º

URBANITE

Supporting the decision-making in urban transformation with the use of disruptive technologies

Deliverable D5.9

URBANITE Ecosystem-v3

Editor(s):	María José López
Responsible Partner:	TECNALIA
Status-Version:	Final – v3.0
Date:	30.04.2023
Distribution level (CO, PU):	PU

Project Number:	GA 870338
Project Title:	URBANITE

Title of Deliverable:	URBANITE Ecosystem-v3
Due Date of Delivery to the EC:	30.04.2023

Workpackage responsible for the Deliverable:	WP5 - URBANITE ecosystem integration and DevOps
Editor(s):	María José López (TECNALIA)
Contributor(s):	Iñaki Etxaniz/Gorka Benguria/María José López/Alejandro Rodríguez/Iñaki Olabarrieta (TECNALIA) Giuseppe Ciulla (ENG) Maj SmerKol (JSI) Torben Jastrow (FHG)
Reviewer(s):	Sonia Bilbao (TECNALIA)
Approved by:	All Partners
Recommended/mandatory readers:	WP3, WP4, WP5

Abstract:	Report related to the URBANITE Ecosystem-v2, describing the environment where it is deployed, and the methods and resources used for the integration of the corresponding components.
Keyword List:	DevOps framework, integration, environments, prototype.
Licensing information:	The license of the URBANITE Ecosystem software is under discussion and will be detailed in the deliverable D7.9. The document itself is delivered as a description for the European Commission about the released software, so it is public.
Disclaimer	This document reflects only the author's views and neither Agency nor the Commission are responsible for any use that may be made of the information contained therein

Document Description

Document Revision History

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
v0.1	20/04/2023	First draft version to be checked and completed by the technical partners	TEC
V0.2	24/04/2023	Contributions from ENG	ENG
V0.3	30/04/2023	Updates based on Internal reviewer's comments	TEC
V0.4	27/04/2023	Updates from ENG related to the UI and from TEC about the anonymization module and internal review.	TEC
V1.0	28/04/2023	Final version ready to be submitted	TEC

Table of Contents

List of Figures

List of Tables

Terms and abbreviations

JSON	JavaScript Object Notation
XML	eXtensible Markup Language
API	Application Programming Interface
REST	REpresentational State Transfer
MQTT	Message Queuing Telemetry Transport
DCAT-AP	Data Catalogue vocabulary Application profile for data portals in Europe
UI	User Interface
GUI	Graphical User Interface
DSS	Decision Support System
OD	Origin/Destination
IDM	IDentity Manager

Executive Summary

This deliverable is the final of a series of three (D5.7 [1], D5.8 [2] and this D5.9), covering the description of the final version of the URBANITE Ecosystem, integrating the final version of the components developed by the technical work packages. This deliverable presents the final release of the URBANITE Ecosystem, and it substitutes them.

The main updates are related to the requirements, the description of the components, local installation instructions and the complete manual version.

The URBANITE Ecosystem architecture remains stable in the core components since the D5.5 [3] deliverable and is considered as the final structure of the URBANITE platform. The modifications since then are included in the corresponding section.

The requirements are almost 100% covered, and an explanation for the rest is reported in the corresponding section. Nevertheless, if some interesting suggestion arises from the evaluation process, the technical partners will discuss the convenience of including those changes, always for the project's sake. These modifications will be part of the final deliverables of the WP6.

Moreover, this document reports the description of a new component belonging to the data management layer that was not included in any deliverable of the WP3.

1 Introduction

1.1 About this deliverable

The main content of this deliverable is the architecture and software of the final release of the integrated URBANITE Ecosystem, including the final versions of the components as well as the user manual of the Ecosystem. This final version will be used for the evaluation performed in the use cases.

This release covers all the technical features described in previous deliverables, regarding requirements whose last version was reported in D5.2 [4] and the general architecture reported as well in D5.8 [3].

Moreover, this document includes the approach for the deployment process of the DevOps strategy used for the integration and execution of the platform in the three planned releases of the URBANITE Ecosystem.

This deliverable is a self-contained document and replaces previous versions as it provides updated information on all the sections that describe the URBANITE Ecosystem. This strategy makes the final description more readable for the final users.

1.2 Document structure

The document is structured in four sections:

- The first section introducing the context related to this deliverable within the project, explaining the objectives and the structure of the document.
- The second section gathers the main requirements and functionalities covered by this prototype and the architecture reflecting the components integrated in it.
- The third section presenting the installation process of the prototype and its deployment, so users can run and test it.
- The fourth section presents some final conclusions.

