36	0.935
baf_model_station_046	1 day ago	1.9min	daily_gri...	baf_model_.../35	0.967
baf_model_station_014	1 day ago	1.2min	daily_gri...	baf_model_.../35	0.949
baf_model_station_046	1 day ago	5.6s	daily_trai...	baf_model_.../34	0.967
baf_model_station_014	1 day ago	7.0s	daily_trai...	baf_model_.../34	0.949

Figure 30 : MLFlow monitoring interface for bikes availability service.

The metrics provided within MLFlow indicate an accuracy above 97%, which is mostly due to the fact that bikes availability does not change much in between the 15 minutes interval.

5.4.1.2 Murcia/Molina use case specificity

The Bikes Availability Forecasting is implemented in all the stations in the city of Murcia, but the use case focusses specifically on the stations around the Campus of Espinardo, because of the interaction with the other services. The Figure below shows the location of all the stations that is integrated, highlighting the stations in Espinardo.

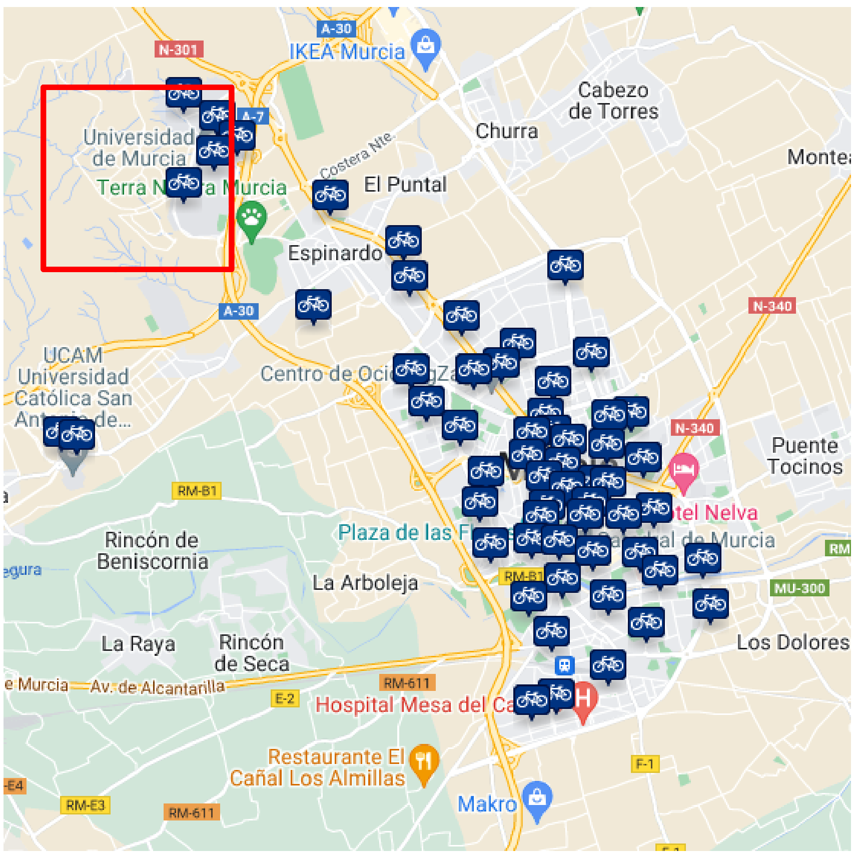


Figure 31 : Murcia Bike Stations.

5.4.1.3 Flanders use case specificity

In the Flanders use case, all 110 "Blue-Bike" stations in Belgium are selected to forecast bike availability. The objective of this service is to provide accurate predictions of bike availability, nearby a train station.

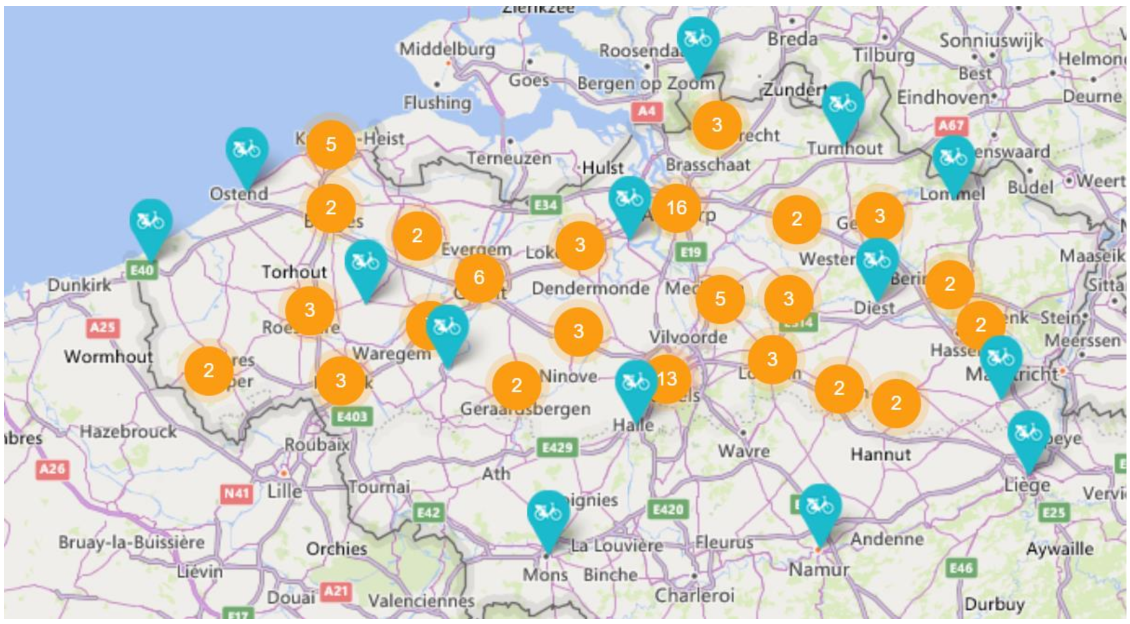


Figure 32 : Blue bike stations in Belgium.

The implementation of the bike forecasting service in Flanders use case began by gathering historical data on bike availability over the past few months, to train the AI algorithm. The model was trained using historical data provided manually in CSV-format. As the LDES streams contains all historical data as immutable objects the model could be retrained using data retrieved from the LDES stream. After training a model per bike station is available. In the figure 28 below each dot represents one or more bike station(s). The R-value above the 75% of accuracy confirms the quality of the model.

The exhaustive description can be found in GreenMov D3.2 chapter 4.

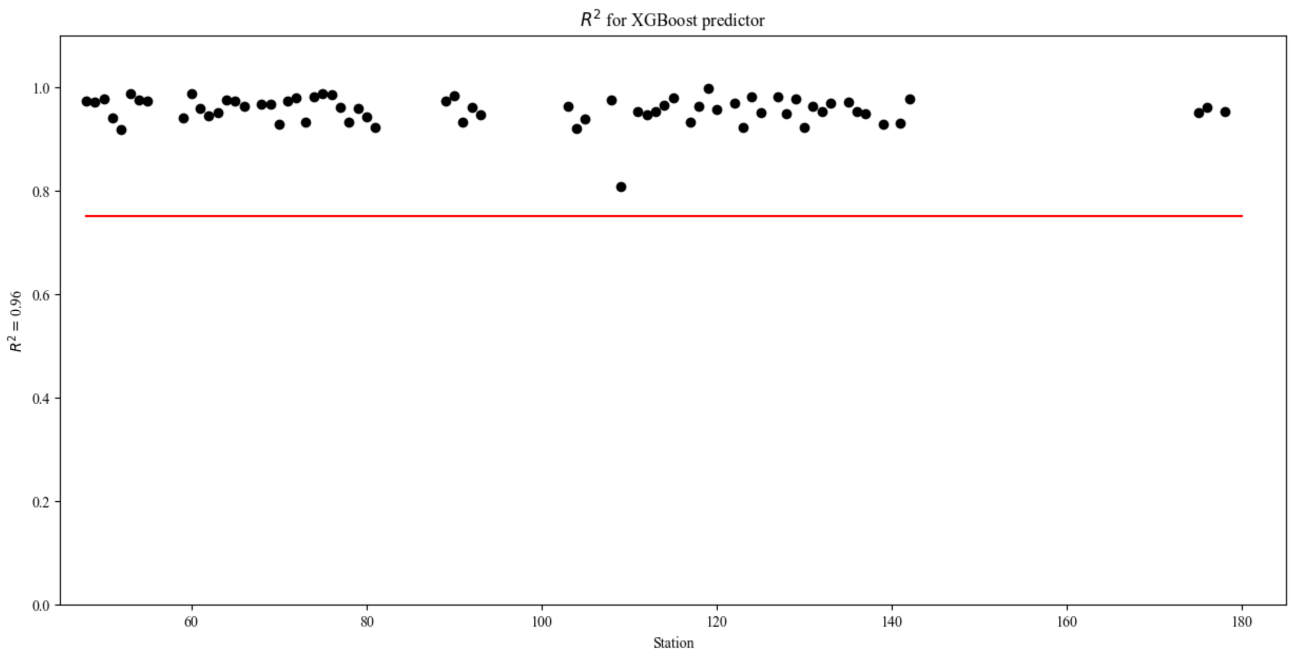


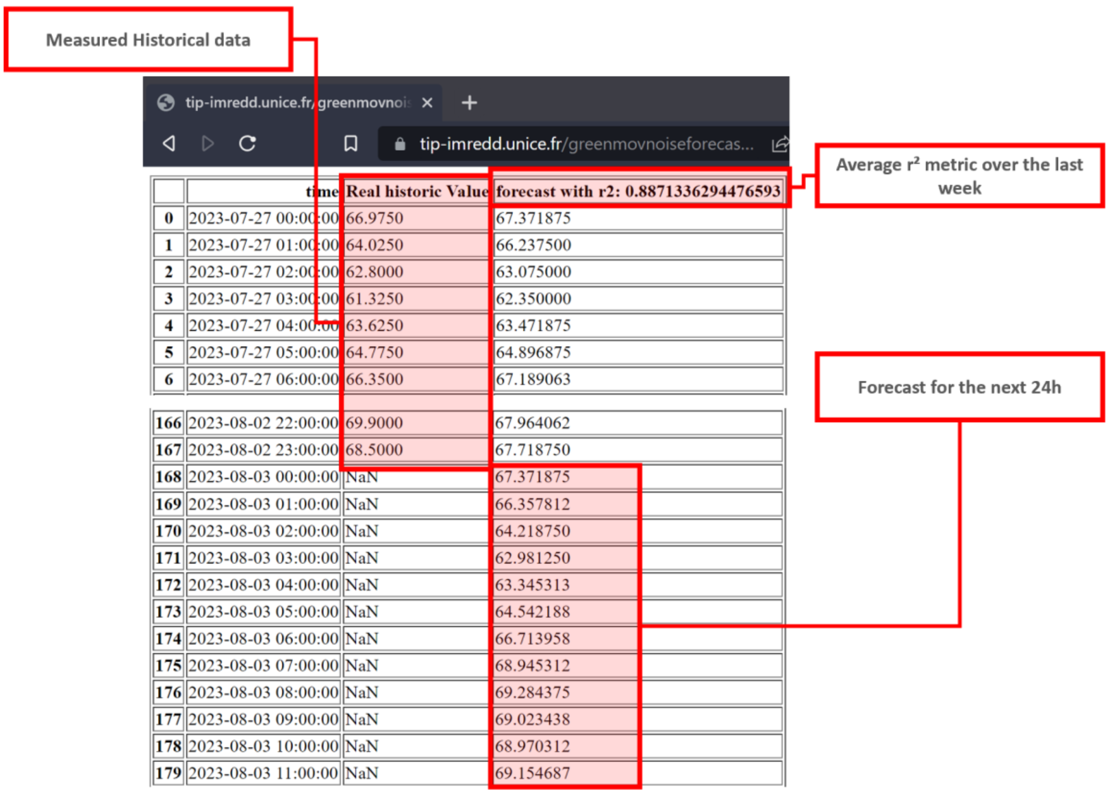
Figure 33 : R values for model per bike station.

5.5 Noise Annoyance Forecasting

Noise Annoyance Forecasting is a cutting-edge service that offers a comprehensive prediction of the level of noise annoyance in a specific geographic location within a specified time frame as shown in the Figure 34 below. This service receives inputs from the noise annoyance calculation service and is an essential component of the traffic recommendation generation service, which also uses multiple data inputs to provide valuable insights and recommendations for managing and reducing the traffic environmental impact.

The service is accessible for public via this link:

- https://tip-imredd.unice.fr/greenmovnoiseforecasting/forecast/greenmov/noise?aggregation=1H&forecast_horizon=1H, or via the greenmov Nice use case interface available here: https://tip-imredd.unice.fr/greenmovinterface/


Measured Historical data
tip-imredd.unice.fr/greenmovnoi
tip-imredd.unice.fr/greenmovnoiseforecas...
Average r² metric over the last week
Forecast for the next 24h

	time	Real historic Value	forecast with r2: 0.8871336294476593
0	2023-07-27 00:00:00	66.9750	67.371875
1	2023-07-27 01:00:00	64.0250	66.237500
2	2023-07-27 02:00:00	62.8000	63.075000
3	2023-07-27 03:00:00	61.3250	62.350000
4	2023-07-27 04:00:00	63.6250	63.471875
5	2023-07-27 05:00:00	64.7750	64.896875
6	2023-07-27 06:00:00	66.3500	67.189063
166	2023-08-02 22:00:00	69.9000	67.964062
167	2023-08-02 23:00:00	68.5000	67.718750
168	2023-08-03 00:00:00	NaN	67.371875
169	2023-08-03 01:00:00	NaN	66.357812
170	2023-08-03 02:00:00	NaN	64.218750
171	2023-08-03 03:00:00	NaN	62.981250
172	2023-08-03 04:00:00	NaN	63.345313
173	2023-08-03 05:00:00	NaN	64.542188
174	2023-08-03 06:00:00	NaN	66.713958
175	2023-08-03 07:00:00	NaN	68.945312
176	2023-08-03 08:00:00	NaN	69.284375
177	2023-08-03 09:00:00	NaN	69.023438
178	2023-08-03 10:00:00	NaN	68.970312
179	2023-08-03 11:00:00	NaN	69.154687

Figure 34 : Example of the Noise annoyance forecasting service in Nice use case.

