Green mobility data models and services for smart ecosystems

D5.2 Pilot deployment and validation v1

Document Identification	
Contractual Delivery Date	28/02/2023
Actual Delivery Date	28/02/2023
Responsible Beneficiary	IMREDD
Contributing Beneficiaries	ATOS, HOPU, IMEC, FF, AIV
Dissemination Level	PU
Version	1
Total Number of Pages	65

Keywords
Smart Data Models, Services, Deployment, Validation Plan, GreenMov

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.

Document Information

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

List of Contributors	
Name	**Partner**
Benoit Couraud	IMREDD
Mehdi Nafkha	IMREDD
Eduardo Illueca Fernández	HOPU (Libelium Murcia)
Alberto Abella	FIWARE Foundation
Tom Callens	AIV (Digital Flanders)
Ismael Kutlu	IMEC
Zargham Kahn	IMEC
Filip Goselé	IMEC

Document History			
Version	**Date**	**Change editors**	**Changes**
0.1	24/10/2022	IMREDD	First Version created
0.2	13/01/2023	IMREDD	ToC validated; sections input in progress
0.4	14/02/2023	IMREDD	HOPU inputs
0.5	20/02/2023	IMEC/AIV/ FIWARE	Inputs from IMEC, AIV and FIWARE, first version for review
0.6	23/02/2023	ATOS	Review
0.9	27/02/2023	ATOS	Quality Review Form

Document History			
Version	**Date**	**Change editors**	**Changes**
1.0	28/02/2023	ATOS	FINAL VERSION TO BE SUBMITTED

Quality Control		
Role	**Who (Partner short name)**	**Approval Date**
Reviewer1	Miguel Aguilar (ATOS)	24/02/2023
Reviewer 2	Carmen Perea Escribano (ATOS)	24/02/2023
Quality manager	María Guadalupe Rodríguez (ATOS)	27/02/2023
Project Coordinator	Carmen Perea Escribano (ATOS)	28/02/2023

Table of Contents

List of Tables

List of Figures

List of Acronyms

Abbreviation / acronym	Description
API	Application Programming Interface
AQI	Air Quality Index
AQC	Air Quality Calculation
AQF	Air Quality Forecasting
AQ	Air Quality
CB	Context Broker
Dx.y	Deliverable number y belonging to Activity x
EC	European Commission
EGM	Easy Global Market
GBFS	General Bikeshare Feed Specification
GHG	Green House Gaz
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
LDES	Linked Data Event Stream
URL	Uniform Resource Locator
NCA	Nice Côte d'Azur
NGSI	Next Generation Service Interfaces
OSLO	Open Standards for Local Administrations

Abbreviation / acronym	Description
UX	User Experience

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 a first version of the KPIs 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 and a meta data validation tool that was used for validation of Milestone 14.

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 in order to test and validate these services but also 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, 3 and 4, 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 of 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]. The deliverables mentioned above propose the reference documents for the services and architectures, whereas this deliverable focusses on their implementation within the use cases.

Finally, as the deployment of the use cases is not final, the results presented are not considered as final neither.

Section II of the document includes a description of the use cases and the KPI's adopted. In Section III, we provide the general architecture along with use cases' specificities. In Section IV, we summarize the data models that are used in the use cases. Section V focuses on the services implementation, describing the process followed for services deployment. In section VI, we present the deployment process, including the technical but also the non-technical requirements, such as the involvement of stakeholders. Finally, section VI proposes a validation plan of the use cases implementation, as well as the first version of the Key Performance Indicators. Section VIII concludes this deliverable by describing the metadata validation tool that is used to validate the quality of the meta data associated with the datasets provided by GreenMov use cases, such as the ones used in Milestone 14.

1.1 Purpose of the document

This document proposes a description of the data models, services and GreenMov architecture deployment processes in the three pilots. It also includes the first feedback after the validation of the other Activities results.

1.2 Relation to other project work

This is the second 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. Similarly, 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:

- **Sections 2, 3, 4 and 5** present the three pilots, the architecture, the data models used and their services respectively.
- **Sections 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 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 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 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 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.

2 Use Cases

This section briefly presents the use cases adopted in the three cities Nice, Murcia/Molina and Flanders 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 public transport. This use case aims to:

- Support the decisions of urban transport managers (in particular by allocating more public transport or alternatives to 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 areas of work and expansion. 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 rent bike points to mitigate the impacts of traffic on air quality. To aim this objective, we will report the citizens the necessary information to decide to leave the car and search for green alternatives. We will also report to the public administration the knowledge to make 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 the green services for this use case. To delimit a bit, our pilot will focus 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 good combination of public transport.

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 all the 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

2.3 Flanders

2.3.1 Description

By providing forecasts of the availability of shared bikes, Flanders to improve the combined use of shared bikes and public transport. 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".

To enable citizens to make a data-based decision to use a shared bike in combination with their train trip an app will be developed. The app will ingest the Bike Forecasting services, the LDES-NGSI-LD bike availability service and the real time train schedule service.

Users will be asked questions 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.

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 will 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 in the core solutions that will allow data routing and storage. Indeed, the third layer consists in the context broker, that will ensure consistency of the data received with existing datamodels, and that will then store the data in relevant databases. These databases can either be real-time databases, such as MongoDB bases databases, or historical database, to store timeseries values, such as postgreSQL databases. APIs 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. Then, the deployment can use container technology based on docker, or Kubernetes, these technologies are defined in Section 1.4

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

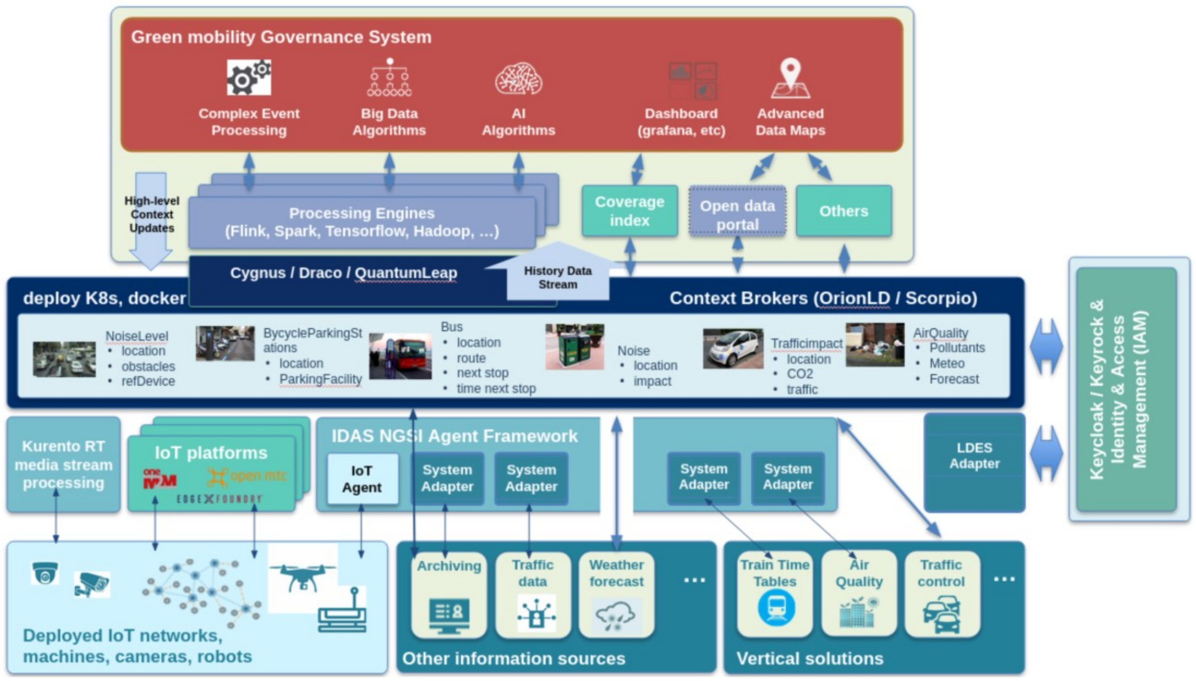


Figure 1 : Generic Architecture [3].

Finally, in parallel of these layers, we can find subservices such as securitisation of access to data, using Keycloack and Keyrock identity and access management, these technologies are defined in Section 1.4. Also, the addition of LDES adapter can help in the supply of time series data by speeding up the process of data retrieval.

Although one generic architecture was proposed within activity 4, the use cases defined in activity 5 have implemented their own specificities based on this generic infrastructure. 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

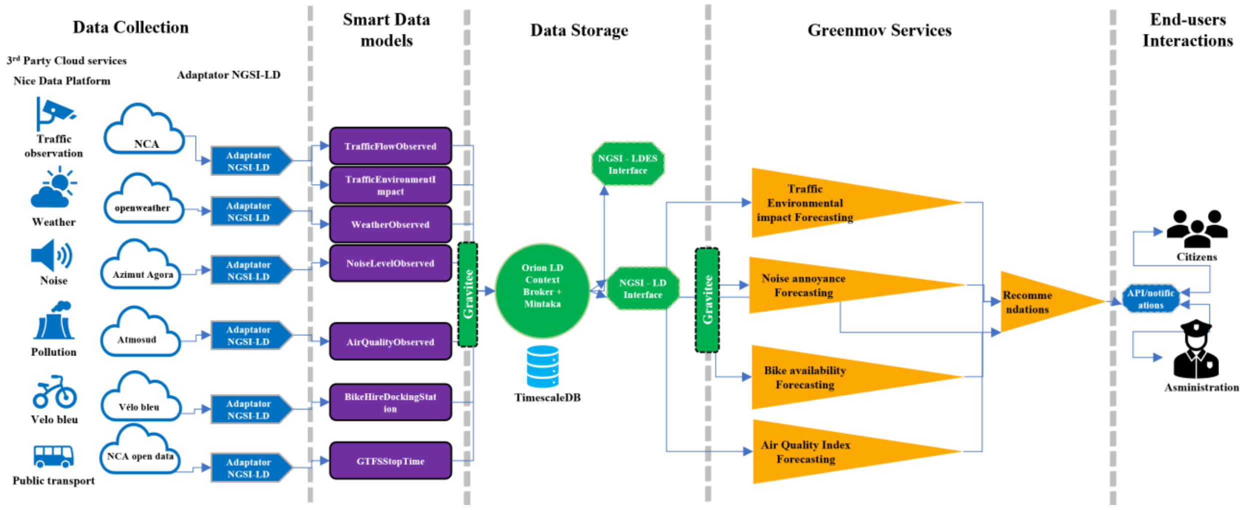


Figure 2 : Nice use case architecture.

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, but also the traffic monitoring. Once each of this data has been converted 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 (mintaka 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/airquality/temporal/entities/urn:ngsi-ld:AirQualityObserved:Nice-FR24035-Lenval?lastN=10&timeproperty=ts&api-key=afa51ec6-0014-46a2-ae52-5850116a348b instead of an address beginning by: https://orion-ld:1026/ngsi-ld/v1/.
- Also, it is worth noting that another interest 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 orientate 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 this section.

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 nodered 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.
- Public transport: a nodered 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 (in order to replace the standard Orion-LD URLs with custom URLs), and a secured access to data through API keys.

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

3.2.1.4 Services implementation

For the services implementation, many software components were designed and developed. In all of the cases, these services were implemented in dockerized Python images. 1describes the services implementation in more details, however, 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 can be done through kserve component, FastAPI or Flask. 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 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). 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.