Besides, there are 2 additional annexes:

- ANNEX I with the instructions for installing the Ecosystem in a private infrastructure.
- ANNEX II describing the user manual of the whole platform.

2 Implementation

2.1 Functional description

The final release of the URBANITE Ecosystem covers the main requirements and functionalities of the platform.

This deliverable is considered the final prototype of the URBANITE Ecosystem, accomplishing the seventh Milestone named "Final release of URBANITE Ecosystem".

In order to set a platform for supporting the evaluation of the different use cases, four customized instances of the platform have been deployed, so the particular components developed for each of the use cases can be observed in a separate way.

The platform provides an entry point to the URBANITE UI from where the users of the different use cases can access the developed functionalities.

Requirements:

The requirements gathered and refined through the 3 versions of the URBANITE Ecosystem are listed in the tables below. Updates from previous releases are highlighted in green.

Each table can be mapped to a layer in the URBANITE Architecture, described in section 2.2.

Table 1: Virtual SoPoLab requirements.

Req ID	Req. Description	Final Prototype situation
VSPL.01	VSPL should allow collaboration among its users, enabling co-creation approach. In the case of URBANITE, the co-creation sessions will be oriented to address and analyse the issues/barriers/ lack of trust of the usage of disruptive technologies in the public sector.	Covered.
VSPL.02	The users of the VSPL should be able to report needs in the context of the analysis of the attitudes, trust, and barriers in the use of disruptive technologies	Covered.
VSPL.03	The VSPL must allow to create challenges to solve the needs expressed related to the usage of disruptive technologies in the Public Sector	Covered.
VSPL.04	The users of the VSPL should be able to report ideas (possible solutions) to address the lack of trust, usage reticence, problems, needs of the usage of disruptive technologies in the Urban Mobility context.	Covered.
VSPL.05	The VSPL must allow to evaluate those proposed ideas to address the problems /needs related to the usage of disruptive technologies by the Public Administrations (Pas) for urban mobility.	Covered.
VSPL.06	The VSPL must allow selecting the best ideas to be refined and implemented in the context of the usage of disruptive technologies by the PAs for urban mobility.	Covered.
VSPL.07	The VSPL must allow to suggest refinements for selected ideas.	Covered.

VSPL.08	The VSPL must allow to select ideas to be implemented in the context of the usage of disruptive technologies by the PAs for urban mobility.	Covered.
VSPL.09	The VSPL must allow to host different kinds of resources created by the project, i.e., guidelines, methodologies, best practices.	Covered.
VSPL.10	The VSPL must allow the exchange of information between different participants of different nodes and cities.	Covered.

Table 2 Data Harvesting requirements (detailed information available in deliverable D3.3 [5] "Data Harvesting Module and Connectors Implementation v2")

Req ID	**Req. Description**	**Final Prototype situation**
DH01	The harvesting component will retrieve data from various sources (municipal services, open data portals, GIS, city private service providers) with varying formats (e.g. JSON, XML) from different data sources (e.g. open/private data portals, GIS system), raw data from APIs or data coming from sensors.	Covered.
DH02	Data Harvester should allow pagination of large amounts of data. This means that in case some data source APIs cannot provide data in bulk, the harvesting component should be able to fetch only chunks of limited size until all data has been harvested.	Covered.
DH03	Data Harvester should be extensible with new connectors if new, unsupported data sources are discovered.	Covered.
DH04	Data harvester must support at least HTTP(S) and MQTT protocol to fetch the data.	Covered.
DH05	For client/server APIs the harvester will download data from the configured APIs at recurring intervals of varying length (e.g. daily, weekly). The schedule will depend on the volatility of data. For example, weather data will change more frequently than map data highlighting current road construction work.	Covered.

Table 3. Data Curation, Preparation, Transformation and Anonymisation requirement (detailed information available in deliverable D3.6 [6] "Data curation module Implementation v2")

Req ID	**Req. Description**	**Final Prototype situation**
DC01	The harvested data may not be in a format and/or structure suitable for data storage. In this case, the data will need to be transformed in an automated way.	Covered.
DC02	Data curation functionality should be able to clean the data coming from the harvester eliminating duplicates or error.	Covered.
DC03	Data Transformation functionality should add an annotation in the form of metadata to data to help the analysis. This metadata will be included in the data itself.	Covered.