The architecture of the service is detailed in Figure 35.

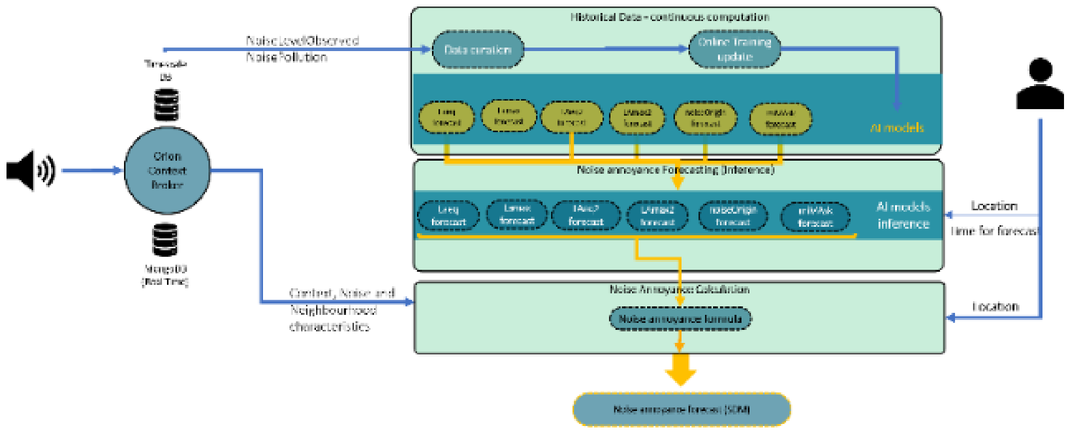
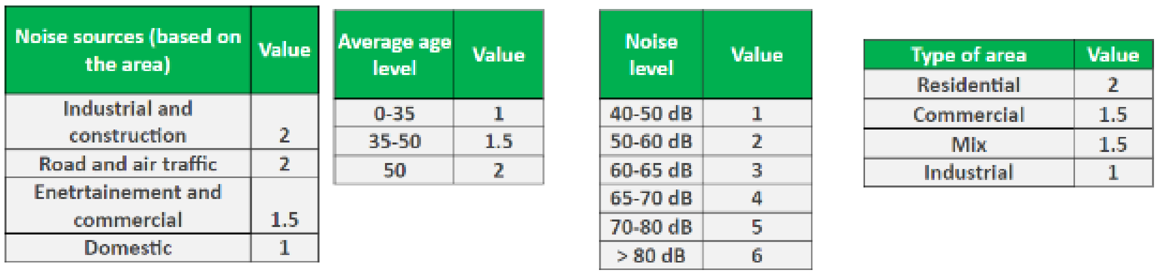


Figure 35 : Overview of the whole noise annoyance forecasting services [5].

5.5.1 Noise annoyance Calculation

The noise annoyance service is a part of the Noise annoyance forecasting process and takes into account the various parameters that contribute to noise annoyance in a determined area using specific mathematical formulas based on the parameters shown in the Figure 36 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

Figure 36 : Noise annoyance calculation parameters.

To ensure the accuracy of the calculations, the service utilizes a variety of historical data sets. These include the noise level Laeq (which is an average sound level measured every 15 minutes), the noise source, the average age level of the population in the area, and the type of area. These data sets provide a comprehensive understanding of the noise levels in a given area and are crucial to the calculation and implementation of the sub-service.

The outputs of the service are shown in the Figure 37 below:

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

Figure 37 : Noise annoyance calculation outputs.

5.5.2 Use case specificity

5.5.2.1 Nice use case specificity

Like many cities, Nice also faces an important challenge against noise pollution. For this reason the city has implemented a variety of noise pollution sensors throughout the city to collect data on noise levels. These sensors are typically installed in areas that are known to have high levels of noise, in our case we chose to use the closest sensor to our application area that is located west of the city centre and on the “Promenade des Anglais” just south of the “Voie Pierre Mathis”. The exact coordinates of the sensor are “43°40’56.4”N 7°13’58.1”E “, which make it a strategic location for our case given that it is situated between the two major traffic axis mentioned before, as shown in the Figure below. Moreover bikes, bus and tram stations are situated nearby making the perfect spot to quantify the noise pollution for the use case.

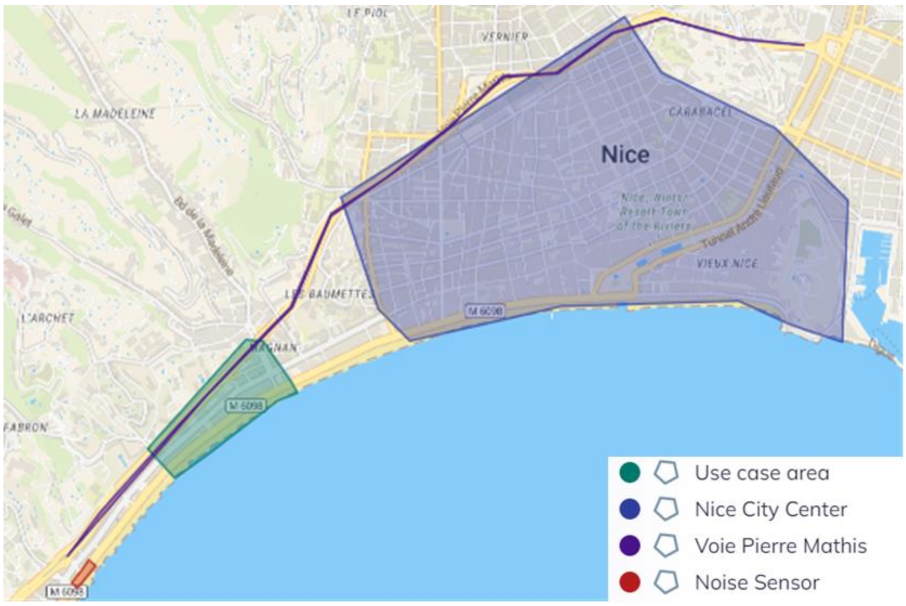


Figure 38 : Nice use case location with Noise Sensor.

Document name:	D5.3 Pilot deployment and validation					Page:	53 of 86
Reference:	D5.3	Dissemination:	PU	Version:	1.0	Status:	Final

The Noise Annoyance Calculation service for the city of Nice utilizes various IoT infrastructures, including 3rd party cloud services and databases, an NGSI-LD adaptor, and an Orion LD context broker, to accurately calculate the noise annoyance levels in a given area and within T minutes. It uses multiple parameters, including noise level Laeq, noise source, average age level, and type of area, which are accessed from historical data sets. The service also makes use of smart data models, such as NoisePollution and NoiseLevelObserved, to ensure accurate calculations. The final service is fed by NGSI-LD data from the Orion LD context broker, providing a robust and flexible data management system.

Figure 39 displays a forecast and the associated measurement (ground truth) for 5 days. The level of accuracy was 88% (r^2) for this forecast.

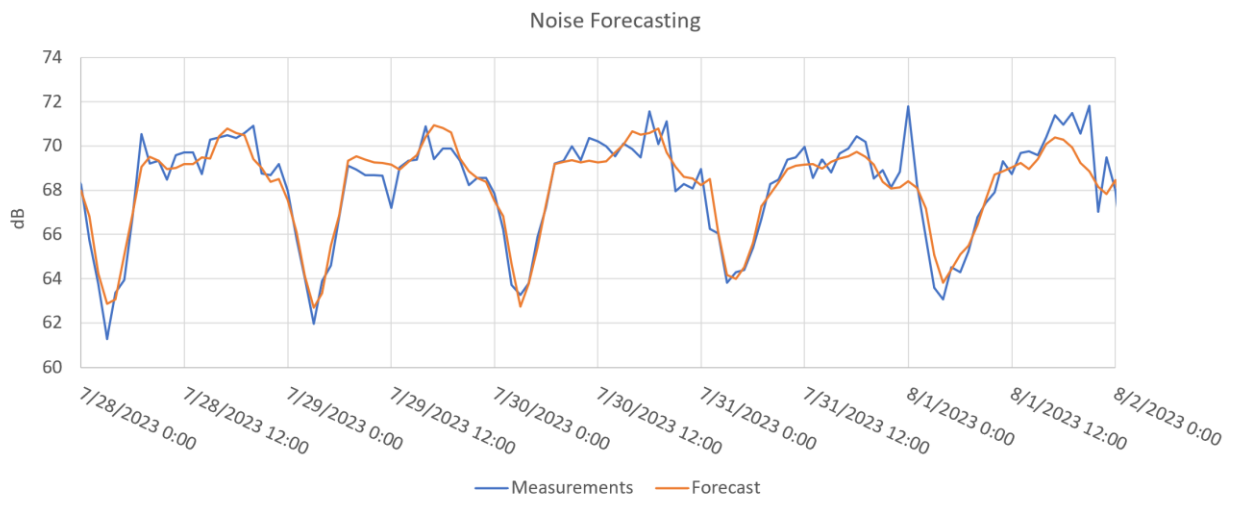
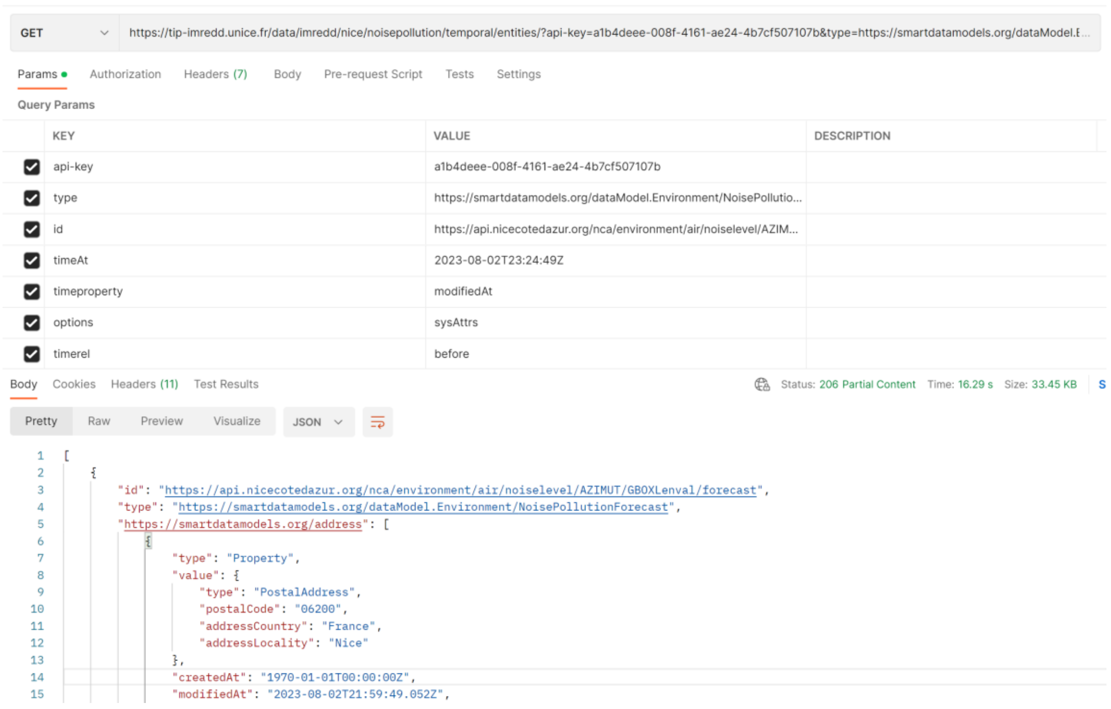


Figure 39. Noise Annoyance Forecast accuracy.

Also, Figure 40 displays a POSTMAN request to obtain the last forecasts, which were stored in the database:



GET https://tip-imredd.unice.fr/data/imredd/nice/noisepollution/temporal/entities/?api-key=a1b4deee-008f-4161-ae24-4b7cf507107b&type=https://smartdatamodels.org/dataModel.E...

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

	KEY	VALUE	DESCRIPTION
✓	api-key	a1b4deee-008f-4161-ae24-4b7cf507107b	
✓	type	https://smartdatamodels.org/dataModel.Environment/NoisePollutio...	
✓	id	https://api.nicecotedazur.org/nca/environment/air/noiselevel/AZIM...	
✓	timeAt	2023-08-02T23:24:49Z	
✓	timeproperty	modifiedAt	
✓	options	sysAttrs	
✓	timerel	before	

Body Cookies Headers (11) Test Results — Status: 206 Partial Content Time: 16.29 s Size: 33.45 KB

Pretty Raw Preview Visualize JSON

```
[
    {
        "id": "https://api.nicecotedazur.org/nca/environment/air/noiselevel/AZIMUT/GBOXLenval/forecast",
        "type": "https://smartdatamodels.org/dataModel.Environment/NoisePollutionForecast",
        "https://smartdatamodels.org/address": [
            {
                "type": "Property",
                "value": {
                    "type": "PostalAddress",
                    "postalCode": "06200",
                    "addressCountry": "France",
                    "addressLocality": "Nice"
                },
                "createdAt": "1970-01-01T00:00:00Z",
                "modifiedAt": "2023-08-02T21:59:49.052Z",
```

Figure 40 : Noise Annoyance forecast storage in NGIS-LD Context Broker

5.5.2.2 Murcia/Molina use case specificity

As well as for Air Quality, the noise annoyance forecasting service is fed by Smart Spots in the city of Molina – this service is only deployed in this city – to provide accurate and real-time forecasts of noise levels within the city, The Figure 41 below shows the location of the Smart Spot in Molina de Segura.