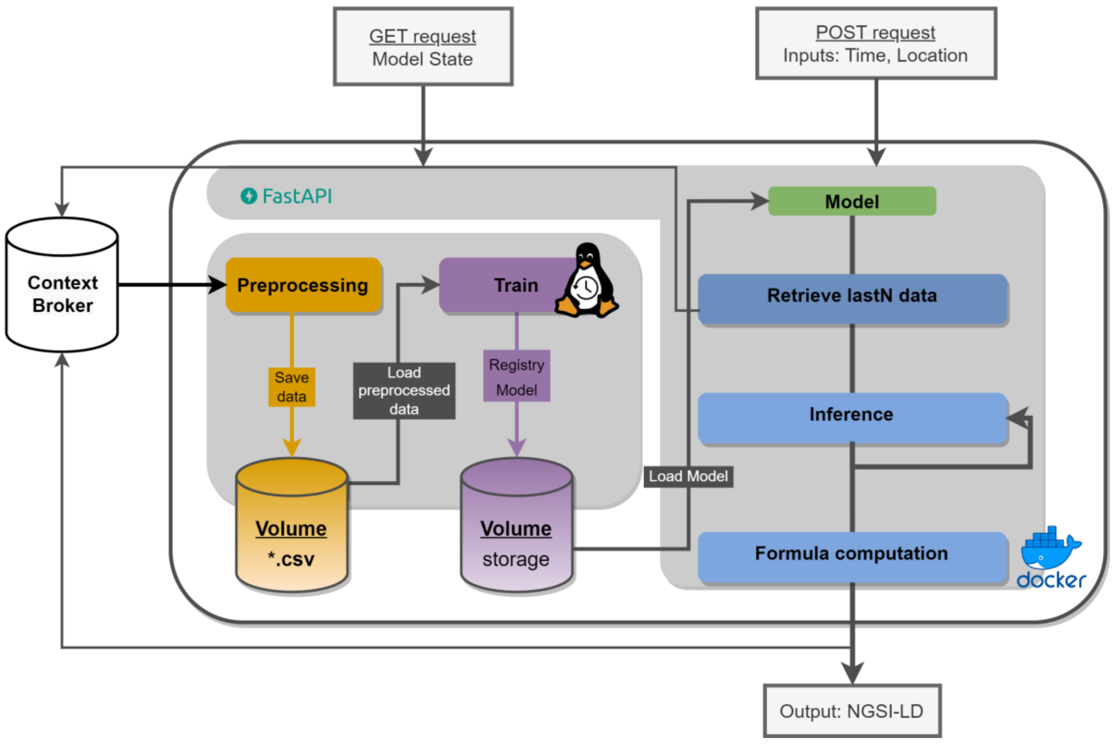


Figure 3 : Services Generic architecture [5].

3.2.1.5 Front end implementation

Finally, front end services in Nice do not apply to the end-users. Therefore, no interface was designed. However, services outputs such as traffic recommendations were sent out either as replies from a specific request, or by daily email to subscribed end-users. For the rest of the services, the API component is the only front end provided in the 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 building. This is depicted in Figure 4 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.

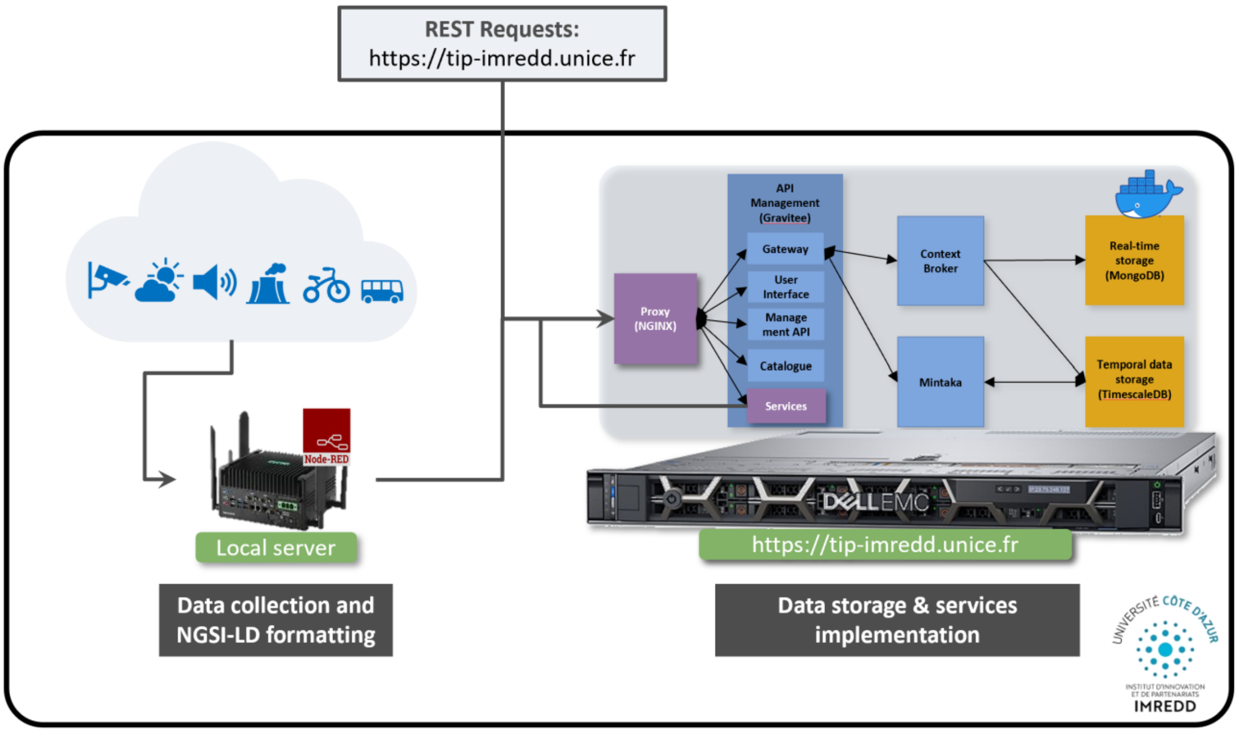

REST Requests:
https://tip-imredd.unice.fr
Node-RED
Local server
Data collection and NGSI-LD formatting
API Management (Gravitee)
Gateway
User Interface
Management API
Catalogue
Services
Proxy (NGINX)
Context Broker
Mintaka
Real-time storage (MongoDB)
Temporal data storage (TimescaleDB)
DELL EMC
https://tip-imredd.unice.fr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

Figure 4: Overall infrastructure for Nice Use Case.

3.2.2 Murcia/Molina

The architecture of Murcia/Molina follows the architecture proposed in Nice, and is detailed in Figure below:

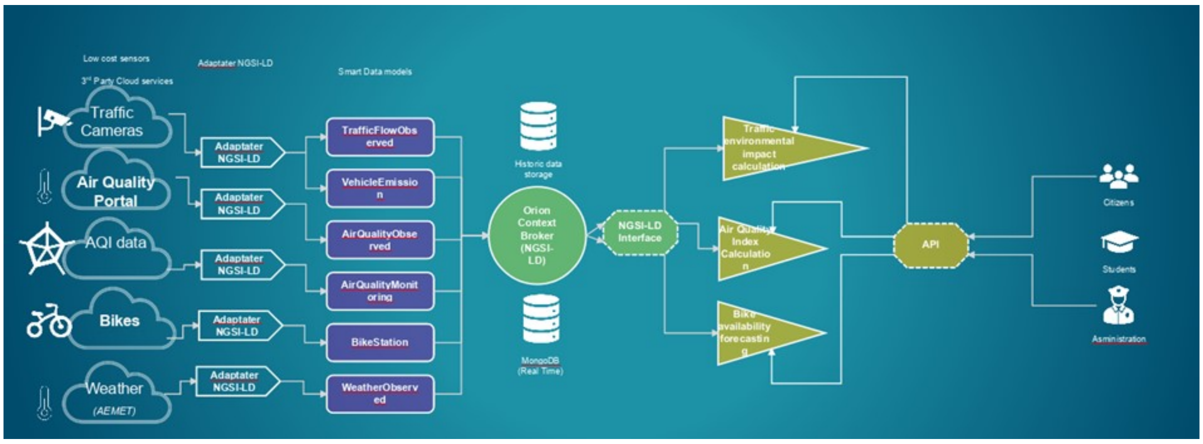


Figure 5: Architecture Murcia / Molina Use Case.

The first layer is called the data collection layer, including the adapters that will convert the data into NGSI-LD format. In the case of Murcia/Molina, several sensors are used, such as air quality monitoring, noise, bikes stations and traffic, once each of this data has been converted into the right data model, it is sent to the storage layer, with the context broker and the storage technology. In the case of Murcia, the broker is a federation of context brokers (Orion LD) to address the integration of both cities, as it is specified in Figure above. This federated structure will feed the green services, that can be accessed via API by the different actors and stakeholders.

3.2.2.1 Software

The software integrated in the architecture of this use case include all the green services that will be deploy, 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 will implement:

- 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 will be deploy, namely Air Quality Index Calculation, Air Quality Forecasting, Bike Availability Forecasting and Noise Annoyance Forecasting, described in detail in *Chapter 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, many software components were designed and developed. In all of the cases, these services were implemented in dockerized Python images, using Flask. 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 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 will be hosted by HOPU in an Azure machine.

3.2.3 Flanders

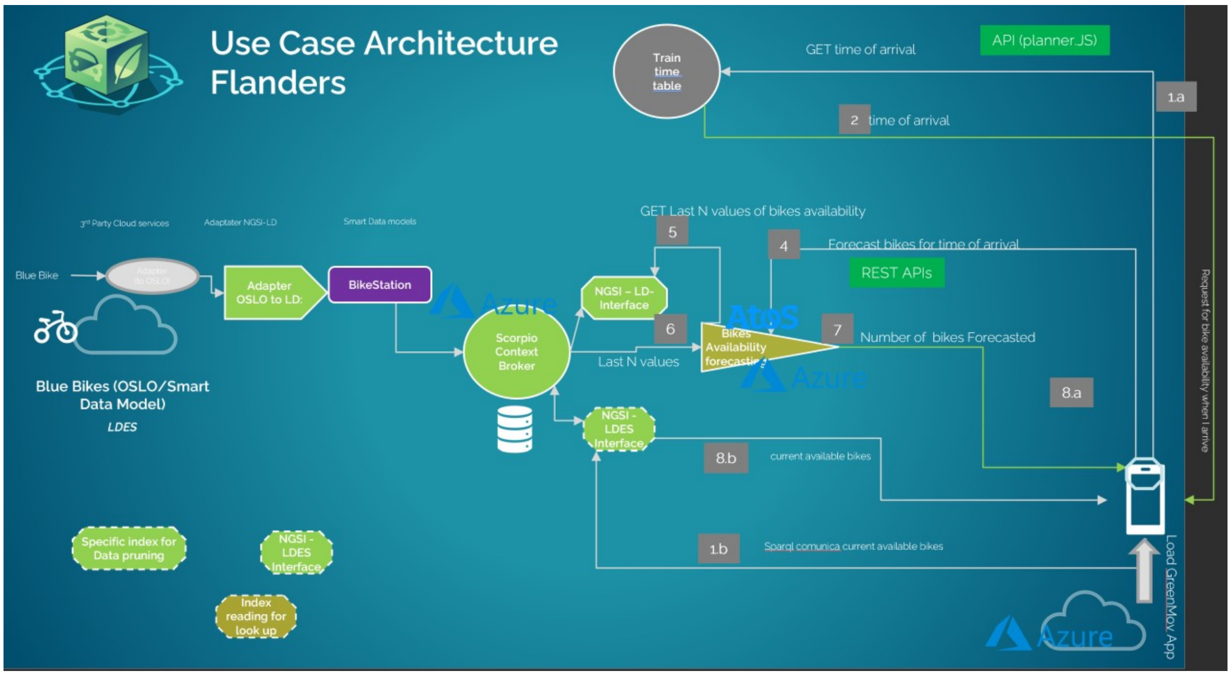


Figure 6 : Architecture of Flanders' Use Case.