DC04	This functionality shall anonymize or pseudonymize data if the need arises. Data anonymization could be done at the source or before storing it, depending on the use case. In any case, URBANITE platform can provide the anonymization functionality for users (UCs) to use it before the data is uploaded/used by the URBANITE platform	Covered, the new component Data Anonymization has been included in the Data Management Layer. It allows a) to anonymize generic values by transforming them to a hash value, and b) to anonymize trajectories by removing GPS locations outside an area or by removing the start and end GPS locations closer to a given distance in meters.
DC05	Data validation and quality check. The data curation functionality must be able to validate the data provided by the data harvesting module and its quality based on a defined format if encountered data sources happen to contain sensitive information.	Covered.
DC06	Functionalities should be provided to transform cleaned and annotated data to common semantics and data models to guarantee interoperability. It is important to note that there will not be one single common format that all data will be transformed into. Instead, established formats within the various domains will be targeted for transformation.	Covered.
DC07	The data preparation functionality must check the data licenses and provide understandable information to the owners and the user of the data. For combined data sets with different licenses, it detects possible compatibility issues and informs users how to use and share the data.	Partially covered by the Catalogue module including the link to the JoinUp page to check licenses compatibility.
DC08	The data curation functionality (in case of being an algorithm or process) must: • provide an API REST for launching the process and passing the parameters • or allow an MQTT endpoint to be aware of data publication and the launch of the process	Covered.
DC09	The data cleaning functionality must be capable of detecting and removing invalid or missing readings. The result should then be fit in terms of quality and type, for further processing.	Covered.
DC10	Some components in the architecture diagram are labelled as "triggered by user", namely Data Curation and fusion/aggregation. For this to be possible, they must feature a UI that allows for configuration and triggering of the respective functionalities.	Covered.

Table 4. Data Fusion/Aggregation requirements (detailed information available in deliverable D3.8 [7] "Data aggregation and storage module implementation v2")

Req ID	Req. Description	Final Prototype situation
DF01	The component should allow to aggregate curated data coming from different data sources if needed.	Covered
DF02	The component should allow the deduplication of the data.	Covered
DF03	The data should be mapped into EU vocabularies	Covered
DF04	The metadata should be mapped into DCAT-AP metadata. DF03 is required for this one.	Covered
DF05	Weather data coming from different data sources and weather services will be fused to create improved datasets covering more variables	Covered with other datasets as they have been fused as explained in 2.2.2 of D3.8 [7]
DF06	The component should allow temporal aggregation of traffic data at given intervals, e.g., every 15 min.	Covered (Aggregation component)
DF07	The component should allow calculating maximum, minimum, average, and standard deviation values of datasets in a given interval, e.g., daily, monthly, etc.	Covered (Aggregation component)

Table 5. Data Storage & retrieval requirements (detailed information available in deliverable D3.8 [7] "Data aggregation and storage module implementation v2")

Req ID	Req. Description	Final Prototype situation
DS01	The harvested data should be persistent with a big-data-storage solution capability.	Covered
DS02	The data storage component should be able to process and store DCAT-AP compliant metadata.	Covered.
DR01	The data retrieval component must expose API to retrieve and query the data stored in the different repositories	Covered.
DR02	The metadata stored in the repositories should be accessible through a data hub in a uniform way taking advantage of the DCAT-AP standard and related profile.	Covered

Table 6. Data Catalogue requirements (detailed information available in deliverable D3.8 [7] "Data aggregation and storage module implementation v2")

Req ID	Req. Description	Final Prototype situation
DCA01	The data catalogue should be able to retrieve existing metadata from existing heterogenous Open Data Portals.	Covered.
DCA02	Data Harvester should be extensible with new connectors if new unsupported data sources are discovered.	Covered.

DCA03	The Data Catalogue has a built-in scheduler that is able to synchronise the federated catalogues (collecting metadata) at recurring intervals.	Covered.
DCA04	The data catalogue, being one of the main interfaces to the users, must feature a UI that covers all relevant functionalities of the data catalogue.	Covered.
DC05	Data Catalogue will provide a wizard to create charts.	Covered
DC06	The Data Catalogue should allow downloading of transformed data stored in the URBANITE repositories.	Covered

Table 7. Advanced Visualization requirements

Req ID	**Req. Description**	**Final Prototype situation**
AV01	The harvested data must be available to be visualised through analysis and simulations provided by the URBANITE Ecosystem.	Covered.
AV02	The component must allow to visualize the analysis results on a combination of map layers, heat maps, traffic flow graphics and other kind of visualization to help with understanding data e. g.: • a description of the layers and base maps • show different charts and graphs in the same view • allow the activation and deactivation of map layers • allow the user to make publicly accessible selected charts, graphs, map layers • allow users to access to the results of the analysis	Covered.
AV03	The component must allow users to interact with the visualized data by, for instance, zooming, highlighting, and displaying additional information.	Covered.