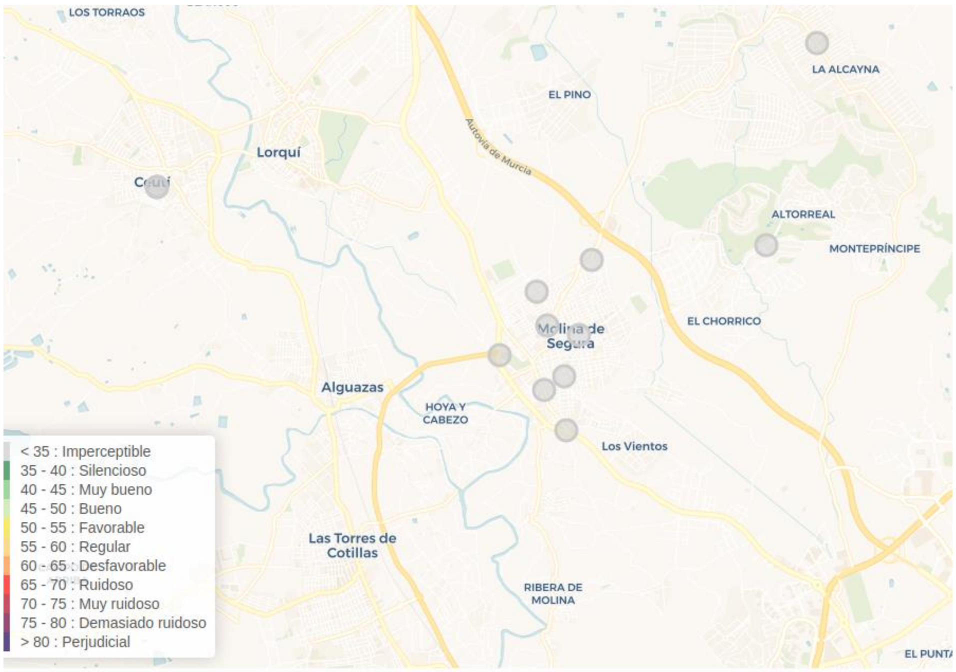


Figure 41 : Noise Sensors in Molina de Segura.

5.5.2.3 Flanders use case specificity

Flanders use case does not use the noise annoyance service.

6 Use Cases Deployment

This section describes the main deployment steps followed during the implementation phase. Figure 42 shows the main steps that were required during the deployment phase. The first main task to achieve for use case deployment consisted in the data collection task. This task was split into three main steps, that include the data identification, the stakeholder's engagement, and the design and implementation of data collection services. Stakeholders' engagement was indeed critical as it was the only way to get access to real-time data.

The second task consists in the deployment of a NGSI-LD storage architecture. Leveraging existing architectures and deployment experiences, this task consists in the choice of a storage solution, such as the context broker or the database for real time and temporal data. In the case of GreenMov, the storage architecture solution was either Orion-LD or Scorpio. Once the choice for a storage solution was done, use cases have implemented a first version of NGSI-LD data storage. However, in most use cases, services came with requirements that the proposed architectures did not fulfil, such as the retrieval of historical data, which is mandatory for services implementations. Therefore, the second task requires a continuous improvement process to improve the storage solution and ensure all the requirements from other tasks are met. Along with this task, the use cases required to implement the storage solution and the data collection software in real, fixed and exposed hardware. Therefore, a specific task was added to determine the infrastructure requirements in terms of hardware and network characteristics.

The third task consists in the design of services. The first step consisted in the conceptual design of all services, quickly followed by the local development of these services in local machines. Once training and testing are successful, the services' design task migrate into the deployment phase that consists in replicating the local work into an exposed server using replication tools such as docker-compose. Once services are deployed, the computation of KPIs requires improvements in the service, that last until the end of the experiment. Along with the services, a front-end is usually designed to ensure a good experience for the end-users. Not all use case have a front end, but this task also includes the design of interfaces with the services.

Finally, the last task of the deployment is the collection of end-users feedback, and the continuous computation of KPIs defined in deliverable D.5.1 [4].

The rest of the section details each of these task's implementations for all the use cases.

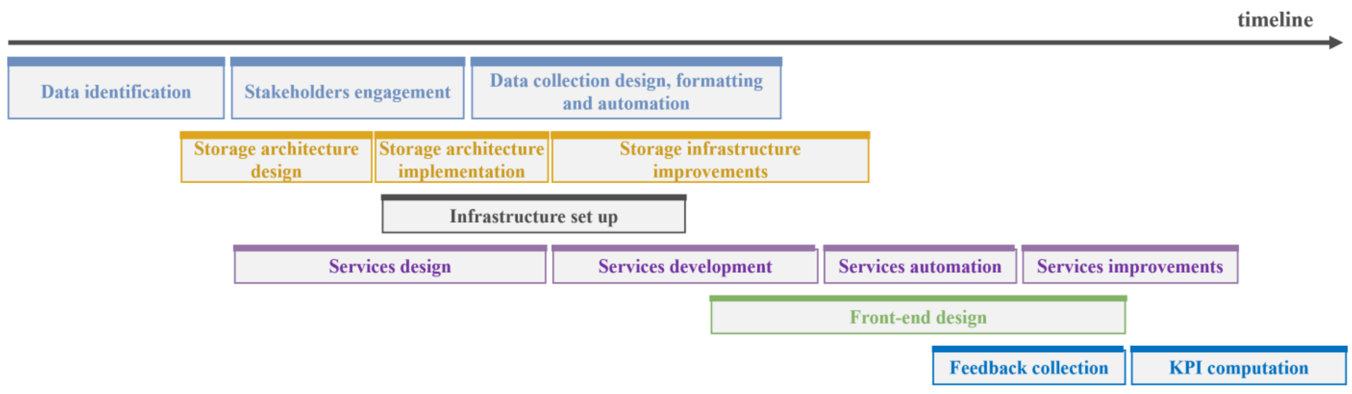


Figure 42 : Use cases deployment steps.

6.1 Nice

6.1.1 Deployment Steps

In the context of the Nice use case, the steps proposed in Figure 42 were all reproduced. Everything was finalised and deployed in July 2023 and is working since then. Validation was done throughout the month of August.

6.1.2 Technical Deployment

The technical deployment of the use case in Nice was done following the infrastructure description detailed in Figure 8. Two main servers were deployed to host the data collection services, the data storage solution, and the services.

- Data collection was achieved through Node-RED flows that gather data and format it in the right smart data model. An example of such implementation is provided in Figure 43.

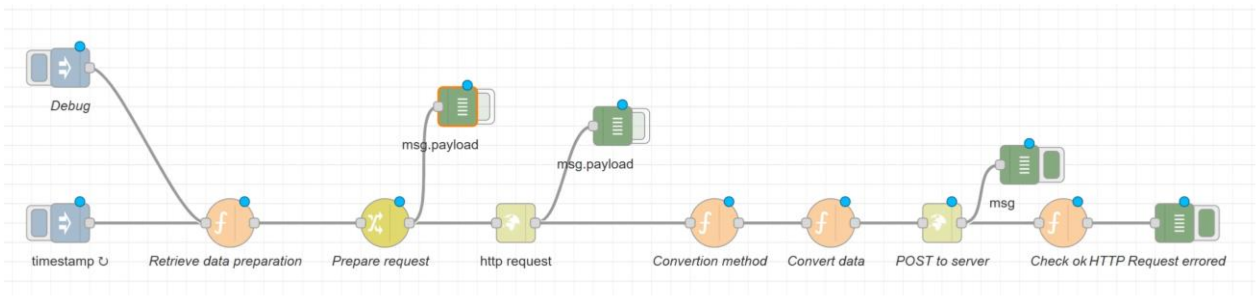
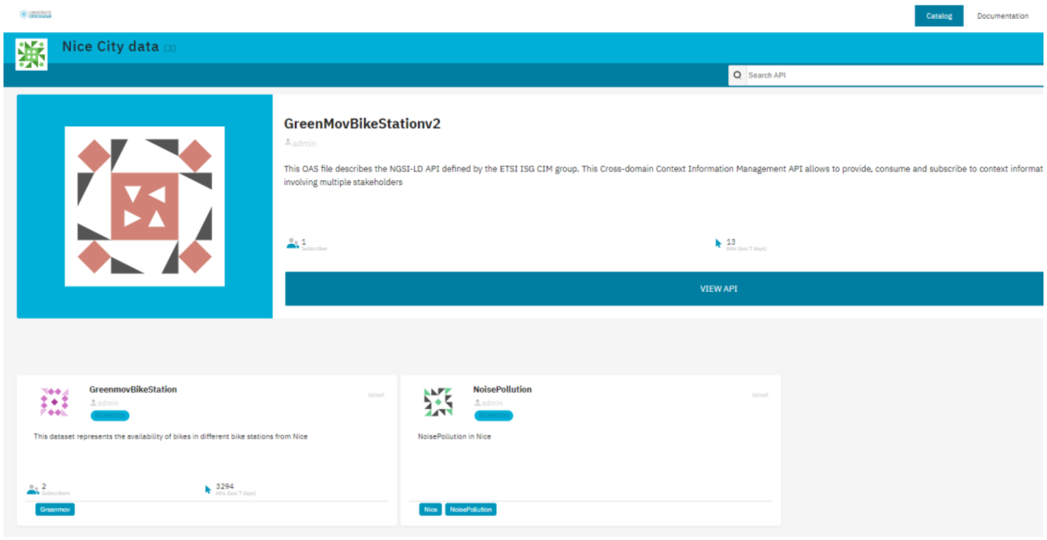


Figure 43 : Example of NodeRed flow implementation for air quality data collection in Nice.

- Storage in the Nice use case uses Orion-LD as a context broker, with Mintaka and TimescaleDB for temporal data storage. Although the basic implementation process was successful, services quickly required last N stored values. However, it appeared that this simple task was made difficult because of the choice for data models without an “observedat” attribute, and because Orion-LD is expecting to find such a parameter so it can sort the values. Solving the issue required to engage with FIWARE Foundation technical advisors. Also, in the Nice use case, an API manager was implemented along with a NGINX proxy to provide the following advantages:

- An accessible catalog of the data set available, as shown in Figure 44.



This OAS file describes the NGSI-LD API defined by the ETSI ISG CIM group. This Cross-domain Context Information Management API allows to provide, consume and subscribe to context informat involving multiple stakeholders

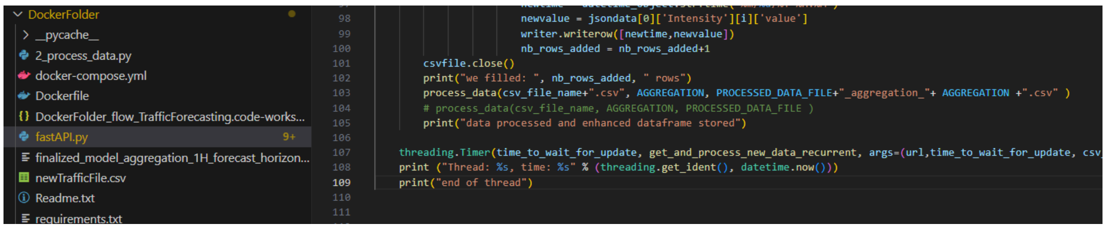
This dataset represents the availability of bikes in different bike stations from Nice

NoisePollution in Nice

Figure 44 : API manager catalogue example for Nice.

- A rerouting process that allows to replace the standard URLs such as "http://orion-ld:1026/ngsi-ld/v1/" or "http://mintaka:8080/temporal/" by an explicit URL such as: https://tip-imredd.unice.fr/data, then followed by the entity details, as for example: "/imredd/nice/bikestation/temporal/entities/urn:ngsi-ld:GreenMov-Nice-VeloBleu-BikeHireDockingStation-446".
- Secure access to data through API key management.

- Services deployment was achieved through two main steps in the case of Nice: in the first step, services were designed in local computers and included benchmarking of Machine Learning models and training dataset size. Once the local implementation was working, the development team added docker compose compatibility to help deploying and replicating the service. This required the main server to have docker-compose installed. Also, because the use case of Nice relies on automatic daily updates on traffic recommendations, a threading timer was defined in docker to start the traffic recommendation service on a daily basis for the day ahead. Also, in Nice, all APIs are using FastAPI except for air quality related services that use kserve component. An example of service architecture is proposed in Figure 45.



```
                    newvalue = jsondata[0]['Intensity'][i]['value']
                    writer.writerow([newtime,newvalue])
                    nb_rows_added = nb_rows_added+1
        csvfile.close()
        print("we filled: ", nb_rows_added, " rows")
        process_data(csv_file_name+".csv", AGGREGATION, PROCESSED_DATA_FILE+"_aggregation_"+ AGGREGATION +".csv" )
        # process_data(csv_file_name, AGGREGATION, PROCESSED_DATA_FILE )
        print("data processed and enhanced dataframe stored")

    threading.Timer(time_to_wait_for_update, get_and_process_new_data_recurrent, args=(url,time_to_wait_for_update, csv
    print ("Thread: %s, time: %s" % (threading.get_ident(), datetime.now()))
    print("end of thread")
```

Figure 45 : Zoom on a docker compose and threading for tasks scheduling.

- The interface is also composed of an html, css and javascript files that are hosted locally at IMREDD' servers.
- Finally, a monitoring of all statuses in the NGSI-LD data base and on the services health is provided in case a task is crushing or in case data is not available anymore.