3.2.3.1 Software

The software integrated in the Flanders architecture use case include LDES replicator (LDES to NGSI-LD), Redis, Scorpio Context Broker which is deployed in a distributed way & it consists of the following components: At context server, Config server, Entity Manager, Eureka server, Gateway, History Manager, Query Manager, Registry Manager, Registry Subscription Manager and Subscription Manager. The Scorpio 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 NGSI-LDES component and 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, 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 faster and perform better. The ReactJS library provides a component-based approach to 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 architecture of the use case in Flanders leverages Azure servers to store the data gathered during the experimental phase. Figure 7 This enables high availability and scalability of the service.

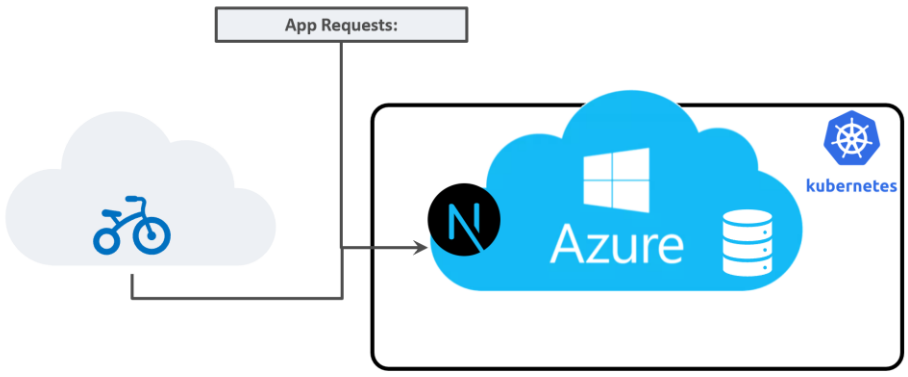


Figure 7 : Infrastructure in Flanders use case [1].

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	Basically, it is 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 :

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:

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 will be 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 (D5.1 [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, that 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: this smart data model was left out because its 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 in order to provide easy-readable information to users. A certain prediction of a given set of pollutants will be output by AQF service. Then, AQC service will process this prediction to get the corresponding air quality index. The whole process is related to a the 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 8.

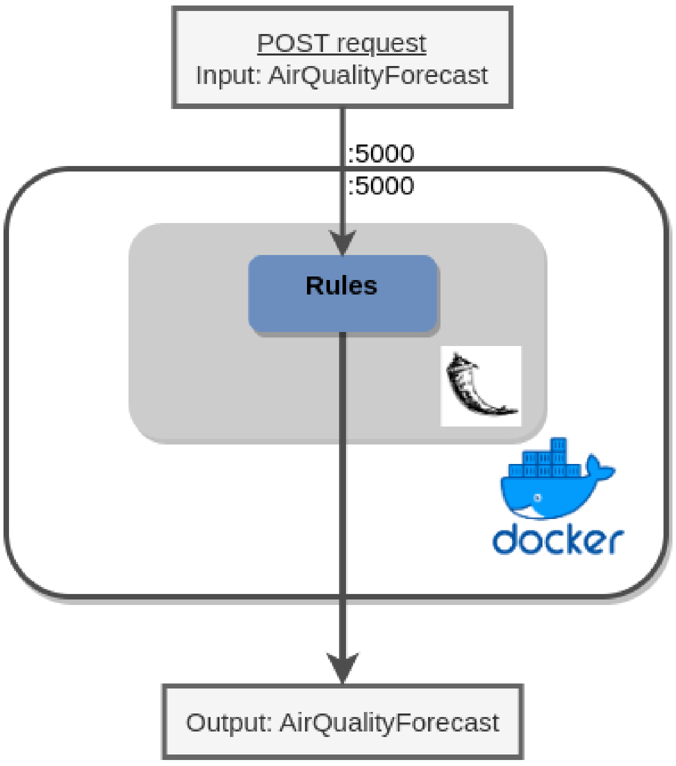


Figure 8 : Air Quality Index Calculation service architecture [1].

5.1.2 Air Quality Index Forecasting

As it is defined in D3.2 [1], this service provides an hourly based prediction of a given set of pollutants. Having received a request containing a timestamp and a localization, it will output the corresponding pollutant levels.

The architecture of the service is detailed in the Figure 9.

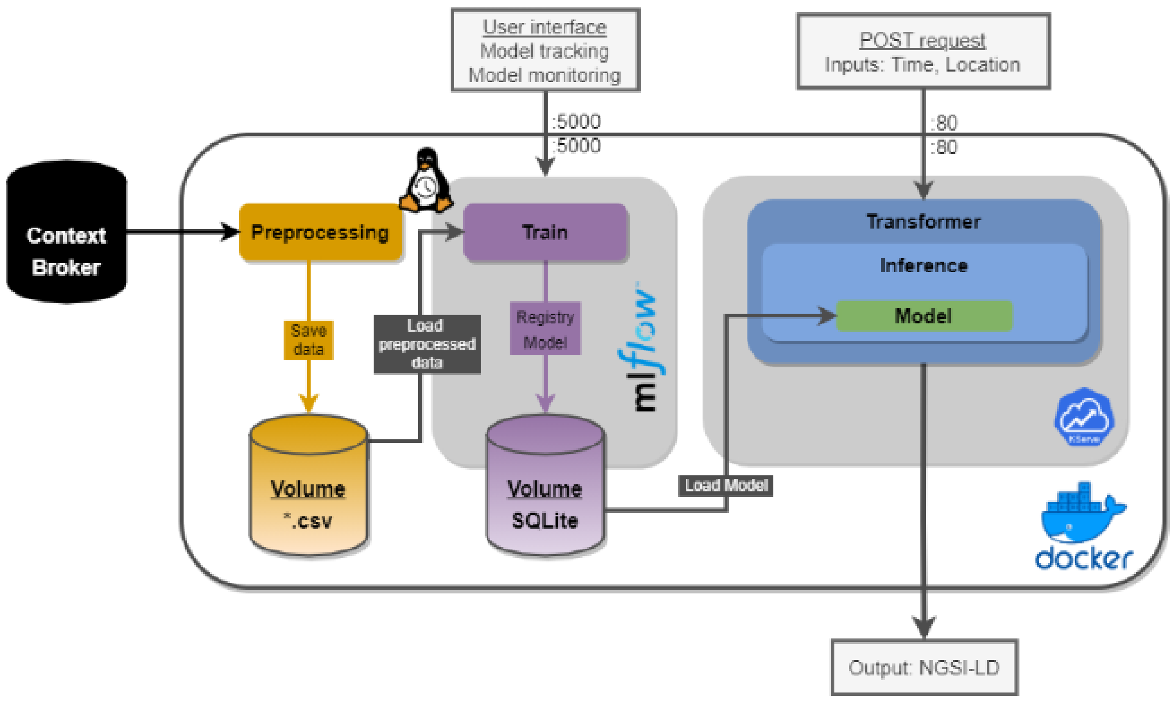


Figure 9 : Architecture AQF service [1].

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 will use a combination of IoT infrastructures and machine learning techniques to provide accurate and real-time forecasts of AQI levels within the city, the sensor of the AQ belonging to the south of France air quality observatory ATMOSUD is located inside the use case area as seen in the Figure below.

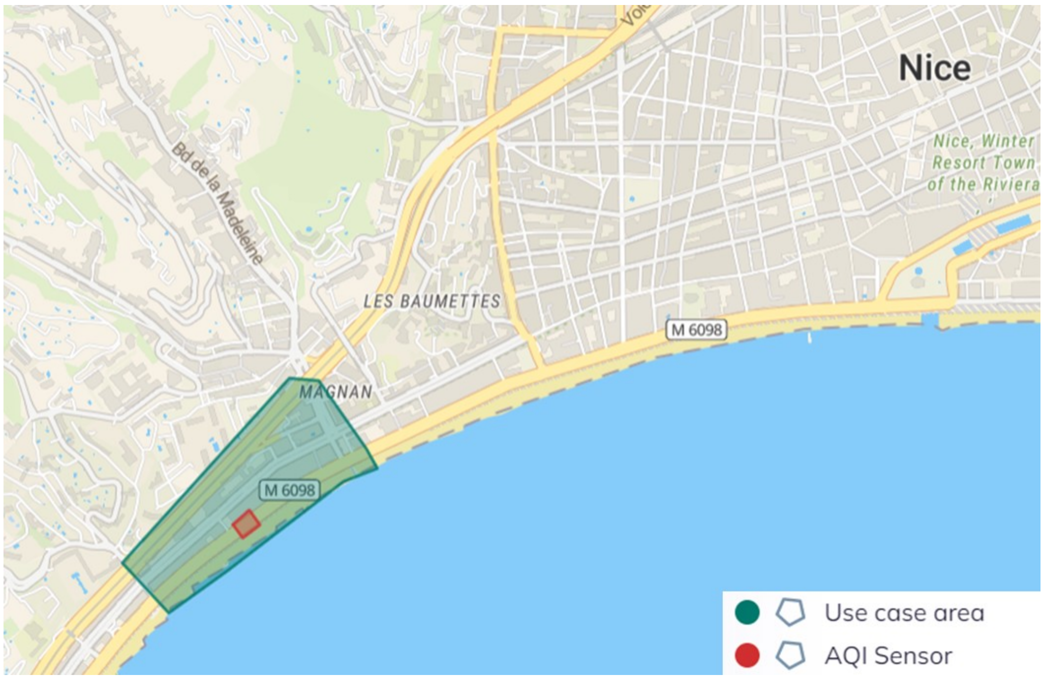


Figure 10 : Nice use case AQI sensor.

The service will make use of 3rd party cloud services and database as well to access and process historical AQI data for the past 2 years, up to the first trimester of 2022.

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.

5.1.3.2 Murcia/Molina use case Specificity

As well as in Nice, The Air Quality Index (AQI) forecasting service will use 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 also it will be applied to the air quality station in San Basilio. The Figure below shows the location of the Air Quality Devices in Molina de Segura.