6.1.3 Stakeholders and End-users engagement

Stakeholders' engagement was critical in the case of the Nice use case for several reasons:

- First, during the data collection process, it appeared that many data that were supposed to be fully and publicly available in real-time were not available. Therefore, a significant effort aimed to build strong relationships with data owners, which include the metropole of Nice Côte d'Azur, as well as their subcontractors who provide them with IoT services, such as traffic measurement, bikes availability or noise data frame.
- Second, the storage implementation led the Nice use case to engage with FIWARE Foundation expert teams to solve two unrelated issues.
    - The first one consisted in the fact that the database's size was getting too big after a few months of data collection, mostly because of the way linked data are stored. A proposed solution was to add indexing of the database, and to partition the data base to help the context broker to retrieve temporal data.
    - The second issue was related to the difficulties for Orion-LD (and Scorpio) to provide the last N values stored for a given entity. This happened to be mostly due to the fact that the context broker is expecting to store data with a time parameter named "observedat" and is actually storing time parameters named "dateobserved". As a result, the context broker is not able to sort the data by date of observation. This was solved for Orion-LD implementation by using a built-in time parater named "ts" for timestamp. However, engineers from Fiware Foundation taught us to use the timerel system attribute to get all the data available before (or after) a certain date: https://tip-imredd.unice.fr/data/imredd/nice/noisepollution/temporal/entities/?api-key=a1b4deee-008f-4161-ae24-4b7cf507107b&type=https://smartdatamodels.org/dataModel.Environment/NoisePollution&timeproperty=modifiedAt&options=sysAttrs&id=https://api.nicecotedazur.org/nca/environment/air/noiselevel/AZIMUT/GBOXLenval&lastN=2&attrs=Lamax2&timerel=before&timeAt=2023-08-21T15%3A48%3A37Z .
      However, issues of timeout appeared when using the "after" timerel instead of the "before", which created several issues for the forecasting services (it is better/more efficient to get data from after a certain date (the date of the last data gathered) than data before now). Indenting the temporal database seemed to improve the speed of response of the database.
- Finally, interaction with the end-users was realised through a WebApp that was presented in Figure 5, which allows end-users to select the time of the forecasts and traffic recommendations, but also to visualise the accuracy of the predictions.

6.1.4 Replication Process

The implementation of all services is done with docker-compose, which ensures replicability of the solutions. Then, docker-compose files for data storage solution are available in the Gitlab page of the project so anyone can replicate the proposed architecture. Finally, node red flows are also included in the gitlab page for replication purposes.

6.2 Murcia/Molina

6.2.1 Deployment Steps

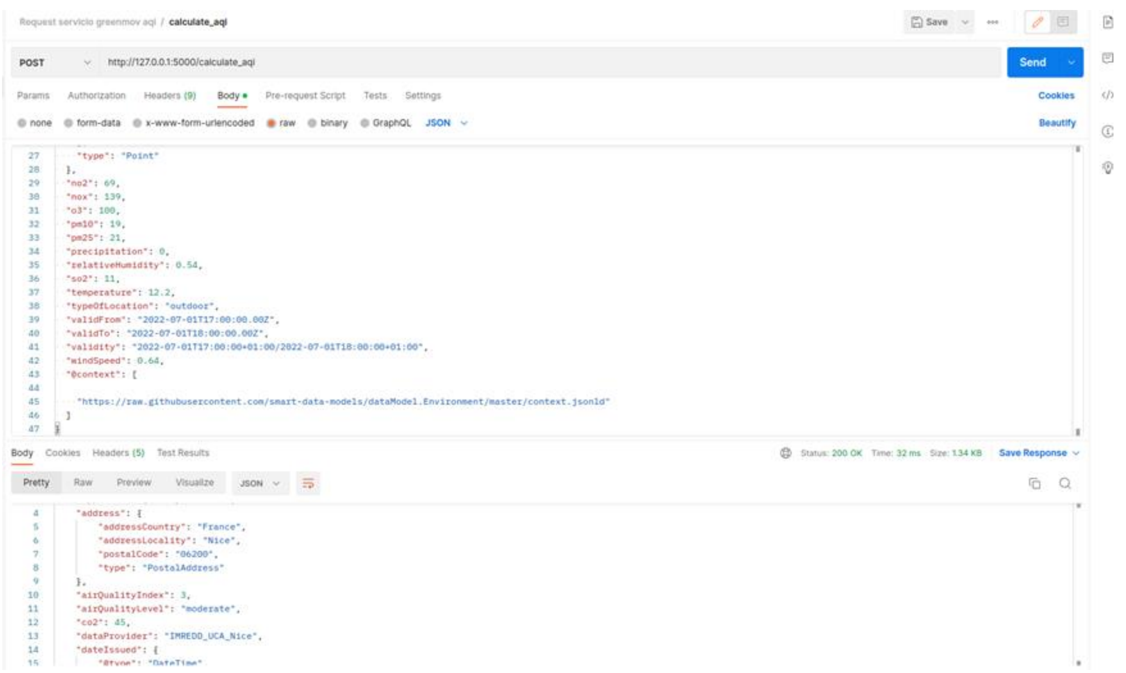
As well as in Nice, the steps proposed in Figure 42 were followed in the deployment of the use case in Murcia/Molina. Next, the status of each one of the steps is summarized and updated.

- Data identification and collection: all the air quality/noise sensors as well as the bike stations are working and reporting data. Status: Completed
- Architecture design and implementation: the city context brokers, and the top context broker are deployed. In addition, the Molina context broker is federated with the top context broker. It remains the federation of Murcia context broker. The persistence components (QuantumLeap + CrateDB) are deployed. Status**:** Completed.
- Services deployment: all the services are working, but it was necessary to connect them to the final architecture to finalize the deployment. Status: Finalisation phase.
- Front-End implementation: Status: Finalisation now that the last services have been deployed.
- Use case validation: Status: partially completed as the Front-End integration will be fully validated at the end of August / very early September.

6.2.2 Technical Deployment

For the deployment in Murcia/Molina the infrastructure proposed in Figure 13 : Infrastructure in Murcia/Molina use case.was used. The storage facility was implemented within Azure server to ensure high availability and scalability. The entire back end was dockerised to ensure high replicability and easy maintenance. Similarly, the front end is also hosted by Azure servers, for the same reasons.

- Data collection was done through different APIs. In the case of HOPU's devices, the data is accessible through HOPU internal API, that perform a request to the CrateDB database. In this case, as HOPU is member of the FIWARE foundation and strongly follows the guidelines and architectures provided by them, these data are modeled with the correcto data models. On the other hand, regarding external sources as the European Air Quality Portal or the AEMET, public APIs has been used and the proper adaptors to the correct data models has been deployed. An example of this workflow is showed in Figure 46.
- Data storage in Murcia/Molina uses Orion-LD as local context broker for the cities and Scorpio for the central broker. Regarding persistence, it is reached through a QuantumLeap connector to a CrateDB database.
- Services deployment was achieved through two main steps. In the first step, services were designed in local computers. Once the local implementation was working, the service was dockerice by creating docker compose compatibility to help deploying and replicating the service. This process was tested in local, though POSTMAN.



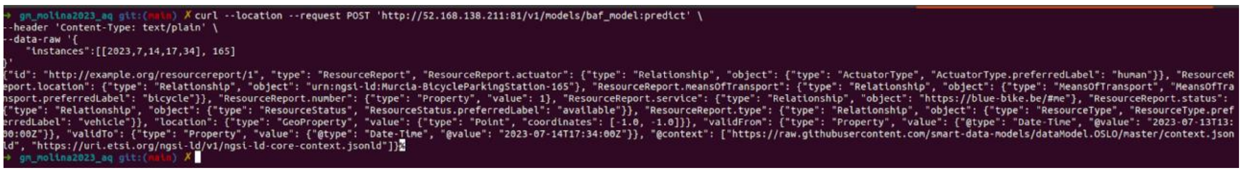
```
    "type": "Point"
  },
  "no2": 69,
  "nox": 139,
  "o3": 100,
  "pm10": 19,
  "pm25": 21,
  "precipitation": 0,
  "relativeHumidity": 0.54,
  "so2": 11,
  "temperature": 12.2,
  "typeOfLocation": "outdoor",
  "validFrom": "2022-07-01T17:00:00.00Z",
  "validTo": "2022-07-01T18:00:00.00Z",
  "validity": "2022-07-01T17:00:00+01:00/2022-07-01T18:00:00+01:00",
  "windSpeed": 0.64,
  "@context": [

    "https://raw.githubusercontent.com/smart-data-models/dataModel.Environment/master/context.jsonld"
  ]
}
```

```
    "address": {
        "addressCountry": "France",
        "addressLocality": "Nice",
        "postalCode": "06200",
        "type": "PostalAddress"
    },
    "airQualityIndex": 3,
    "airQualityLevel": "moderate",
    "co2": 45,
    "dataProvider": "IMREDD_UCA_Nice",
    "dateIssued": {
        "@type": "DateTime"
```

Figure 46 : Example of Murcia/Molina POSTMAN requests.

Then, the services were deployed in a production machine (Azure) and it was tested that is it possible to reach the service through CURL.



```
→ gm_molina2023_aq git:(main) ✗ curl --location --request POST 'http://52.168.138.211:81/v1/models/baf_model:predict' \
--header 'Content-Type: text/plain' \
--data-raw '{
    "instances":[[2023,7,14,17,34], 165]
}'
{"id": "http://example.org/resourcereport/1", "type": "ResourceReport", "ResourceReport.actuator": {"type": "Relationship", "object": {"type": "ActuatorType", "ActuatorType.preferredLabel": "human"}}, "ResourceR
eport.location": {"type": "Relationship", "object": "urn:ngsi-ld:Murcia-BicycleParkingStation-165"}, "ResourceReport.meansOfTransport": {"type": "Relationship", "object": {"type": "MeansOfTransport", "MeansOfTra
nsport.preferredLabel": "bicycle"}}, "ResourceReport.number": {"type": "Property", "value": 1}, "ResourceReport.service": {"type": "Relationship", "object": "https://blue-bike.be/#me"}, "ResourceReport.status":
{"type": "Relationship", "object": {"type": "ResourceStatus", "ResourceStatus.preferredLabel": "available"}}, "ResourceReport.type": {"type": "Relationship", "object": {"type": "ResourceType", "ResourceType.pref
erredLabel": "vehicle"}}, "location": {"type": "GeoProperty", "value": {"type": "Point", "coordinates": [-1.0, -1.0]}}, "validFrom": {"type": "Property", "value": {"@type": "Date-Time", "@value": "2023-07-13T13:
00:00Z"}}, "validTo": {"type": "Property", "value": {"@type": "Date-Time", "@value": "2023-07-14T17:34:00Z"}}, "@context": ["https://raw.githubusercontent.com/smart-data-models/dataModel.OSLO/master/context.json
ld", "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"]}%
→ gm_molina2023_aq git:(main) ✗
```

Figure 47 : Example of the cron job in the Docker file.

Also, because the use case of Molina-Murcia is based on automatic periodic updates of the forecasts, a cron job was defined in docker to trigger the services every 6 hours to provide new recommendations.

6.2.3 Stakeholders and End-users engagement

During the data collection process, it was noticed that many data that were not available in open data portals, especially in the case of Molina de Segura. For this reason, a significant effort has been done between HOPU and cities to generate the data sets an upload it to the corresponding data portals, which include air quality data. Furthermore, this engagement with the cities will be pursued even after the project to take advantage of future datasets that will be open.

The engagement also includes the relationship with end-users, that focused first on the GreenMov team members, before expanding to employees from HOPU and city councils, and finally to all stakeholders from the use case. These stakeholders were also made aware of all the work that was done within GreenMov during dissemination activities, so they are aware of GreenMov 's software components.

On the other hand, the storage implementation was done according to Fiware guidelines. One of the issues still pending is the increasing size of the database over the time. The current solution consists in overwriting the values every year, after extracting the old data and storing in a different database – and publishing into open data portals. The second issue was related to the difficulties for Scorpio to provide the last N values stored for a given entity, as for the case of Nice with Orion-LD.

6.2.4 Replication Process

The implementation of all services is done with docker-compose, which ensures replicability of the solutions. Then, docker-compose files for data storage solution are available on the Gitlab page of the project so anyone can replicate the proposed architecture.

6.3 Flanders

6.3.1 Deployment Steps

For what concerns the Flanders' use case, the steps proposed in Figure 42 were all reproduced, as shown below.

6.3.2 Technical Deployment

In Flanders, the deployment of the use case used the infrastructure proposed in Figure 12. The storage facility was implemented within Azure server to ensure high availability and scalability. Then, Kubernetes was used to deploy the docker containers. The entire back end was dockerised to ensure high replicability and easy maintenance. Several issues were encountered while implementing the back end, especially for what relates to the context broker and the database itself as advanced features of new versions of the software were used. As mentioned above for Nice, discussions with Fiware experts and exchanges within the GreenMov consortium led to selecting Orion-LD as context broker. After multiple iterations the correct configuration of Orion-LD was achieved. Similarly, the front end is also hosted by Azure servers, for the same reasons as in Nice. The development of the front end went fast as state of the art proven solutions were used. For better integration, the NEXT.js framework was used along with React.js for a smooth user experience. All the components are running as docker containers on the Azure Kubernetes Service (AKS). The deployment of all the components has been done using helm charts expect for the context broker (first Scorpio, then Orion-LD), front-end client, LDES-replicator and NGSI-LDES, which consist of plain deployment files.