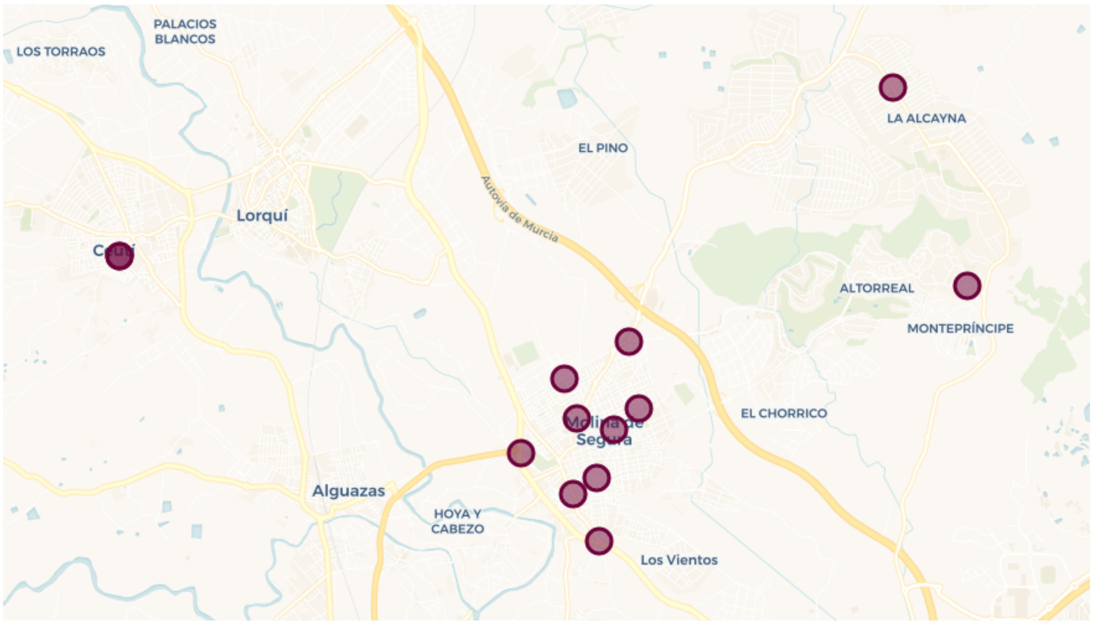


Figure 11 : Molina Air Quality Sensors.

5.1.3.3 Flanders use case Specificity

Air quality services are not used in 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 discomposed into several components, as depicted in Figure 12. First, on the left side, the data is collected by the use cases collection software solutions and is then formatted into 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 then a calculation component that computes the actual particles emissions from the categorised traffic intensity forecast.

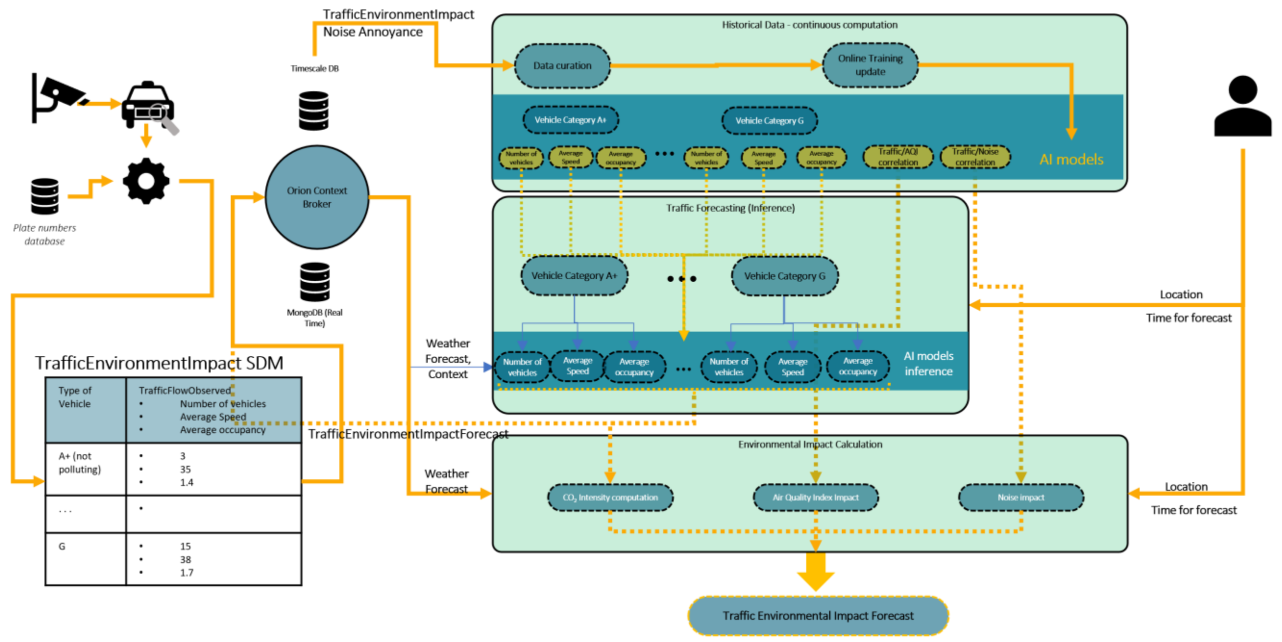


Figure 12 : Traffic environmental impact forecasting service [1].

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 equals 0 means the current traffic flow. This service includes an API subservice to provide forecasts on requests. According to Figure 13 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.

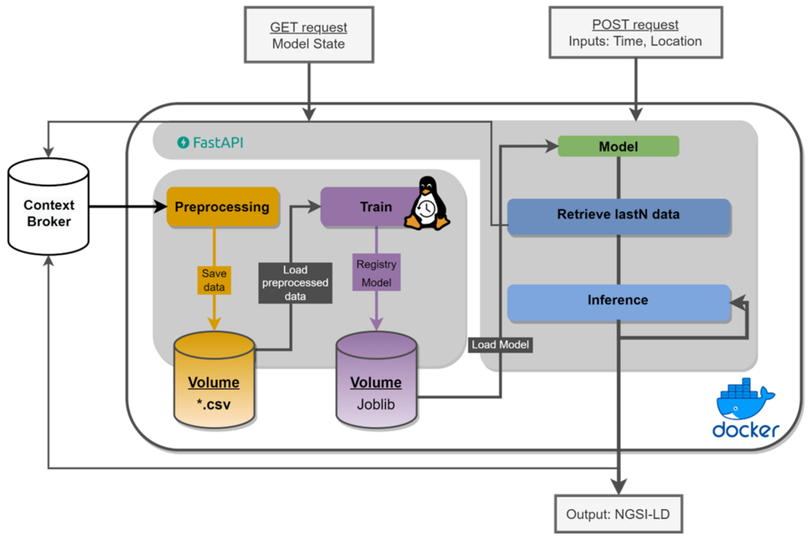


Figure 13 : Architecture of Traffic forecasting service [1].

5.2.2 Traffic Environmental Impact Calculation

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

This service is as well implemented through a docker image as seen in Figure 14:

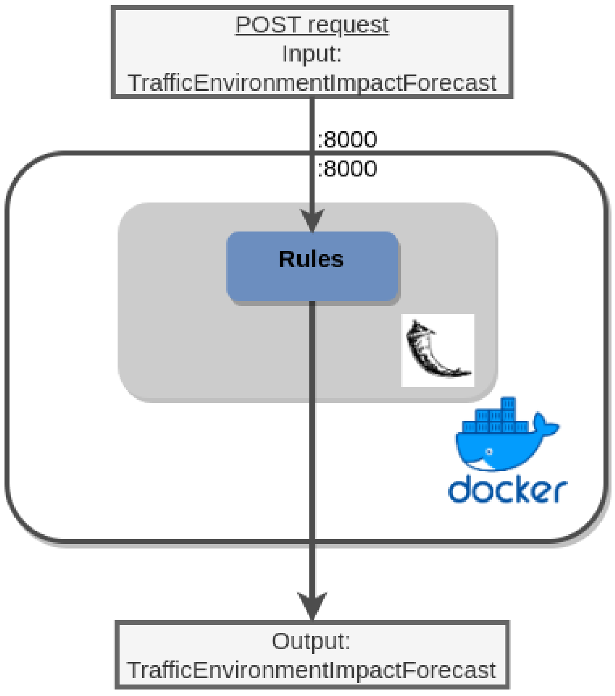

POST request
Input:
TrafficEnvironmentImpactForecast
:8000
:8000
Rules
docker
Output:
TrafficEnvironmentImpactForecast

Figure 14 : Traffic Environmental Impact Calculation service architecture [1].

5.2.3 Use Case characteristics

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, more over many tram, bus and bike stations "Velo Bleu" are situated in the same area which makes it the perfect spot to implement this service.

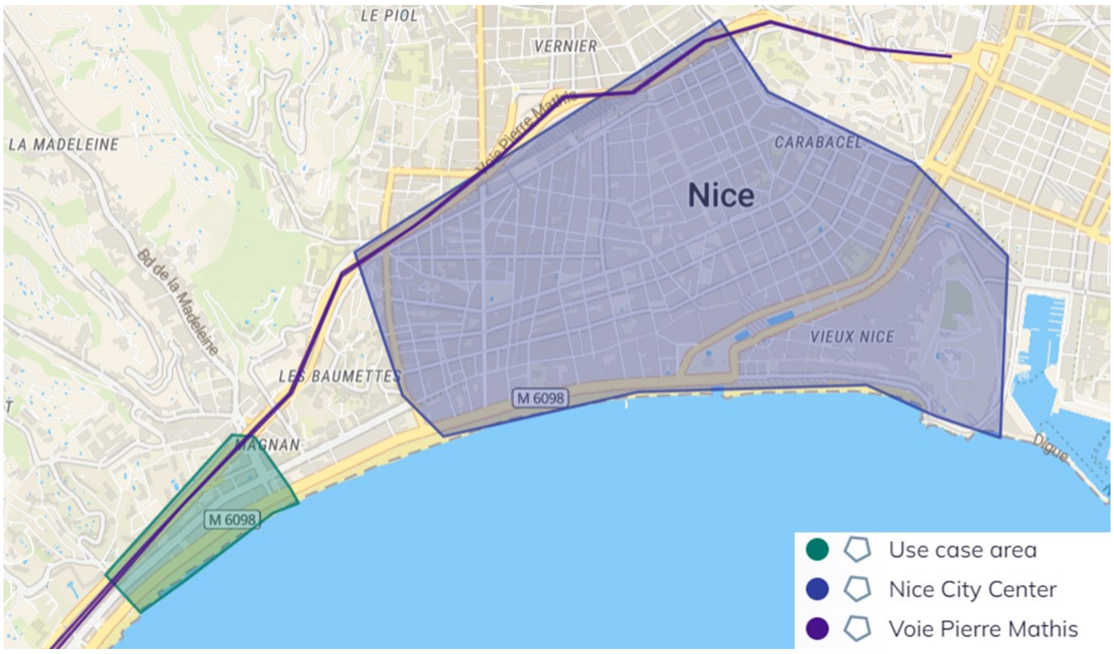


Figure 15 : Nice use case location.

5.2.3.2 Murcia/Molina use case Specificity

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

5.2.3.3 Flanders use case Specificity

Traffic environmental impact service is not involved in Flanders use case.

5.3 Traffic Recommendations Generation

The service will provide traffic recommendations at a specific location and time given by 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 sent to each hour to the project stakeholders to help make more informed decisions about how to travel.

5.3.1 Use Case characteristics

5.3.1.1 Nice Use case Specificity

The implementation of the service begins with processing the outputs of other services, such as noise pollution, air quality, and traffic 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 analyze the public transportation data and other forecasted datasets that will be 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 services. 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.

5.4 Bikes availability Forecasting Service

The several use cases require a knowledge of the availability of bikes in order 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 development 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 16.

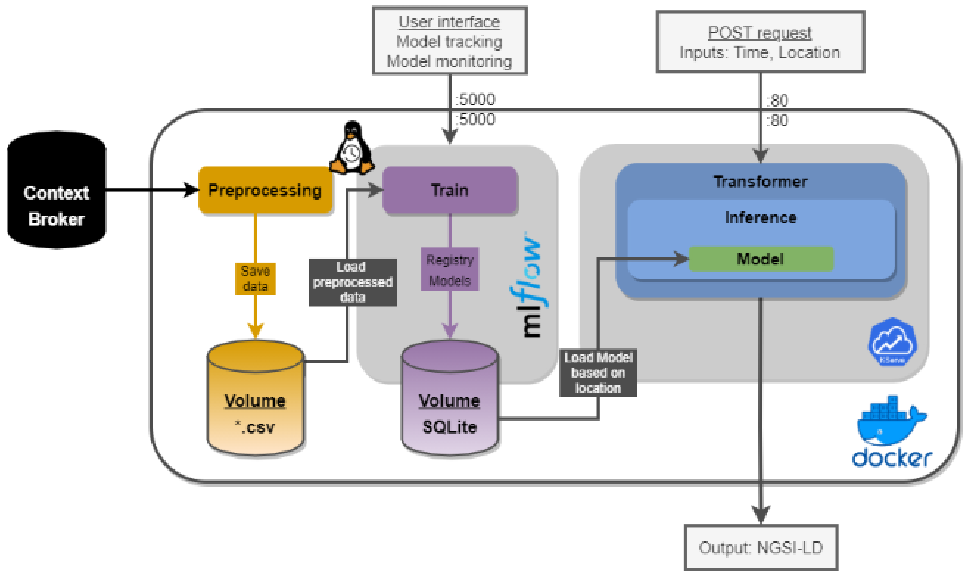


Figure 16 : Architecture Bikes availability forecasting service [1].

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 16 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, in order to make an inference of the bikes 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 the 30 minutes (for example. Can be lower time depending on the dataset 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, whereas another subservice called MLFlow runs in parallel to allow the service operator to monitor the ML components. Finally, 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 see 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.

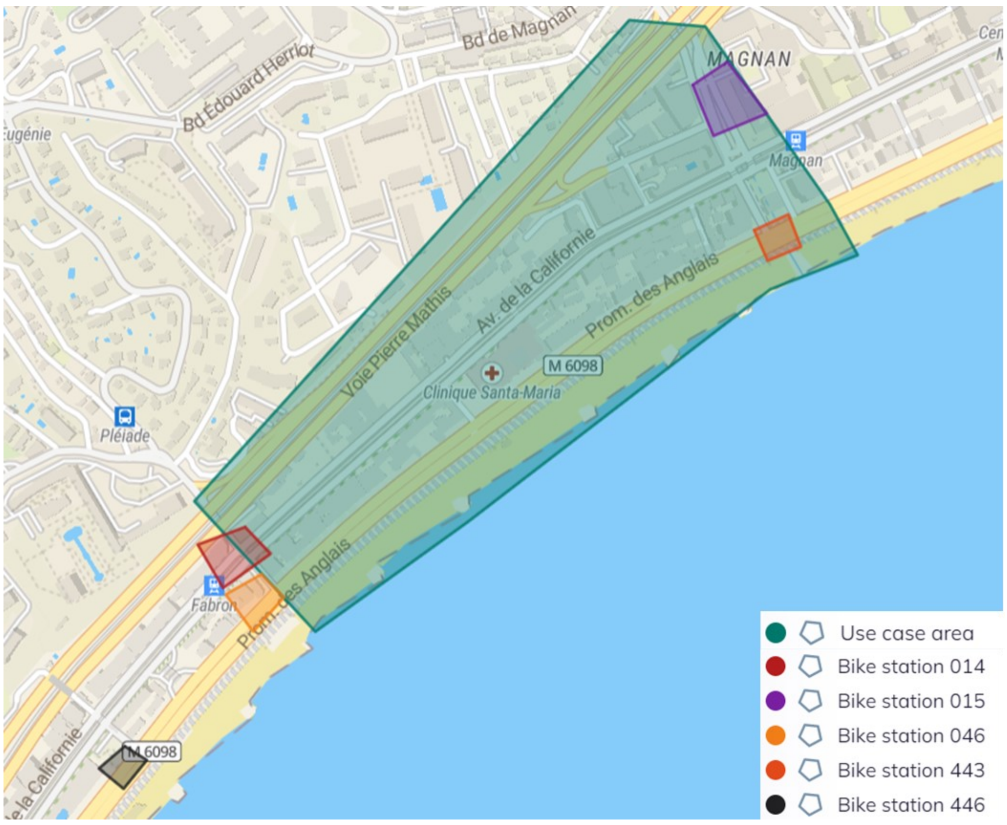


Figure 17 : Nice use case bike stations.

The implementation of the bike forecasting service in 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.

5.4.1.2 Murcia/Molina use case specificity

The *Bikes Availability Forecasting* will be implemented in all the stations in the city of Murcia, but the use case will focus specially on the stations in the area of the Campus of Espinardo, because of the interaction with the other services. The Figure below shows the location of all the stations that will be integrated, highlighting the stations in the area of Espinardo.

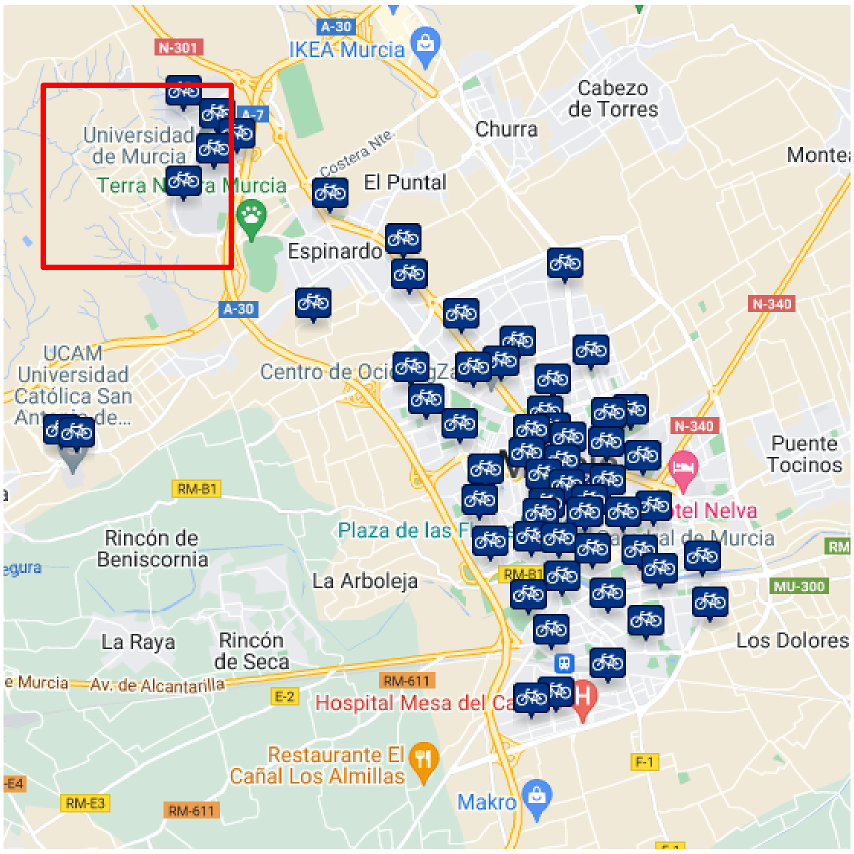


Figure 18 : Murcia Bike Stations.

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 train station.

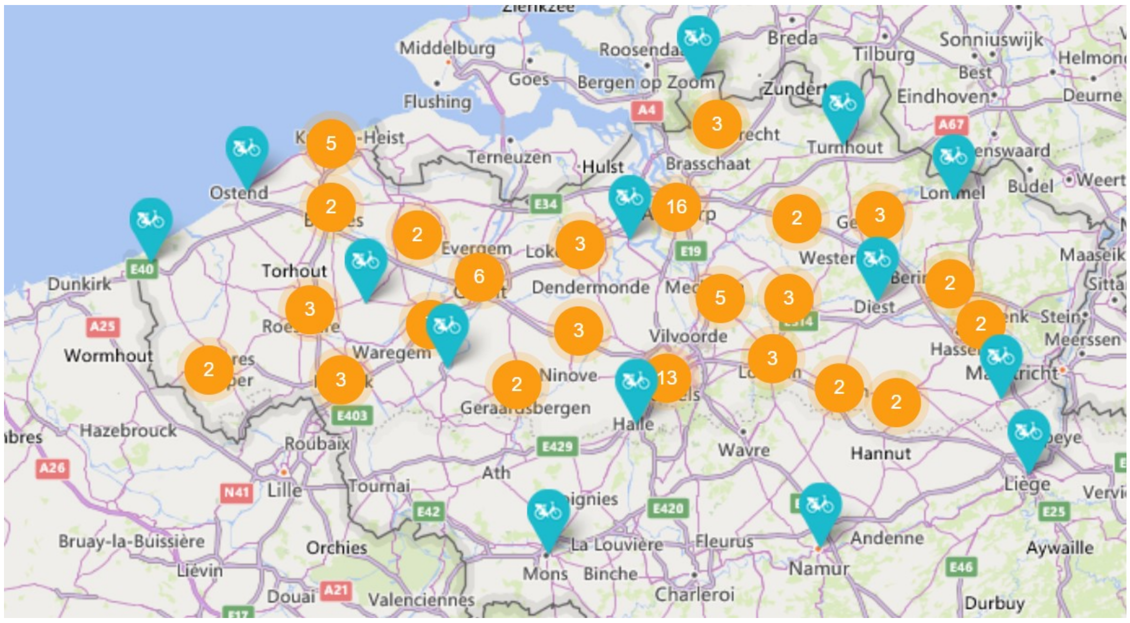


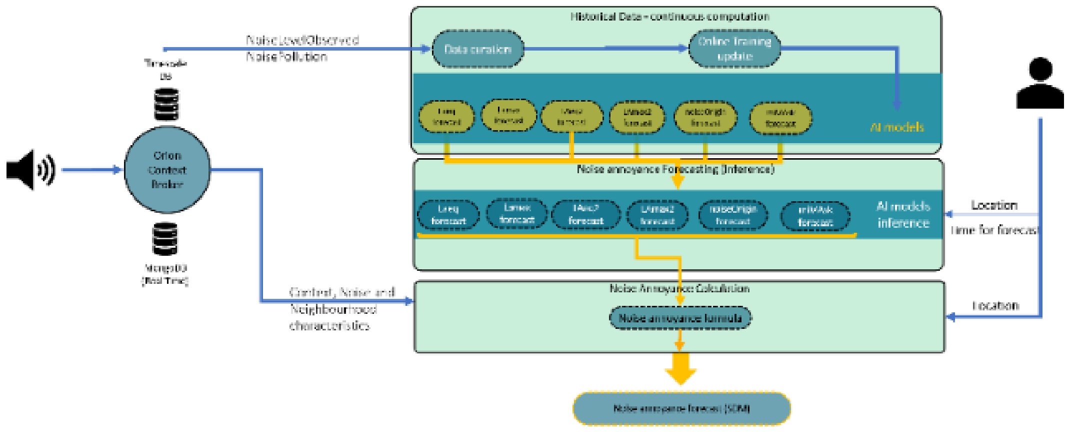
Figure 19 : Blue bike stations in Flanders/Belgium.