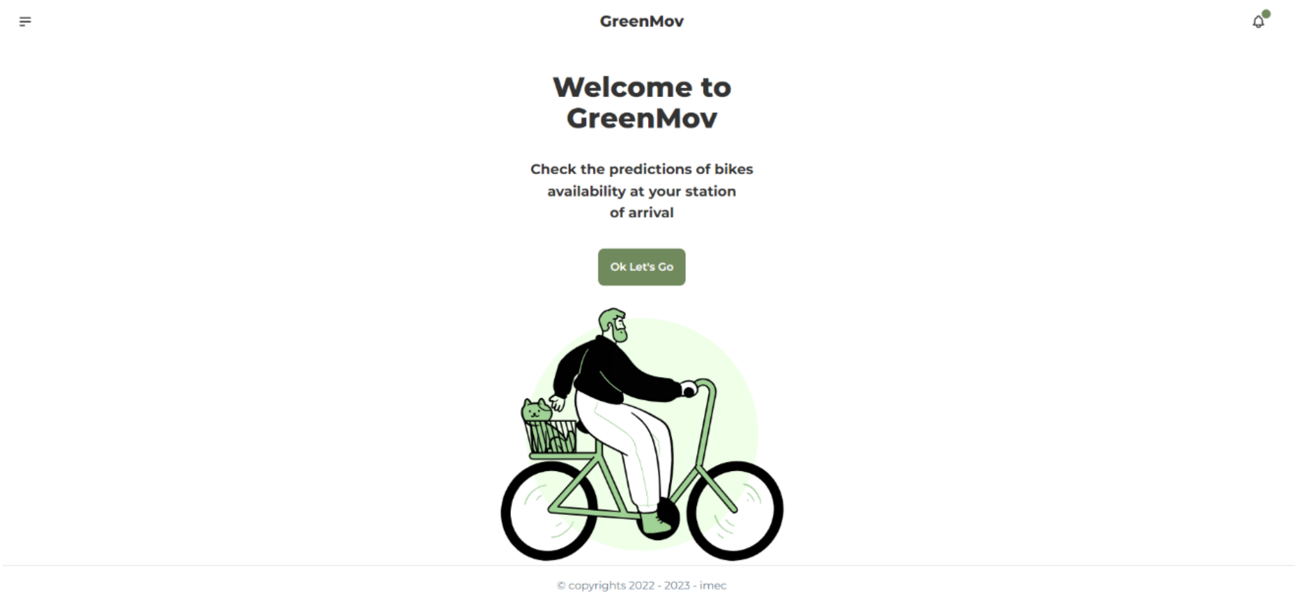
All the helm charts are open-source and are the docker containers are available at Fiware GitHub repository and Fiware Docker Hub. [14]. The deployment YAMLs for Scorpio Context Broker are available in Scorpio Broker GitHub repository. The deployment YAML s for LDES-replicator, NGSI-LDES and front-end client are available in the Imec GitHub repository.

6.3.3 Stakeholders and End-users engagement

For this use case, Imec and AIV have been able to leverage their relationship with the organisation responsible for Blue-bikes, that publishes the data of bikes availability. They agreed that their data was used in GreenMov. In GreenMov the data were "upgraded" to linked data formats LDES and NGSI-LD. The team used the open https://planner.js.org/ software to access train data for this use case. Finally, strong involvement with the OSLO interoperability team of AIV helped in achieving the right data models architecture and alignment with the Flanders Smart Data Space principles and technical standards.

The engagement also includes the relationship with end-users, that focused first on the GreenMov team members, before expanding to employees from Imec, and finally to all stakeholders from Imec's ecosystem. These stakeholders were also made aware of all the work that was done within GreenMov, so they are aware of GreenMov's software components, and aware of the fact that these software components can be downloaded from their respective repository on https://artifacthub.io/. The components without helm charts can be delivered by converting the deployment files into helm charts and uploading them on https://artifacthub.io/ from where they can be downloaded.

The interaction with end-users was realized via an online application available via a dedicated website [15], the front end of the application is as shown in Figure 48.


GreenMov

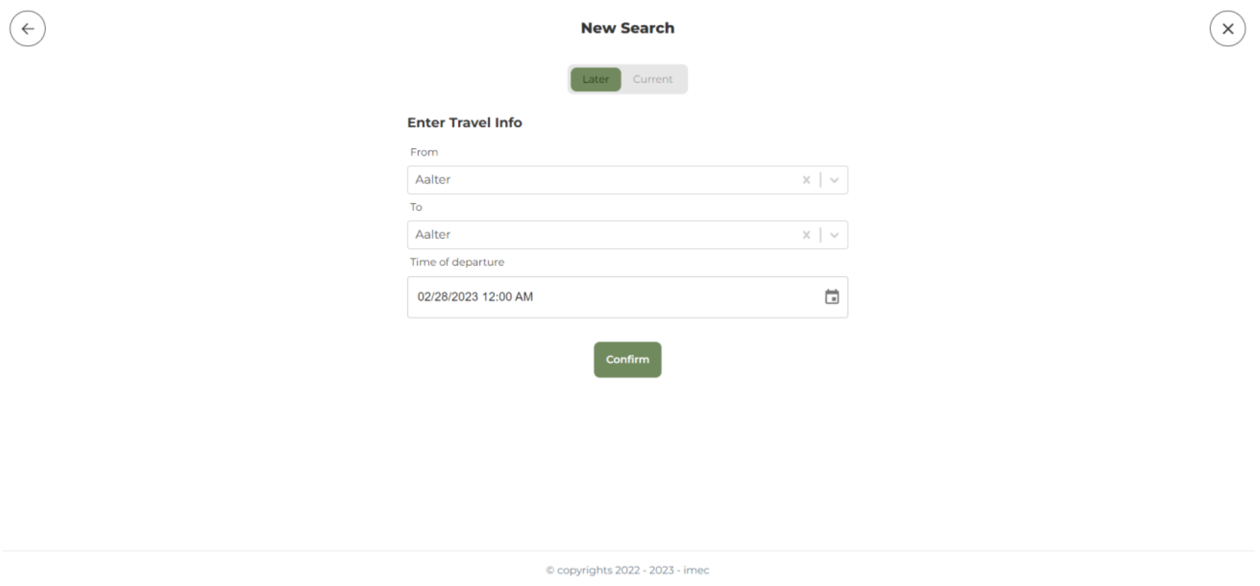
Welcome to GreenMov

Check the predictions of bikes availability at your station of arrival

© copyrights 2022 - 2023 - imec

Figure 48 : Bikes availability application welcome page.

The search feature of the application is as shown in Figure 49.


New Search

Enter Travel Info

From: Aalter

To: Aalter

Time of departure: 02/28/2023 12:00 AM

© copyrights 2022 - 2023 - imec

Figure 49 : Bikes availability application search page.

6.3.4 Replication Process

Since all the components are running as docker containers, they can easily be run on any Kubernetes platform. The end user deploys the helm chart on its Kubernetes platform and the software components can be run. The same is valid for the LDES-stream that can be deployed in any hosting environment.

7 Use Case Validation Plan

Each use case that implemented technical solutions has used a specific validation plan to validate each part of the end-to-end solution. The use case validation plan includes the validation of the following components: storage infrastructure, data collection, services and end-users' interaction. On top of these components' validation, the validation plan also includes an overarching validation of the use cases. This section details the process of the validation plan following the structure below:

- 1.1: Data storage infrastructure:
  - 1.1.1: Storage of last data: this unitary test consists in sending a known data with a known time stamp to the data storage solution, and to retrieve it using the context broker API. The context broker should retrieve the data sent.
  - 1.1.2: Storage and retrieval of historic data: this unitary test consists in sending a known series of data with known timestamps to the data storage solution, and to retrieve the last N values using the context broker API. The context broker should retrieve the last N data sent.
  - 1.1.3: Secure access to data: this unitary test consists in retrieving the last data stored for a given entity without using the secure process (Keyrock or Gravitee). The context broker should not retrieve any data and an "Unauthorized" reply should be received. Similarly, when using the right API key or security process, the context broker should retrieve and reply to the requested data.
  - 1.1.4: The time for last N data retrieval with N<100 should be less than 3 seconds.

  Then, other capabilities of the context broker were assessed within the validation plan, such as:

  - 1.1.5: the list of entities by type
  - 1.1.6: the selection of a specific parameter
  - 1.1.7: the following of specific operations as mentioned in the document: ETSI GS CIM 009 V1.4.1 (2021-02) [16]
  - 1.1.8: the update of a specific parameter for a given entity.
- 1.2: Data collection and formatting:
  - 1.2.1: For each data source (traffic intensity, noise, air quality and bikes), the main unitary test consists in retrieving the last data stored in the NGSI-LD storage solution. The data retrieved by the context broker should be:
    - updated within the last hour.
    - match the corresponding smart data model.
    - be accurate, as validated by an expert or by on-site check.
  - 1.2.2: For one complete day, ensure that data is updated continuously by setting up a continuous check (using a dedicated check service, developed in Python or javascript/nodered), with the right data format, and accurate value as validated by an expert (noise level within specific ranges).
- 1.3: Services:
  Traffic forecasting:

- 1.3.1.1: the unitary test consists in requesting a traffic forecast for the next hour. The service should send back a forecast in the right data format. After one hour, the value predicted by the service should be checked against real value as provided by the context broker.
- 1.3.1.2: The unitary test described above should be repeated over a whole day of forecast, to compute the average r^2 value between the predictions and measurement stored in the context broker.
- 1.3.1.3: A secondary test consists in assessing the time of reply of the service, that should be below three seconds.

Traffic environmental computation:

- 1.3.2.1: the unitary test consists in requesting a traffic forecast for the next hour, and a traffic environmental impact forecast for the next hour. Using the traffic forecast, it should be checked manually that the environmental impact was well calculated.
- 1.3.2.2: The response time of the whole chain (request for a traffic environmental impact forecast) should be measured (e.g., using postman) and below 3seconds.

Traffic recommendations:

- 1.3.3.1: Unitary tests consist in sending fake forecasts data to the service to ensure all recommendations are sound (e.g., fake rainy weather forecast should lead to recommendations for using public transport and not shared bikes, a large environmental impact from the traffic should lead to a recommendation to reduce the traffic, …).
- 1.3.3.2: A request for traffic recommendations for the next day (each hour of the next day) along with bike availability forecasts, traffic environmental impact forecast, noise forecast and Air quality index forecast, public transport GTFS (requested to the context broker) and weather forecast. Soundness of the reply should be checked.

Air Quality forecasting:

- 1.3.4.1: the unitary test consists in requesting an Air quality forecast for the next hour. The service should send back a forecast in the right data format. After one hour, the value predicted by the service should be checked against real value as provided by the context broker.
- 1.3.4.2: The unitary test described above should be repeated over a whole day of forecast, to compute the average r^2 value between the predictions and measurement stored in the context broker.
- 1.3.4.3: A secondary test consists in assessing the time of reply of the service, that should be below three seconds.

Air quality index calculation:

- 1.3.5.1: the unitary test consists in requesting a air quality forecast for the next hour, and a air quality index forecast for the next hour. Using the air quality forecast, it should be checked manually that the air quality index was well calculated.
- 1.3.5.2: The response time of the whole chain (request for an air quality index forecast) should be measured (e.g., using postman) and below 3seconds.

Noise level forecasting:

- 1.3.6.1: the unitary test consists in requesting a noise forecast for the next hour. The service should send back a forecast in the right data format. After one hour, the value predicted by the service should be checked against real value as provided by the context broker.

- 1.3.6.2: The unitary test described above should be repeated over a whole day of forecast, to compute the average r² value between the predictions and measurement stored in the context broker.
- 1.3.6.3: A secondary test consists in assessing the time of reply of the service, that should be below three seconds.

Noise annoyance calculation:

- 1.3.7.1: the unitary test consists in requesting a noise forecast for the next hour, and a noise annoyance forecast for the next hour. Using the noise forecast, it should be checked manually that the noise annoyance forecast was well calculated.
- 1.3.7.2: The response time of the whole chain (request for a noise annoyance forecast) should be measured (e.g., using postman) and below 3seconds.

Bikes availability forecasting:

- 1.3.8.1: the unitary test consists in requesting a bike availability forecast for the next hour. The service should send back a forecast in the right data format. After one hour, the value predicted by the service should be checked against real value as provided by the context broker.
- 1.3.8.2: The unitary test described above should be repeated over a whole day of forecast, to compute the average r² value between the predictions and measurement stored in the context broker.
- 1.3.8.3: A secondary test consists in assessing the time of reply of the service, that should be below three seconds.

- 1.4: Users interaction:
  - 1.4.1: functional validation:
  - 1.4.1.1: the end-user front end should be working on any device accessible to end-users (phones, tablets, computers)
  - 1.4.1.2: the end-user front end should allow the user to find relevant information for his/her transportation.
  - 1.4.1.3: Subscription to the services and hourly/daily check that recommendations are sent to the end-users.
  - 1.4.1.4: assessment of the speed of services procurement. Should be below 3 seconds for each service under a normal WIFI based internet connection.
- 1.4.2: UX validation:
  - 1.4.2.1: Ergonomic of the front end validated over a group of 20 testers.
- 1.5: End to end:
  - 1.5.1: The end-to-end validation consists in ensuring that the services outputs sent to the end-user correspond to the data collected. For each service, the test is done by requesting outputs from each considered service. The accuracy and soundness of the result from the services output should be validated once the real future data is available, using r² values.

7.1 Nice

7.1.1 Functional Validation

In the context of the Nice use case, the following results were obtained for the functional validation:

- Data storage infrastructure:
  - Although an issue was highlighted on the access of the last N temporal data, all functional requirements were validated in January 2023.
- Data collection and formatting:
  - Functional validation was done for traffic measurement, Noise, Air quality and bikes measurements. We note however that data from air quality and noise is data that is 24 hours old.
- Services:
  - At the time of this report's redaction, accuracy of the forecasting services was validated against several days of data, and formulas calculation was validated as well.
- Users' interaction:
  - End-users' interaction for Nice is restricted to a daily email that includes recommendations. This was validated over several days of recommendations' accuracy check. Then, the responsivity of the API for traffic recommendations was also technically assessed, especially against time for recommendations.
- End to end:
  - The end-to-end validation in the use case of Nice was achieved during the months of July and August 2023.

7.1.2 Qualitative Validation

In the case of the Nice use case, end-users only receive notifications by email or through the accessible API. A group of 12 end-users was considered to test the suitability of recommendations sent by email or through the API.

Table 8: Validation plan results for Nice use case.

Validation test number	Result	Comments
1.1.1	Passed	
1.1.2	Passed	
1.1.3	Passed	
1.1.5	Passed	

Validation test number	Result	Comments
1.1.6	Passed	
1.1.7	Passed	
1.1.8	Passed	
1.2.1	Passed	
1.2.2	Passed	Manual check after automatically computing the time interval between recommendations
1.3.1.1	Passed	
1.3.1.2	Passed	
1.3.1.3	Passed	
1.3.2.1	Passed	
1.3.2.2	Passed	
1.3.3.1	Passed	
1.3.3.2	Passed	
1.3.4.1	Passed	
1.3.4.2	Passed	
1.3.4.3	Passed	
1.3.5.1	Passed	
1.3.5.2	Passed	
1.3.6.1	Passed	
1.3.6.2	Passed	

Validation test number	Result	Comments
1.3.6.3	Passed	
1.3.7.1	Passed	
1.3.7.2	Passed	
1.3.8.1	Passed	
1.3.8.2	Passed	
1.3.8.3	Passed	
1.4.1.3	Passed	
1.4.1.4	Passed	
1.5.1	Passed	

7.1.3 KPIs Validation

The list of the validated KPIs is shown in the table below:

Table 9: Validated KPIs for Nice use case.

Indicator name	Description	Quantification range	Target	Frequency	Status
Number of recommendations per day	Refers to the number of recommendations provided by the services implemented, A recommendation can either be a short-term recommendation to change transportation for public transport, or to not change anything.	Highest is best.	1	Everyday	Validated
Accuracy of forecast services	Accuracy of the services forecasting models.	[0-100%] 100% is best	75%	Everyday	Validated 79% r^2 average accuracy over all services

Indicator name	Description	Quantification range	Target	Frequency	Status
Number of stakeholders receiving recommendation per day	Refers to the stakeholders of the project addressed by the implemented solution.	Highest is best.	2	Everyday	Validated Recommendations sent to end users and city's decision makers + User Interface on a web App

7.2 Murcia/Molina

7.2.1 Functional Validation

In the context of the Murcia/Molina use case, the following results were obtained for the functional validation:

- Data storage infrastructure:
    - All the data infrastructures are deployed, including the context brokers and the persistence components. It remains to federate the Murcia context broker due to a change in the location of the broker.
- Data collection and formatting:
    - It is possible to extract all the noise and air quality data from the sensors as well as the bike availability from the sensors.
- Services:
    - At the time of this report's redaction, accuracy of the forecasting services was validated against several days of data, and formulas calculation was validated as well.
- Federation:
    - At the time of this report, the federation of the brokers has been completed. This process has been done through Apache NiFi, by implementing a set of routines that guarantees that the data reach the top Context Broker in a proper format. To validate this process. Figure 50 shows the workflow followed in this process and the validation has been performed by tracking the transformation in NiFi.

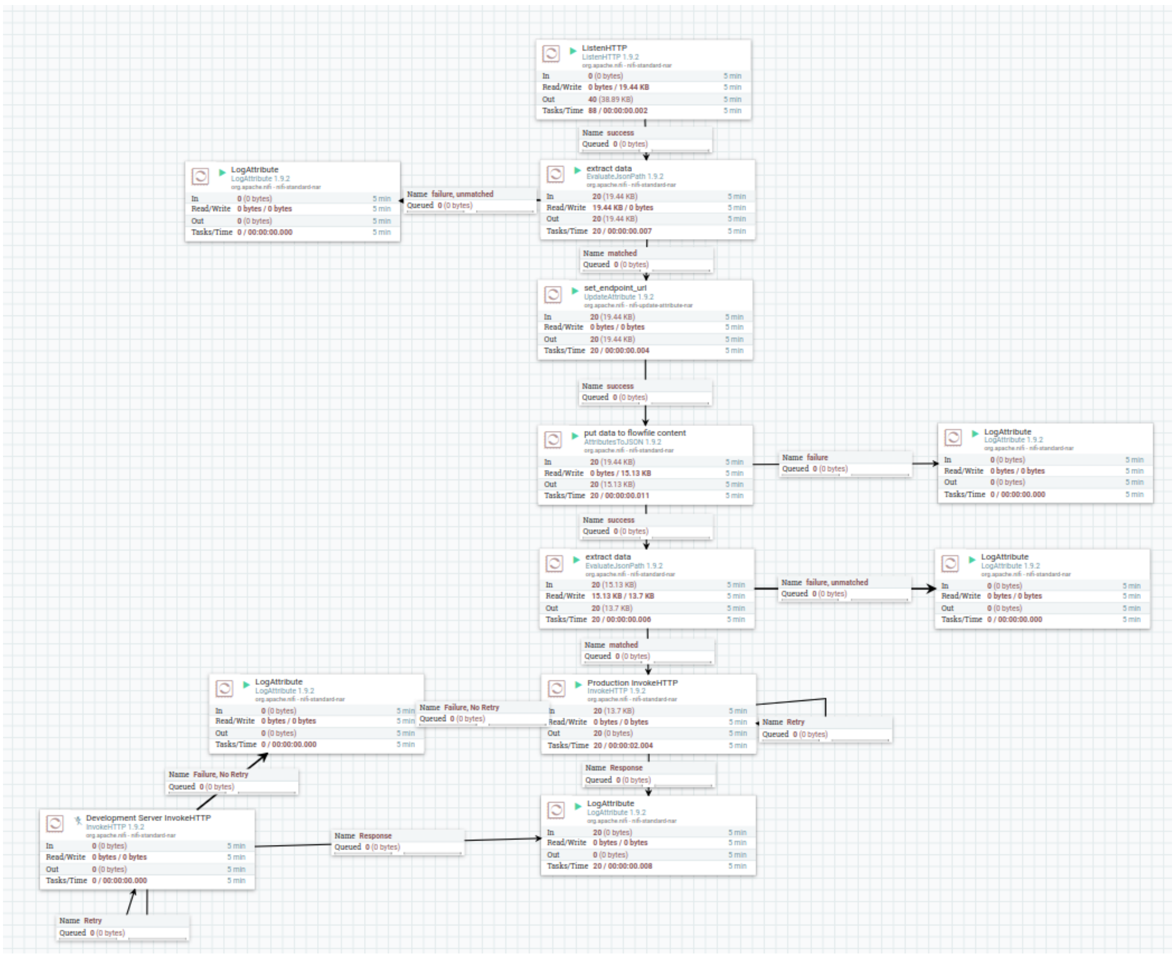
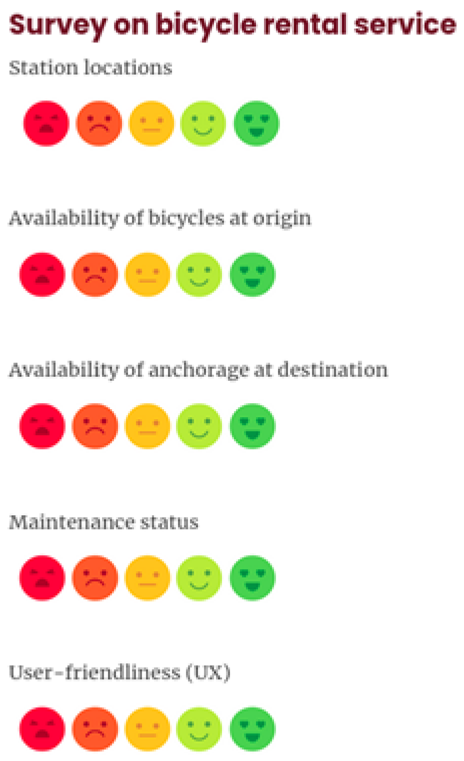


Figure 50 : Federation Validation.

- In addition, periodic queries to the Scorpio have been performed to assess that the data is being federated and to take actions in case of missing values. Figure 46 shows an example of a query using postman and the result obtained, showing that the federation is correct all the time in which the test was performed.

7.2.2 Qualitative Validation

Regarding subjective validation, several surveys has been developed to assess the user experience. In Figure 51, it shown an example for the bike's availability forecasting.


Survey on bicycle rental service

Station locations

Availability of bicycles at origin

Availability of anchorage at destination

Maintenance status

User-friendliness (UX)

Figure 51 : Survey for qualitative validation

Table 10: Validation plan results for Murcia/Molina use case.

Validation test number	Result	Comments
1.1.1	Passed	
1.1.2	Partial	
1.1.3	Passed	
1.1.5	Passed	
1.1.6	Partial	
1.1.7	Passed	
1.1.8	Passed	
1.2.1	Passed	
1.2.2	Passed	

Validation test number	Result	Comments
1.3.3.1	Passed	
1.3.3.2	Partial	
1.3.4.1	Passed	
1.3.4.2	Passed	
1.3.4.3	Passed	
1.3.5.1	Passed	
1.3.5.2	Passed	
1.3.6.1	Passed	
1.3.6.2	Passed	
1.3.6.3	Passed	
1.3.7.1	Passed	
1.3.7.2	Passed	
1.3.8.1	Passed	
1.3.8.2	Passed	
1.3.8.3	Passed	
1.4.1.3	Passed	
1.4.1.4	Passed	
1.5.1	Passed	

7.3 Flanders

7.3.1 Functional Validation

In Flanders the number of bikes forecasted is compared to the number of bikes in real time. The objective for this use case was:

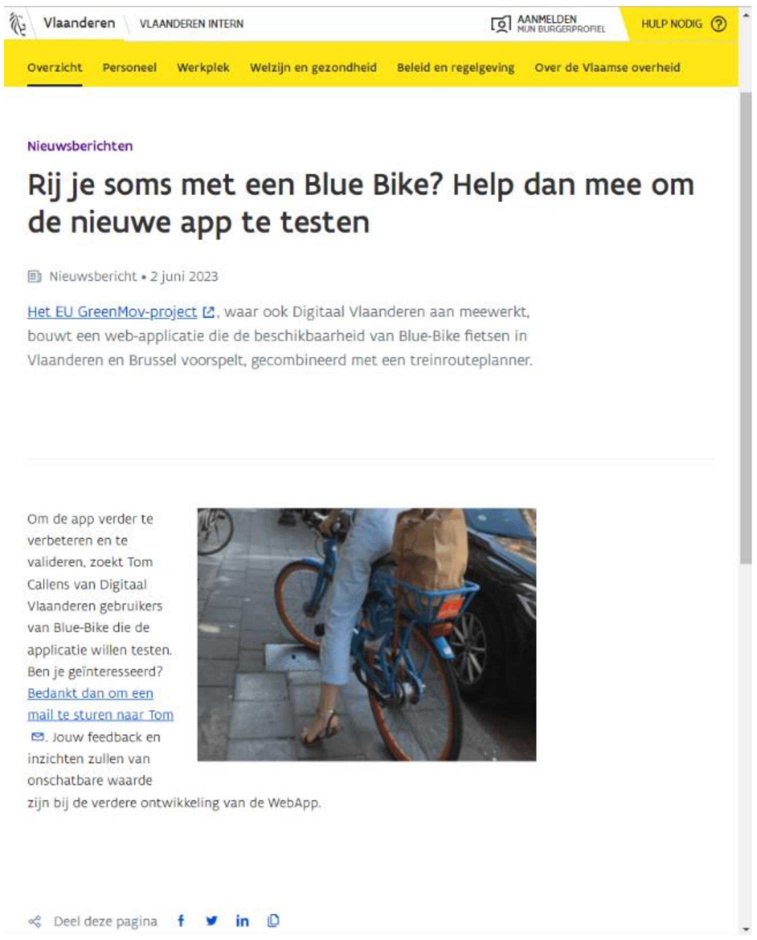
- Bikes Availability Forecast accuracy is 0.75 when comparing predicted number of bikes at (t + 15 min) with real value. Metric: R2.

During the training of the model, it was validated by checking the R2 values. During the validation a pragmatic real-life approach was used: a set of forecasts was triggered manually and the predicted values were stored in a table. At the time used in the forecast the actual values were write down and compared with the forecasts.

The KPI was to check on at least on 3 different days for 5 stations to obtain at least 20 samples with forecasting interval 15 min. Results are shown in the next sections.

7.3.2 Qualitative Validation

We contacted 10 Blue-bike users, showed them the app and asked them if they would like to see the forecasting service integrated in a mobility app and in which ones: Blue-bike, Google Maps, other ...etc. We then proposed them to fill a survey on their experience.



Nieuwsberichten

Rij je soms met een Blue Bike? Help dan mee om de nieuwe app te testen

Nieuwsbericht • 2 juni 2023

Het EU GreenMov-project, waar ook Digitaal Vlaanderen aan meewerkt, bouwt een web-applicatie die de beschikbaarheid van Blue-Bike fietsen in Vlaanderen en Brussel voorspelt, gecombineerd met een treinrouteplanner.

Om de app verder te verbeteren en te valideren, zoekt Tom Callens van Digitaal Vlaanderen gebruikers van Blue-Bike die de applicatie willen testen. Ben je geïnteresseerd? Bedankt dan om een mail te sturen naar Tom. Jouw feedback en inzichten zullen van onschatbare waarde zijn bij de verdere ontwikkeling van de WebApp.

Figure 52 : The call for Blue Bike users in Flanders.