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.

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. 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.

The architecture of the service is detailed in Figure 20.



Historical Data - continuous computation
NoiseLevelObserved
NoisePollution
Data curation
Online Training update
AI models
Orion Context Broker
Noise annoyance Forecasting (Inference)
AI models inference
Location
Time for forecast
Noise Annoyance Calculation
Noise annoyance formula
Context, Noise and Neighbourhood characteristics
Location
Noise annoyance forecast (SDM)

Figure 20 : 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.

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.

5.5.2 Use case specificity

Describes the service implementation/usage in the use case

5.5.2.1 Nice use case specificity

Like many cities, Nice also faces an important challenge against noise pollution, for this matter 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 which 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 as shown in the Figure below a strategic location for our case given that it is situated between the two major traffic axis mentioned before, moreover bikes, bus and tram stations are situated nearby making the perfect spot to quantify the noise pollution for the use case.

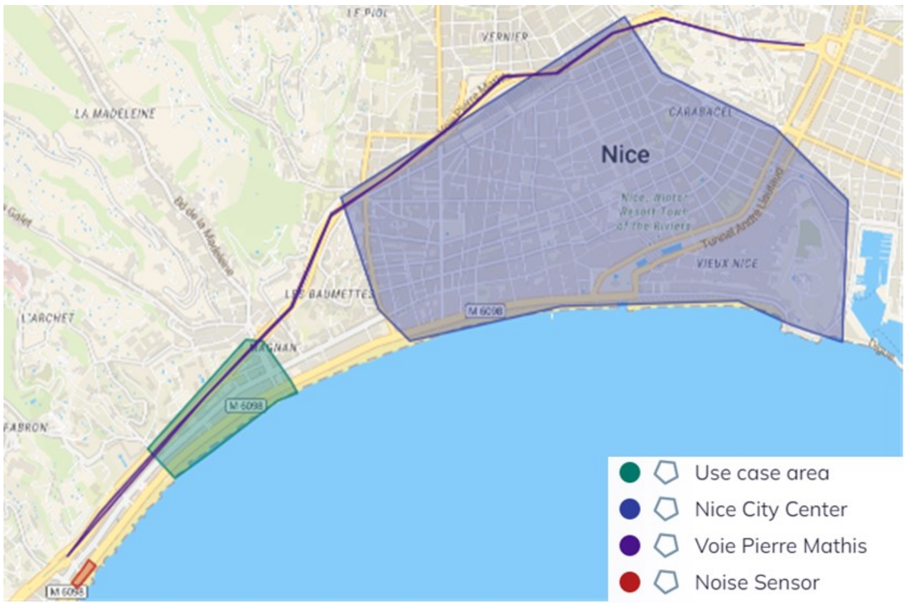


Figure 21 : Nice use case location with Noise Sensor.

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.

5.5.2.2 Murcia/Molina use case specificity

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

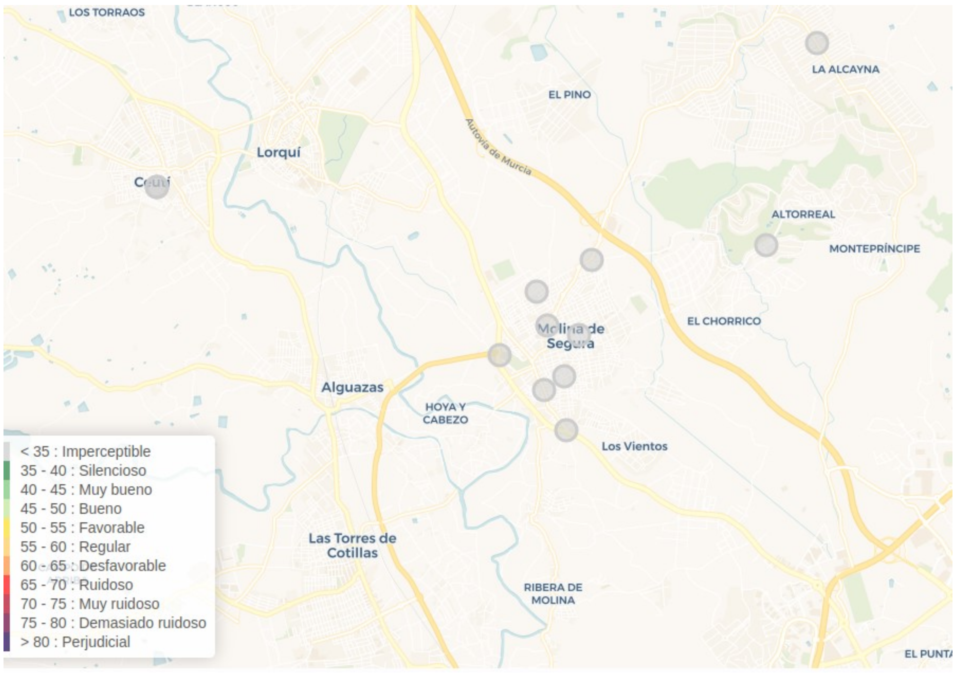


Figure 22 : Noise Sensors in Molina de Segura.

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 23 shows the main steps that were required during the deployment phase. The first main task to achieve for use case deployment consists in the data collection task. This task can be 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 will 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 has 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.

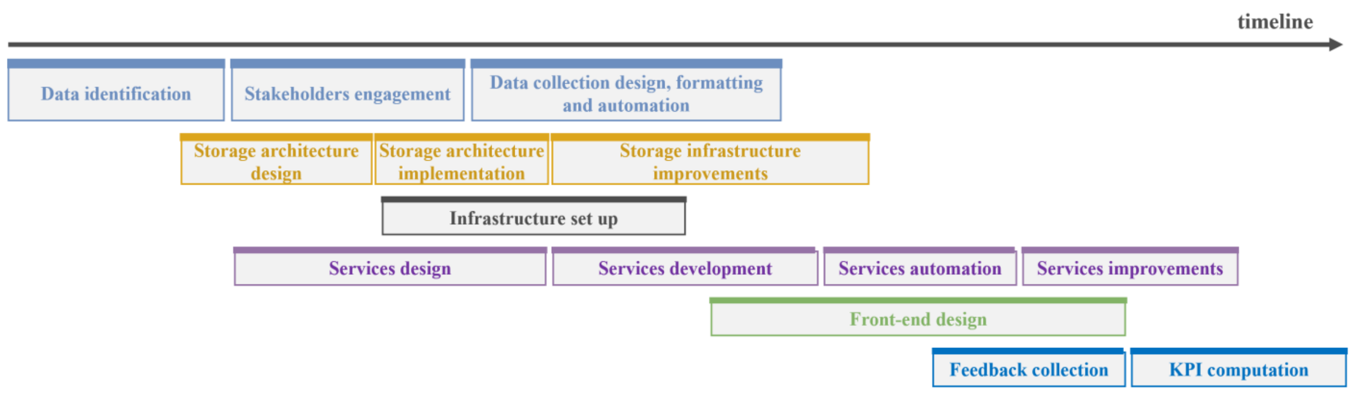


Figure 23 : Use cases deployment steps.

6.1 Nice

6.1.1 Deployment Steps

In the context of Nice use case, the steps proposed in Figure 23 were all reproduced except for the Front end design, which was a much lighter task than in other use cases, because communication with the end-users only consists in email notification to each subscribed end-users.

6.1.2 Technical Deployment

The technical deployment of the use case in Nice was done following the infrastructure description detailed in Figure 4. 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 24.

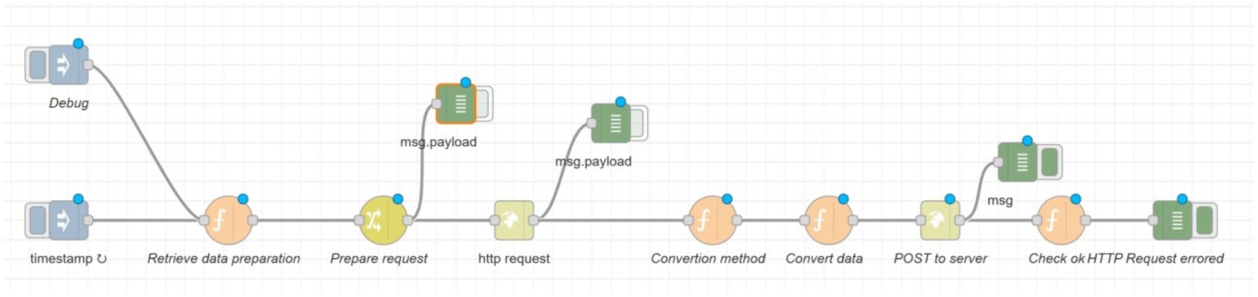
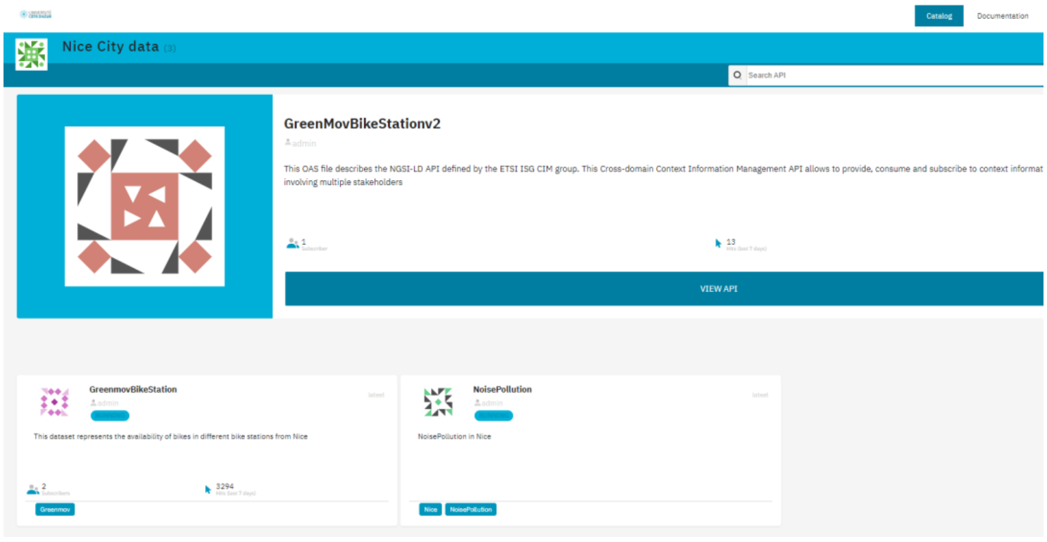


Figure 24 : Example of NodeRed flow implementation for air quality data collection in Nice.

- 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 went successfully, 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 Nice use case, an API manager was implemented along with a NGINX proxy in order to provide the following advantages:

- ○ An accessible catalogue of the dataset available, as shown in Figure 25.