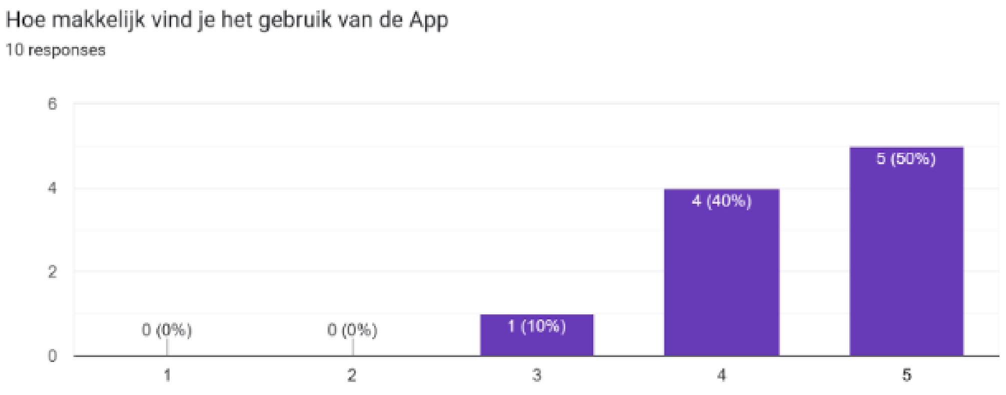


Figure 53 : Survey results example in Flanders.

9 out of 10 users found the application was easy to use (46).

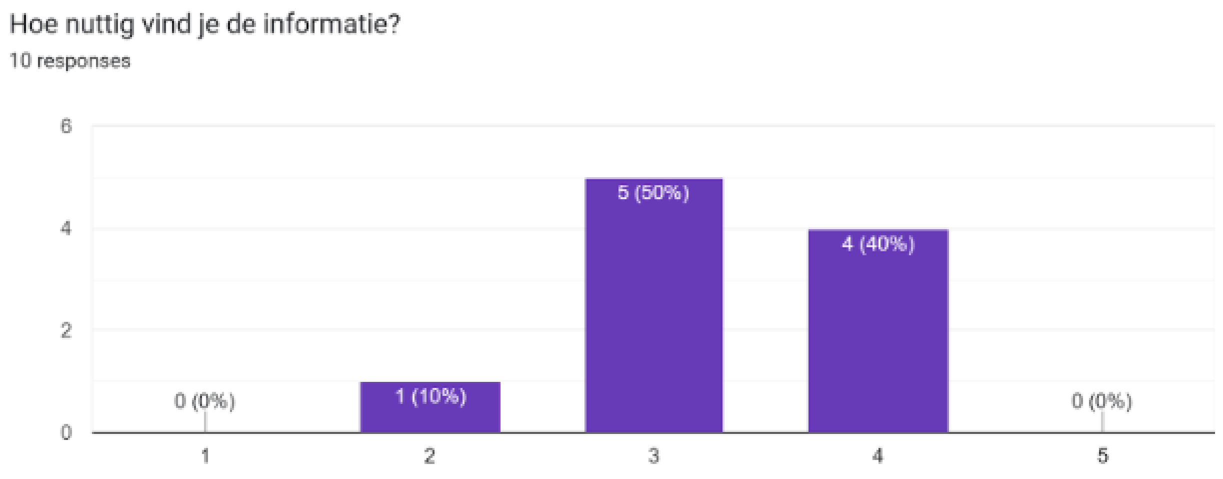


Figure 54 : Survey results example in Flanders (2).

As shown in Figure 54, 4 out of 9 users found the prediction information useful. 5 users were however undecided. This is partly due to some instabilities of the app suffered during the test, but also due to some limitations: the prediction seemed only reliable for the next 48 hours, and the search range (500 m) to find a nearby station was limited. The replies to the open-ended questions also contain explanations and suggestions: e.g. extend the app to other bike-sharing brands or services.

Solving those limitations goes however beyond the scope and aims of GreenMov, i.e. to show and evaluate if shared bike availability prediction information could be useful or not. Implementation improvements can be tackled when integrating the services within existing mobility applications, as indicated by the answers on the next and last question:

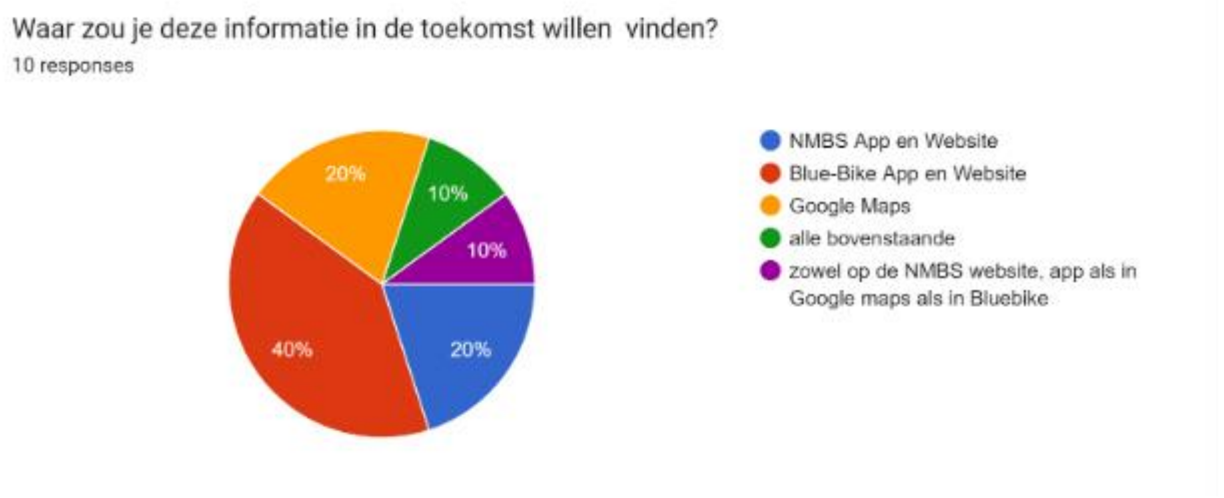


Figure 55 : Survey results example in Flanders (3).

When we asked where they would like to get such prediction information, we got these results (8):

- 6 users would like to see prediction information as part of a Blue-bike information (app, website)
- 4 users would like to see prediction information as part of Google maps.
- 4 users would like to see prediction information as part of National railway information.

Table 11: Validation plan results for Flanders use case.

Validation test number	Result	Comments
1.1.1	Passed	
1.1.2	Partial	
1.1.3	Passed	
1.1.5	Passed	
1.1.6	Passed	
1.1.7	Passed	
1.1.8	Passed	
1.2.1	Passed	
1.2.2	Passed	
1.3.8.1	Passed	
1.3.8.2	Passed	
1.3.8.3	Passed	
1.4.1.1	Passed	
1.4.1.2	Passed	
1.4.1.3	Passed	
1.4.1.4	Passed	Final version checked: response time below 1 sec from users point of view.

Validation test number	Result	Comments
1.5.1	Passed	

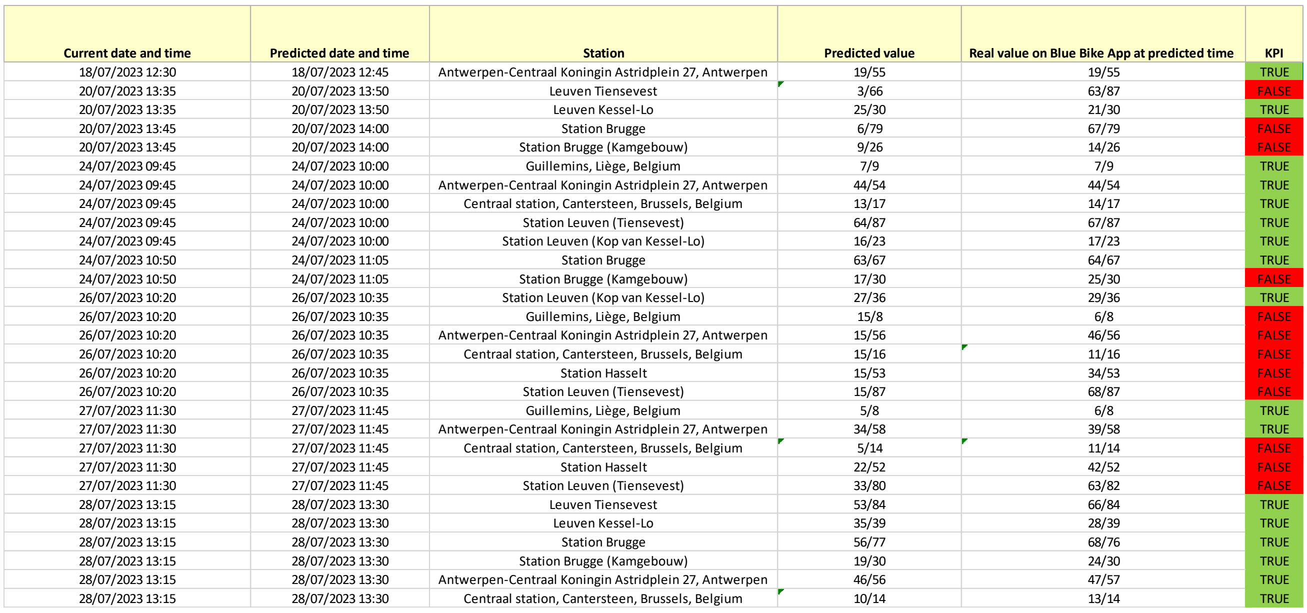
7.3.3 KPIs Validation

The KPI for the forecasting quality are calculated by using the method explained above in 7.3.1 and mostly relates to the accuracy of the service provided, i.e., forecast of the bike's availability at the location and time of arrival of the end-user.

As shown in the table below the accuracy at the time of writing varies from excellent to bad. This has been investigated and is related to delay in the real-time data collection in some cases.

With the available sample set the KPI is 59%: the real value is between 75 and 133% of the forecasted value.

When the test is run with correct real time data (i.e. the delay of the data provided to the model reflecting the current situation < 15 min), the KPI would be close to 99% for a forecast on the next 15 minutes. This has been consistently observed while testing, and is being investigated further.



Current date and time	Predicted date and time	Station	Predicted value	Real value on Blue Bike App at predicted time	KPI
18/07/2023 12:30	18/07/2023 12:45	Antwerpen-Centraal Koningin Astridplein 27, Antwerpen	19/55	19/55	TRUE
20/07/2023 13:35	20/07/2023 13:50	Leuven Tiensevest	3/66	63/87	FALSE
20/07/2023 13:35	20/07/2023 13:50	Leuven Kessel-Lo	25/30	21/30	TRUE
20/07/2023 13:45	20/07/2023 14:00	Station Brugge	6/79	67/79	FALSE
20/07/2023 13:45	20/07/2023 14:00	Station Brugge (Kamgebouw)	9/26	14/26	FALSE
24/07/2023 09:45	24/07/2023 10:00	Guillemins, Liège, Belgium	7/9	7/9	TRUE
24/07/2023 09:45	24/07/2023 10:00	Antwerpen-Centraal Koningin Astridplein 27, Antwerpen	44/54	44/54	TRUE
24/07/2023 09:45	24/07/2023 10:00	Centraal station, Cantersteen, Brussels, Belgium	13/17	14/17	TRUE
24/07/2023 09:45	24/07/2023 10:00	Station Leuven (Tiensevest)	64/87	67/87	TRUE
24/07/2023 09:45	24/07/2023 10:00	Station Leuven (Kop van Kessel-Lo)	16/23	17/23	TRUE
24/07/2023 10:50	24/07/2023 11:05	Station Brugge	63/67	64/67	TRUE
24/07/2023 10:50	24/07/2023 11:05	Station Brugge (Kamgebouw)	17/30	25/30	FALSE
26/07/2023 10:20	26/07/2023 10:35	Station Leuven (Kop van Kessel-Lo)	27/36	29/36	TRUE
26/07/2023 10:20	26/07/2023 10:35	Guillemins, Liège, Belgium	15/8	6/8	FALSE
26/07/2023 10:20	26/07/2023 10:35	Antwerpen-Centraal Koningin Astridplein 27, Antwerpen	15/56	46/56	FALSE
26/07/2023 10:20	26/07/2023 10:35	Centraal station, Cantersteen, Brussels, Belgium	15/16	11/16	FALSE
26/07/2023 10:20	26/07/2023 10:35	Station Hasselt	15/53	34/53	FALSE
26/07/2023 10:20	26/07/2023 10:35	Station Leuven (Tiensevest)	15/87	68/87	FALSE
27/07/2023 11:30	27/07/2023 11:45	Guillemins, Liège, Belgium	5/8	6/8	TRUE
27/07/2023 11:30	27/07/2023 11:45	Antwerpen-Centraal Koningin Astridplein 27, Antwerpen	34/58	39/58	TRUE
27/07/2023 11:30	27/07/2023 11:45	Centraal station, Cantersteen, Brussels, Belgium	5/14	11/14	FALSE
27/07/2023 11:30	27/07/2023 11:45	Station Hasselt	22/52	42/52	FALSE
27/07/2023 11:30	27/07/2023 11:45	Station Leuven (Tiensevest)	33/80	63/82	FALSE
28/07/2023 13:15	28/07/2023 13:30	Leuven Tiensevest	53/84	66/84	TRUE
28/07/2023 13:15	28/07/2023 13:30	Leuven Kessel-Lo	35/39	28/39	TRUE
28/07/2023 13:15	28/07/2023 13:30	Station Brugge	56/77	68/76	TRUE
28/07/2023 13:15	28/07/2023 13:30	Station Brugge (Kamgebouw)	19/30	24/30	TRUE
28/07/2023 13:15	28/07/2023 13:30	Antwerpen-Centraal Koningin Astridplein 27, Antwerpen	46/56	47/57	TRUE
28/07/2023 13:15	28/07/2023 13:30	Centraal station, Cantersteen, Brussels, Belgium	10/14	13/14	TRUE

Figure 56 : Test results bike forecasting with 15 min interval.