Catalog Documentation

Nice City data (3)

Search API

GreenMovBikeStationv2

admin

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

1 Subscriber 13 Hits (last 7 days)

VIEW API

GreenmovBikeStation admin latest

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

2 Subscribers 3294 Hits (last 7 days)

Greenmov

NoisePollution admin latest

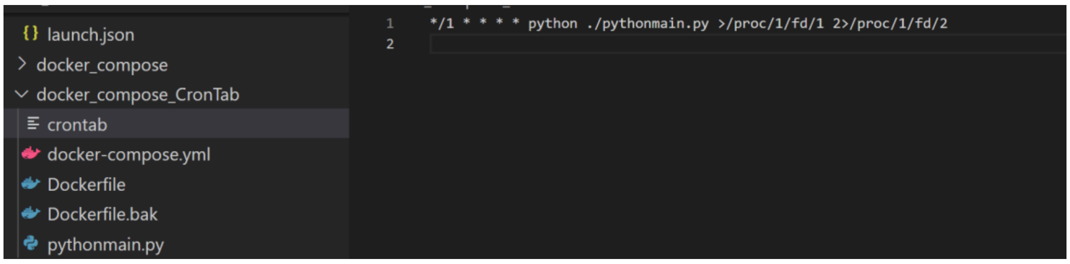
NoisePollution in Nice

Nice NoisePollution

Figure 25 : 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”.
- ○ A 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 in order 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 cron job was defined in docker in order 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 26.



```
{} launch.json
> docker_compose
v docker_compose_CronTab
  crontab
  docker-compose.yml
  Dockerfile
  Dockerfile.bak
  pythonmain.py
```

```
*/1 * * * * python ./pythonmain.py >/proc/1/fd/1 2>/proc/1/fd/2
```

Figure 26 : Zoom on a service folder using docker compose and crontab for tasks scheduling.

- 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 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. In the case of Nice, this engagement is still a work in progress as several datasets are not available through an API, but through a csv file, which is highly inefficient.
- 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, and by using it as follows:
    - https://tip-imredd.unice.fr/data/imredd/nice/bikestation/temporal/entities/urn:ngsi-ld:GreenMov -Nice-VeloBleu-BikeHireDockingStation-446?api-key=32a5519c-fde2-4df4-bcaf-bf29cbbe1989&timeproperty=ts&lastN=10.

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 23 are 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:** *Complete*
- 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:** *In progress.*
- Services deployment: all the services are working, but it is necessary to connect them to the final architecture to finalize the deployment. **Status:** *In progress.*
- Front-End implementation: **Status:** *Not started.*
- Use case validation: **Status:** *Not started.*

6.2.2 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 will be available in 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 23 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 7. The storage facility was implemented within Azure server in order to ensure high availability and scalability. Then, Kubernetes was used to deploy the docker containers. All the 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 due to a lack of knowledge and guidance in the documentation. Similarly, the front end is also hosted by Azure servers, for the same reasons. The development of the front end was faster due to the fact that it did not depend on any other expertise than the one that IMEC has. For better integration, the NEXT.js framework was used along with React.js for better end user experience. All the components are running as docker containers on our Azure Kubernetes Service (AKS). The deployment of all the components have been done using helm charts expect for Scorpio context broker, front-end client, LDES-replicator and NGSI-LDES, they consist of plain deployment files.

All the helm charts are open-source and are the docker containers are available in [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 frontend client are available in our imec GitHub repository.

6.3.3 Stakeholders and End-users engagement

For the use case, IMEC has been able to leverage its relationship with the organisation responsible for Blue Bikes, that publishes the data of bikes availability in linked data format, but not in NGSI. Then, relationships with the administration responsible for the train schedules allowed the team project to access train data for this use case. Finally, strong involvement with OSLO helped in achieving the right data models architecture.

The engagement also includes the relationship with end-users, that focused first on the GreeMov 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 GreeMov, 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 will also be made available via an online application available via a dedicated website [15], the front end of the application is as shown in Figure 27.

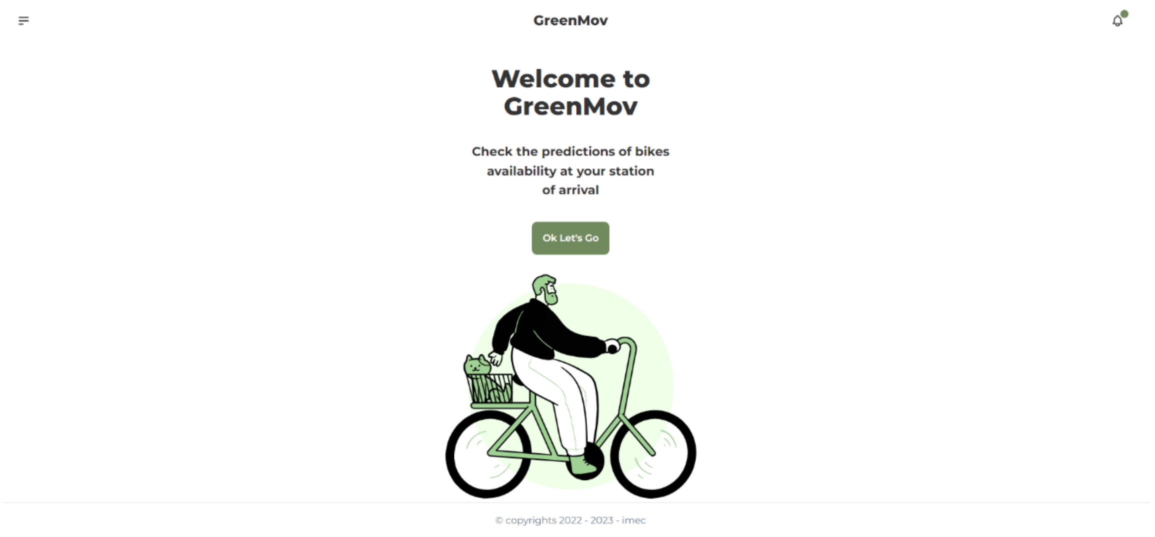
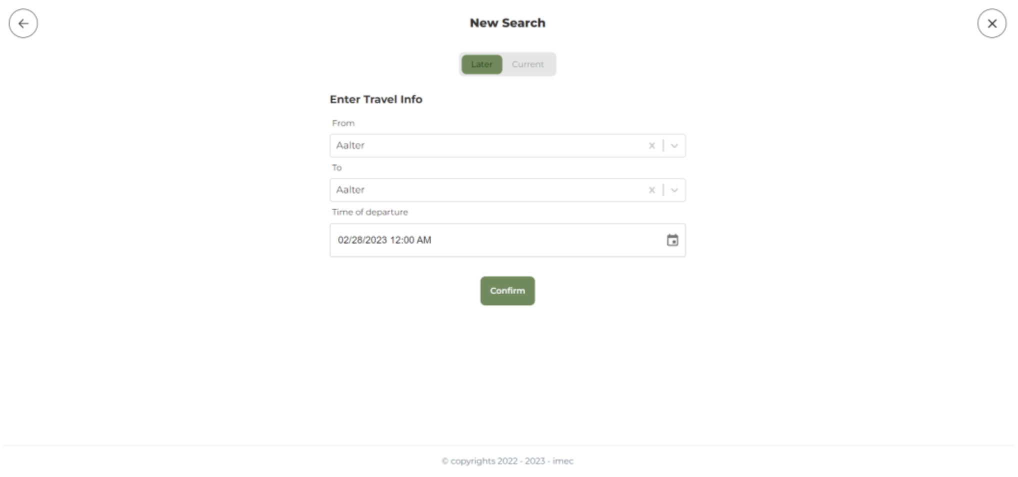

GreenMov
Welcome to GreenMov
Check the predictions of bikes availability at your station of arrival
Ok Let's Go
© copyrights 2022 - 2023 - imec

Figure 27 : Bikes availability application welcome page.

The search feature of the application is as shown in Figure 28.



Enter Travel Info

From: Aalter

To: Aalter

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

Figure 28 : 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 will deploy the helm chart on its Kubernetes platform and the software components will run.

7 Use Case Validation Plan

Each use case that implemented technical solutions have 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 timestamp 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 r^2 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.

Bike 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 r^2 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 r^2 values.

7.1 Nice

7.1.1 Functional Validation

In the context of 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 is subjective to the implementation of the services and is still in progress.

7.1.2 Qualitative Validation

In the case of Nice use case, end-users will only receive notifications by email or through the accessible API. A group of 20 end-users will be considered to test the suitability of recommendations sent by email or through the API.

Table 7: Validation plans intermediary 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	Partial	Passed for Airquality, Bike, but still require more frequent update of noise and traffic data
1.2.2	To be done	Validation tool to be developed
1.3.1.1	Passed	
1.3.1.2	Passed	
1.3.1.3	Passed	To be rechecked after final implementation
1.3.2.1	Passed	
1.3.2.2	Passed	To be rechecked after final implementation
1.3.3.1	To be done	Traffic recommendation service not finalised yet
1.3.3.2	To be done	Traffic recommendation service not finalised yet
1.3.4.1	Passed	
1.3.4.2	Passed	
1.3.4.3	Passed	To be rechecked after final implementation
1.3.5.1	Passed	

Validation test number	Result	Comments
1.3.5.2	Passed	To be rechecked after final implementation
1.3.6.1	Passed	
1.3.6.2	Passed	
1.3.6.3	Passed	To be rechecked after final implementation
1.3.7.1	Passed	
1.3.7.2	Passed	To be rechecked after final implementation
1.3.8.1	Passed	
1.3.8.2	Passed	
1.3.8.3	Passed	To be rechecked after final implementation
1.4.1.3	Partial	Requests reception have been validated, but not subscriptions yet.
1.4.1.4	Passed	To be recheck after final implementation
1.5.1	To be done	

7.1.3 KPIs Validation

The KPIs validation will be part of the next deliverable, once all the services are implemented and running.

7.2 Murcia/Molina

7.2.1 Functional Validation

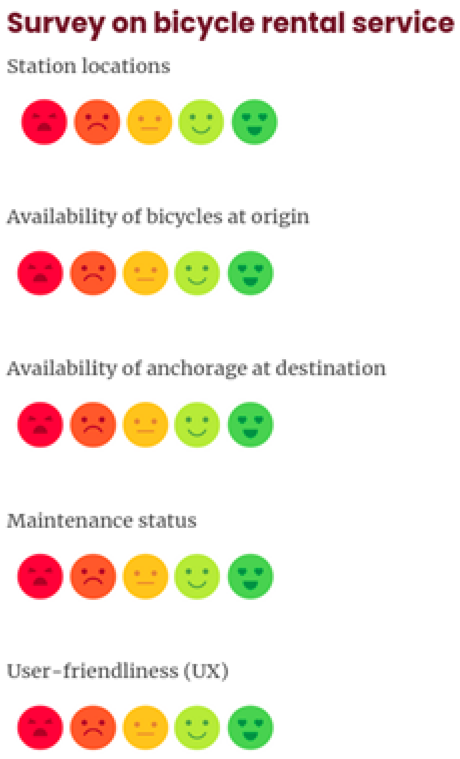
In the context of Murcia/Molina use case, the following results were obtained for the functional validation:

- 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.

7.2.2 Qualitative Validation

Regarding subjective validation, several surveys has been developed in order to assess the user experience. In Figure 29, 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 29 : Survey for qualitative validation

Table 8: Validation plan intermediary results for Murcia/Molina use case.