8 Open Data Sets

8.1 Introduction

The GreenMov project resulted in several different types of datasets:

- Datasets of proprietary measurement data that should remain private, such as traffic density measurements.
- Datasets of non-proprietary measured data, that are sometimes already available in European open data portal.
- Datasets of processed data (such as forecasts for example) that can be published in the European open data portal although they do not represent measurements.

The second type of data is aimed to be released in open data portals such as the corresponding national open data portal, which then feeds the European open data portal.

To make the datasets available, there is an additional information to be added to the raw version of the data. These are the metadata. There is a standard about the metadata to be included in the open data portals, DCAT-AP.

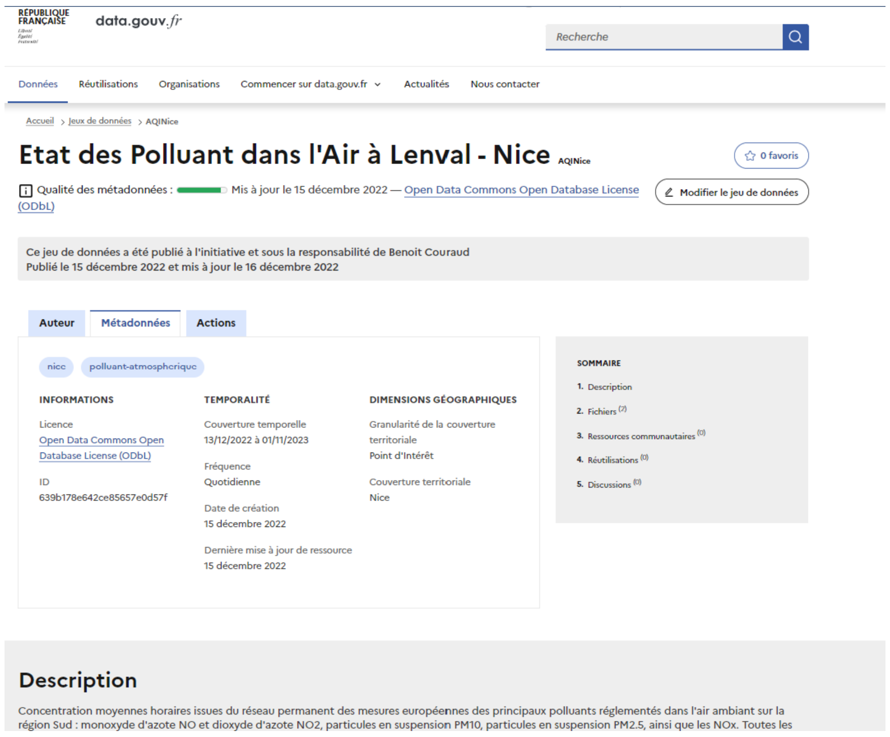
DCAT-AP is based on the DCAT standard and defines the elements of every dataset. The last version of this standard is available [17].

8.2 Results

Some datasets were published within GreenMov. The list of published open datasets is available below:

- Air Quality measurement in Nice : https://data.europa.eu/api/hub/repo/datasets/639b178e642ce85657e0d57f.jsonld?useNormalizedId=true&locale=en
- Bikes availability in Nice: https://data.europa.eu/data/datasets/64ebcf508e6ff7f89ef08dd0?locale=en
- Bikes availability in Flanders:
- Air Quality in Molina: https://datosabiertos.regiondemurcia.es/ayuntamiento-de-molina-de-segura/catalogo.html

These datasets have been exported to the European open data portal along with meta data.



Etat des Polluant dans l'Air à Lenval - Nice AQINice

Qualité des métadonnées : Mis à jour le 15 décembre 2022 — Open Data Commons Open Database License (ODbL)

Ce jeu de données a été publié à l'initiative et sous la responsabilité de Benoit Couraud
Publié le 15 décembre 2022 et mis à jour le 16 décembre 2022

nice polluant-atmospherique

INFORMATIONS
Licence: Open Data Commons Open Database License (ODbL)
ID: 639b178e642ce85657e0d57f

TEMPORALITÉ
Couverture temporelle: 13/12/2022 à 01/11/2023
Fréquence: Quotidienne
Date de création: 15 décembre 2022
Dernière mise à jour de ressource: 15 décembre 2022

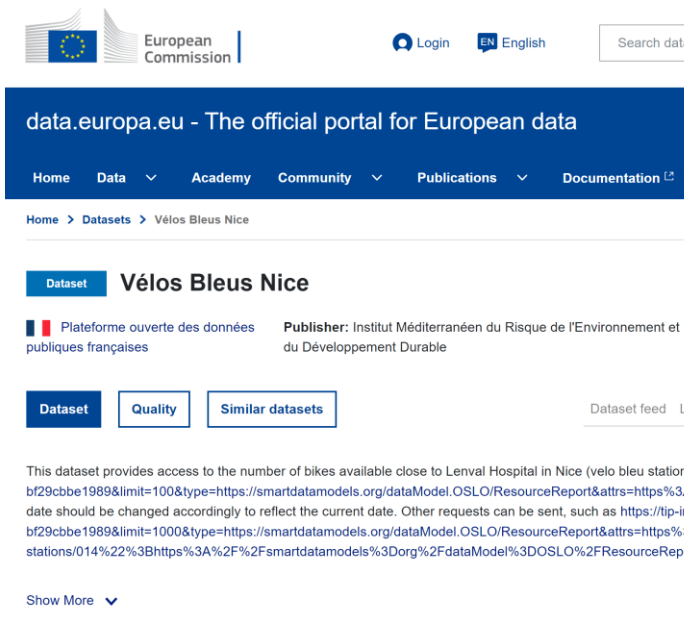
DIMENSIONS GÉOGRAPHIQUES
Granularité de la couverture territoriale: Point d'Intérêt
Couverture territoriale: Nice

SOMMAIRE
1. Description
2. Fichiers (2)
3. Ressources communautaires (0)
4. Réutilisations (0)
5. Discussions (0)

Description

Concentration moyennes horaires issues du réseau permanent des mesures européennes des principaux polluants réglementés dans l'air ambiant sur la région Sud : monoxyde d'azote NO et dioxyde d'azote NO2, particules en suspension PM10, particules en suspension PM2.5, ainsi que les NOx. Toutes les

Figure 57 : metadata validation tool at the French open data portal level.



data.europa.eu - The official portal for European data

Dataset Vélos Bleus Nice

Plateforme ouverte des données publiques françaises

Publisher: Institut Méditerranéen du Risque de l'Environnement et du Développement Durable

This dataset provides access to the number of bikes available close to Lenval Hospital in Nice (velo bleu statior
bf29cbbe1989&limit=100&type=https://smartdatamodels.org/dataModel.OSLO/ResourceReport&attrs=https%3.
date should be changed accordingly to reflect the current date. Other requests can be sent, such as https://tip-i
bf29cbbe1989&limit=1000&type=https://smartdatamodels.org/dataModel.OSLO/ResourceReport&attrs=https%
stations/014%22%3Bhttps%3A%2F%2Fsmartdatamodels%3Dorg%2FdataModel%3DOSLO%2FResourceRep

Figure 58 : Result of the automatic update of the European open data portal

9 Services in the FIWARE Marketplace

3 services from GreenMov have been published in the Fiware Marketplace. They correspond to the bike availability forecast service, the Noise annoyance forecast service, and finally the air quality index calculation service. A short description of each service is proposed below.

- Bike Availability Forecast Service: This service is encapsulated into a docker with an MLFlow image that includes the training and inference of the forecast of bikes available at a location. This service consists in a prediction model for bike-sharing systems using historical data to optimize fleet management and improve user experience.
- Air Quality Index Calculation Service: This service calculates Air Quality Index and Air Quality Level, based on the level of pollutants, as dictated by European normative. It receives the predictions from the Air Quality Forecasting according to the smart data model AirQualityForecast service and outputs an AirQualityForecast instance with the corresponding Air Quality Index and Air Quality Level. Thus, as it receives and produce an instance of a Smart Data Model, this service has been uploaded to the market place as a FIWARE ready software.
- Noise annoyance Forecast Service: This service is encapsulated into a docker that includes the training and inference of a python based machine learning model to forecasts the noise intensity and annoyance in the next hours. This service is available on request through a FastAPI component, but is also generating forecasts for every hour, and stores the result into the NoiseAnnoyanceForecast smart data model in a customizable context broker.
- Traffic Impact Service: This service calculates traffic emissions in CO_2 equivalent from IoT measurements and traffic intensity forecasts, allowing citizens and public authorities to easily understand the emissions amount at a specific location and timespan.
- Traffic Forecast Service: This service is also encapsulated into a docker that includes the training and inference of a python based machine learning model to forecasts the traffic intensity in the next days. This service is available on request through a FastAPI component, but is also generating forecasts for every hour, and stores the result into the TrafficEnvironmentImpactForecast smart data model in a customizable context broker.

10 Conclusions

This document has presented a comprehensive overview of the deployment steps and implementation details of GreenMov's green mobility services across its three use cases. By describing the software and infrastructure architectures, storage solutions, and services deployed in each location, it has provided valuable insights into the interdependencies and underlying technologies of these services.

The replication process has been well-documented, allowing readers to understand how to implement similar solutions in different contexts. The inclusion of validation plans for each use case ensures that the services and data models are rigorously tested and meet the project's objectives.

Overall, the successful deployment and validation of GreenMov's green mobility services in different settings demonstrates the project's potential to promote sustainable transportation solutions. By sharing the knowledge gained from these deployments, GreenMov aims to contribute to the advancement of green mobility technologies and encourage similar initiatives worldwide. Finally, partners of the GreenMov's project will most likely further refine this implementation of the project's use cases and will leverage these initiatives to help to make mobility in Flanders, Nice and Murcia/Molina greener and more sustainable.

11 References

[1] GreenMov, D3.2 Green Mobility Services Definition, https://green-mov.eu/sites/GreenMov /files/public/content-files

[2] GreenMov, D4.2 GreenMov Reference Architecture and guidelines v2, https://green-mov.eu/sites/GreenMov /files/public/content-files/

[3] GreenMov, D4.3 Source Selection Building Block, https://green-mov.eu/sites/GreenMov /files/public/content-files

[4] GreenMov, D5.1 Requirements for the data sets and mobility services, https://green-mov.eu/sites/GreenMov /files/public/content-files/2022/GreenMov %20-%20D5.1_Requirements%20for%20the%20data%20sets%20and%20mobility%20services_v1.0.pdf

[5] GreenMov, D2.1 Extended Smart Data Models v1, https://green-mov.eu/sites/GreenMov /files/public/content-files/2022/GreenMov %20-%20D2.1_Extended%20Smart%20Data%20Models_v1.0.pdf

[6] GreenMov, D2.2 Extended Smart Data Models v2.0, https://green-mov.eu/sites/GreenMov /files/public/content-files/2022/GreenMov %20-%20D2.2_Extended%20Smart%20Data%20Models_v1.0.pdf

[7] GreenMov, D3.1 Green Mobility Services Definition, https://green-mov.eu/sites/GreenMov /files/public/content-files/2022/GreenMov %20-%20D3.1_Green%20Mobility%20Services%20Definition_v1.0.pdf

[8] GreenMov, D4.1 GreenMov Reference Architecture and guidelines v1, https://green-mov.eu/sites/GreenMov /files/public/content-files/2022/GreenMov %20-%20D4.1%20GreenMov %20Reference%20Architecture%20and%20guidelines%20v1_v1.0.pdf

[9] Telefonicaid (no date) Telefonicaid/fiware-orion: Context broker and CEF Building Block for context data managemen, providing NGSI interfaces., GitHub. Available at: https://github.com/telefonicaid/fiware-orion (Accessed: January 10, 2023).

[10] Fiware (no date) FIWARE/context.orion-LD: Context broker and CEF Building Block for context data management which supports both the NGSI-LD and the NGSI-V2 apis, GitHub. Available at: https://github.com/FIWARE/context.Orion-LD (Accessed: January 10, 2023).

[11] ScorpioBroker (no date) Scorpiobroker/Scorpiobroker: NGSI-LD compliant context broker named Scorpio. developed by NEC Laboratories Europe and NEC Technologies India, GitHub. Available at: https://github.com/ScorpioBroker/ScorpioBroker (Accessed: January 10, 2023).

[12] Stellio-Hub (no date) Stellio-hub/stellio-context-broker: Stellio is an NGSI-LD compatible context broker, GitHub. Available at: https://github.com/stellio-hub/stellio-context-broker (Accessed: January 10, 2023).

[13] GreenMov, D2.3 A core vocabulary for shared mobility, https://green-mov.eu/sites/GreenMov /files/public/content-files/2022/GreenMov %20-%20D2.3_A%20core%20vocabulary%20for%20shared%20mobility%20v1.0.pdf

[14] Reference documentation (2023) Docker Documentation. Available at: https://docs.docker.com/reference/ (Accessed: February 27, 2023). GreenMov app (no date) GreenMov. Available at: https://bikeservice.GreenMov .odt.imec-apt.be/ (Accessed: February 27, 2023).

[15] ETSI GS QKD 014 v1.1 (no date). Available at: https://www.etsi.org/deliver/etsi_gs/QKD/001_099/014/01.01.01_60/gs_qkd014v010101p.pdf (Accessed: February 24, 2023).

[16] DCAT-AP (no date) Joinup. Available at: https://joinup.ec.europa.eu/collection/semantic-interoperability-community-semic/solution/dcat-application-profile-data-portals-europe/release/211 (Accessed: February 24, 2023).

[17] The Official Portal for European Data available at: https://data.europa.eu/api/hub/repo/datasets/639b178e642ce85657e0d57f.jsonld?useNormalizedId=true&amp;locale=en.

[18] GreenMov, D5.2 Pilot deployment and validations, https://green-mov.eu/sites/greenmov/files/public/content-documents/2023-04/GreenMov%20-%20D5.2%20Pilot%20deployment%20and%20validation_v1.0%20.pdf