Validation test number	Result	Comments
1.1.1	Partial	Passed with Orion CB
1.1.2	Partial	Passed with Orion CB
1.1.3	Partial	Passed with Orion CB
1.1.5	Partial	Passed with Orion CB
1.1.6	Partial	Passed with Orion CB
1.1.7	Partial	Passed with Orion CB
1.1.8	Partial	Passed with Orion CB
1.2.1	Partial	Passed for Airquality and bike data
1.2.2	To be done	Validation tool to be developed
1.3.1.1	Passed	
1.3.1.2	Passed	
1.3.1.3	Passed	To be rechecked after final implementation
1.3.2.1	Passed	
1.3.2.2	Passed	To be rechecked after final implementation
1.3.3.1	To be done	Traffic recommendation service not finalised yet
1.3.3.2	To be done	Traffic recommendation service not finalised yet
1.3.4.1	Passed	

Validation test number	Result	Comments
1.3.4.2	Passed	
1.3.4.3	Passed	To be rechecked after final implementation
1.3.5.1	Passed	
1.3.5.2	Passed	To be rechecked after final implementation
1.3.8.1	Passed	
1.3.8.2	Passed	
1.3.8.3	Passed	To be rechecked after final implementation
1.4.1.3	Partial	Requests reception have been validated, but not subscriptions yet.
1.4.1.4	Passed	To be recheck after final implementation
1.5.1	To be done	

7.3 Flanders

7.3.1 Functional Validation

In Flanders the number of bikes forecasted will be compared to the number of bikes in real time.

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

To validate the service, we can use two options, the first option is manual, the second one is automated:

Option A

Start a set of forecasts manually and note the predicted values. Then in real time look up the real value and calculate the delta.

We plan to do that on 3 different days for 5 stations to obtain at least 20 samples with forecasting interval 15 min.

Option B

Create a script that generates calls to the forecasting service and stores the results in a table. With a second script we retrieve the actual data at the moment the forecast was made for and calculate the delta.

Together with the team we will decide between Option A and B when the service will be available. The target date for this decision is April 15th. This leaves sufficient time for fine-tuning of the service and the validation so that we can maximize the value for the user.

7.3.2 Qualitative Validation

We will contact 10 Blue-Bike users, show them the app and ask them if they would like to see the forecasting service integrated in a mobility app and in which ones: NMBS, Blue-Bike, Google Maps, other ...etc .

Table 9: Validation plan intermediary results for Flanders use case.

Validation test number	Result	Comments
1.1.1	Passed	
1.1.2	Partial	Limitations from Scorpio to be resolved
1.1.3	Passed	
1.1.5	Passed	
1.1.6	Passed	
1.1.7	Partial	Limitations from Scorpio to be resolved
1.1.8	Passed	
1.2.1	Passed	
1.2.2	To be done	Validation tool to be developed
1.3.8.1	Passed	
1.3.8.2	Passed	
1.3.8.3	Passed	To be rechecked after final implementation

Validation test number	Result	Comments
1.4.1.1	Passed	
1.4.1.2	To be done	
1.4.1.3	To be done	
1.4.1.4	Passed	To be recheck after final implementation
1.5.1	To be done	

7.3.3 KPIs Validation

The KPI for the forecasting quality will be 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. Current results demonstrate an accuracy above the KPI target.

8 Metadata Validation Tool

8.1 Introduction

Every data generated at the GreenMov could be published as open data, initially, with the aim to allow the creation of additional services by other players. These data use to be released in datasets (independently on how often is released or updated). Usually, these datasets are released in open data portals.

In order 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.

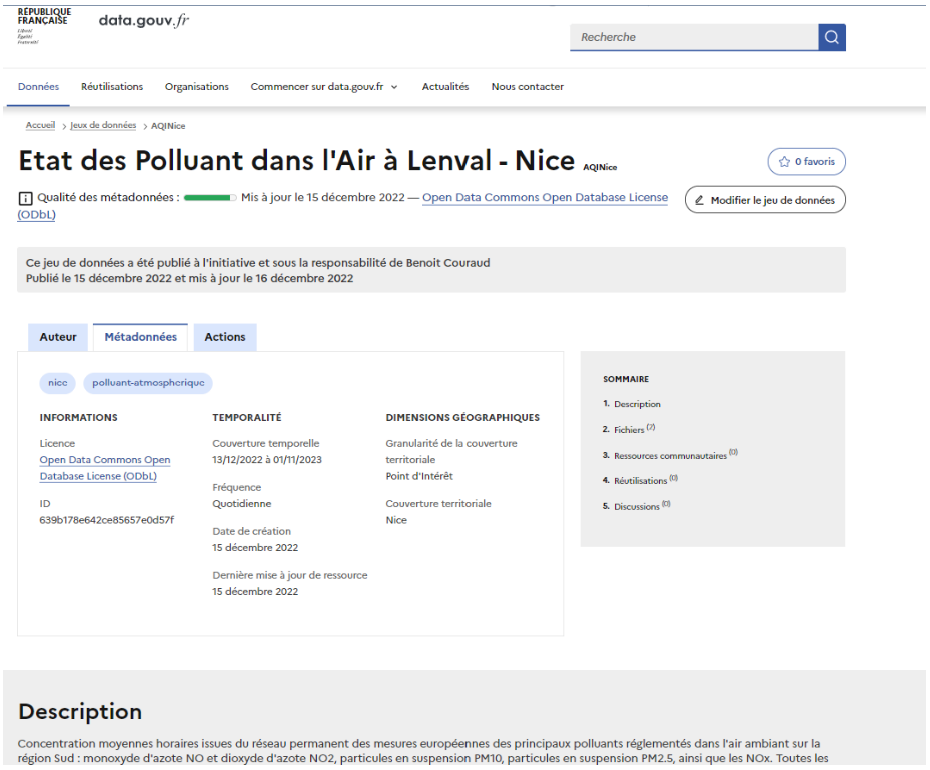
DCAT-AP is based on the DCAT standard and defines the elements of every dataset. The last version of this standard is available [17].

8.2 Results

The metadata validation tool was assessed for some datasets that are published within GreenMov . However, it appears that the validation tool does not actually validates the dataset' metadata but assess the metadata of the imported dataset to the European open data portal [17]. Indeed, taking the air quality data from Nice in France as an example, it appears that the data set's metadata is validated by the French metadata validation tool as stated in Figure 30. However, when harvested by the European open data portal, it appears that the dataset and its metadata are scrambled, as it can be seen in :

- https://data.europa.eu/api/hub/repo/datasets/639b178e642ce85657e0d57f.jsonld?useNormalizedId=true&locale=en

 and the corresponding assessment of metadata quality results in many inconsistencies as shown in Figure 31. Therefore, it should be further clarified how datasets can be imported or exported to the European open data portal without breaking all the meta data quality.


RÉPUBLIQUE FRANÇAISE data.gouv.fr

Données · Réutilisations · Organisations · Commencer sur data.gouv.fr · Actualités · Nous contacter

Accueil > Jeux de données > AQINice

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

Auteur · Métadonnées · Actions

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

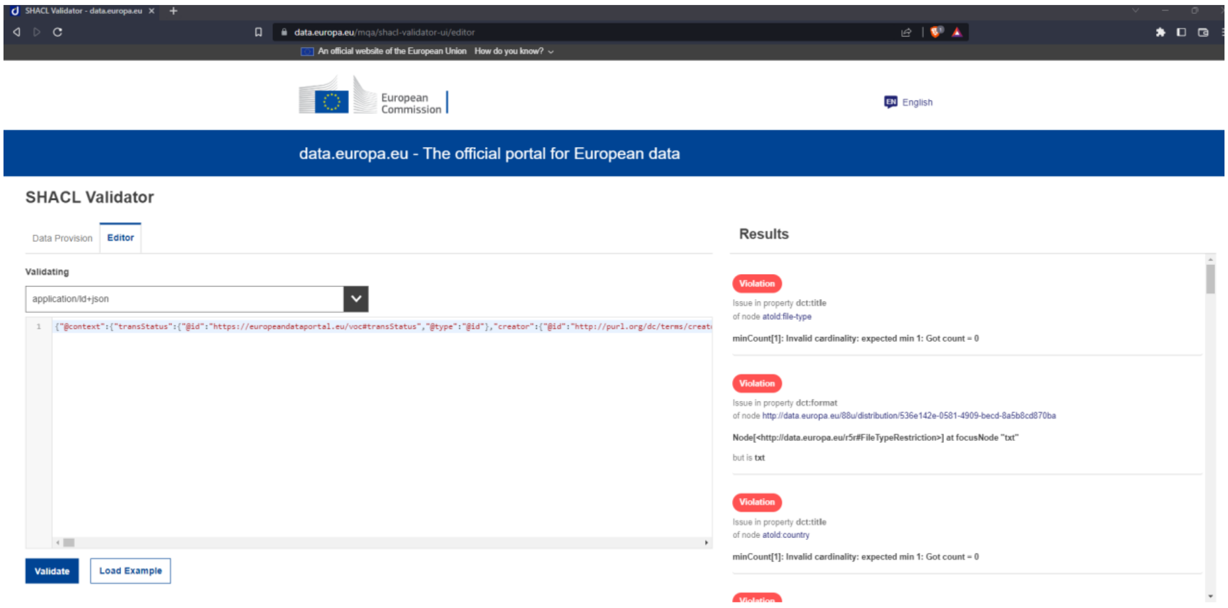
DIMENSIONS GÉOGRAPHIQUES
Granularité de la couverture territoriale: Point d'Intérêt
Couverture territoriale: Nice

SOMMAIRE
1. Description
2. Fichiers (7)
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 30 : metadata validation tool at the French open data portal level.


data.europa.eu/mqa/shacl-validator-ui/editor

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

SHACL Validator

Data Provision · Editor

Validating: application/ld+json

```
{"@context":{"transStatus":{"@id":"https://europeandataportal.eu/voc#transStatus","@type":"@id"},"creator":{"@id":"http://purl.org/dc/terms/creat
```

Validate · Load Example

Results

Violation
Issue in property dct:title
of node atold:file-type
minCount[1]: Invalid cardinality: expected min 1: Got count = 0

Violation
Issue in property dct:format
of node http://data.europa.eu/88u/distribution/536e142e-0581-4909-becd-8a5b8cd870ba
Node[<http://data.europa.eu/r5r#FileTypeRestriction>] at focusNode "txt"
but is txt

Violation
Issue in property dct:title
of node atold:country
minCount[1]: Invalid cardinality: expected min 1: Got count = 0

Figure 31 : Result of European open data portal meta data validation tool for a dataset automatically imported by the European opendata portal.

9 Conclusions

In this document, we carried the description of the use cases to be deployed in Murcia/Molina, Nice and Flanders cities, the data models used, created or updated, the detailed green mobility services and sub services adopted for deployment and their architecture as well as their specificity for each use case,

The use cases deployment described in this document includes the implementation steps, the technical deployment and the replication process so to be able to replicate the results of the project,

This document also covers the validation plan of each use case including the functional and the subjective validation as well as the KPIs validation,

Moreover, the metadata validation tool used in Activity 4 is also elaborated in this document.

The main challenges addressed within this deliverable are the software and hardware architecture of the services and the harmonization of these infrastructures taking in consideration the specificities of each use case, and also the deployment steps and the replication process which are crucial to the aims of this project,

In alignment with the project roadmap, the results of this document will be used as a reference for the final implementation of the services and as a tool to validate the KPIs and to measure the end uses and stakeholders engagement for the next Activity 5 deliverable D5.3.

10 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.