Table 8. Exploratory Data Analysis requirements

Req ID	**Req. Description**	**Final Prototype situation**
DP01	Data projection component will provide dimensionality reduction methods for a better understanding and interpretation of the data.	Covered
DCL01	Data Clustering component will provide methods that will identify groups of similar objects in the data (based on user-defined attributes) and interactively present them to the user.	Covered
SOM01	The Self-Organizing Map will provide the user with a visual topological representation of the data, able to highlight potential clusters.	Covered
REG01	The regression component will enable the user to investigate the relationship between different variables, and to actively search for causal relations in the data.	Covered
PRED.01	The prediction component will provide an engine to produce prediction for a traffic/mobility variable defined as a time series considering a series of time defined features.	Covered

Table 9. Traffic Simulation requirements

Req ID	**Req. Description**	**Final Prototype situation**
TS01	Traffic Simulation component will provide urban traffic simulation based on the collected data, describing the traffic flow locally, for specific parts of interest in the city and combining it in a hierarchical manner.	Partially covered
TS02	Traffic Simulation component will provide the ability to simulate hypothetical situations and the effects of different measures.	Covered
PSV01	The component should support policy-makers for identifying possible policies that tackle events based on specific criteria.	Not covered
PSV02	The component should predict and classify traffic flow changes according to the changes in the policies.	Partially covered
PSV03	Users must be able to select the defined KPIs to evaluate policies.	Covered
PSV04	The component must assign a score to each policy to help the decision-making process.	Partially covered
PSV05	Policy-makers will be able to make an informed decision about which policies should be deployed in the city.	Covered
RE01	The Recommendation Engine will provide suggestions to tackle the potential problems in the city traffic. This component will also provide support for identifying possible policies that tackle events based on specific criteria.	Partially covered
RE02	The recommendation engine must identify and predict events related to mobility (samples could be congestion situations, high-emission scenarios, unbalanced modal share, etc.) based on the analysis of existing models and/or simulated data. Such analysis will be supported by the previously mentioned component as those related to regression, clustering, simulation, or additional ones.	Partially covered
RE03	The recommendation engine should provide support and suggestions to the policy-makers for identifying possible policies that tackle identified problems and undesired events related to mobility based on specific criteria. Effective hierarchical multi-criteria decision models based on aggregated data and a rule-based approach will be adopted.	Partially covered

Table 10. Analytical Framework requirements

Req ID	**Req. Description**	**Final Prototype situation**
AF01	The bike analysis sub-component provides an engine to produce models to compute OD matrixes for bike city services considering different timing attributes such as the day of the week or a specific hour in a day. In addition, different zoning options can be considered for the calculation.	Covered
AF02	The traffic prediction sub-component allows to produce prediction models to compute prediction for the flow of vehicles	Covered

	at the locations of the traffic flow sensors considering the day of the week or for a specific hour. In addition to the raw prediction, the models are capable to provide an interval of confidence for the generated result values.	
AF03	The application SHOULD automate part of the analysis performed on the collected data (e.g. extract relevant information and provide it in a more usable manner)	Covered

Note: More analysis modules have been developed, but not reported as requirements. They will be included in section 2.2.3 as part of the components of the prototype.

Table 11. URBANITE UI requirements

Req ID	Req. Description	Final Prototype situation
UUI01	The UI must provide uniform access to URBANITE tools and components.	Covered
UUI02	The UI must be integrated with the DSS visualization capabilities.	Covered
UUI03	The UI must support different user profiles, offering different functionalities for administrators and final users.	Covered.
UUI04	The UI must be responsive to support different types of devices.	Covered.
UUI05	The UI must allow personalisation through custom dashboards.	Covered
UUI06	The UI should allow sharing custom dashboards among the users.	Covered
UUI07	The UI must include functionalities for the identification of URBANITE users.	Covered.
UUI08	The UI must allow the management of roles and groups of the users.	Covered.
UUI09	The UI must provide functions for user management (e.g., searching for users, creating and/or editing and/or deleting users).	Covered.
UUI10	The application MUST be accessible through a Web Browser.	Covered.

2.2 Technical description

2.2.1 Continuous integration overview

The technical strategy in the URBANITE development, adopted and applied along the whole project, is described in the D5.3 deliverable [3], and it is based on a DevOps approach for the development of the components.

The software components that make up the final version of the URBANITE Ecosystem have been implemented by different partners, under different technologies and prepared to be deployed as docker containers, in order to facilitate the deployment. All the micro-services communicate with each other through RESTful APIs over the HTTPS secure protocol.

As detailed in the Integration Strategy, the DevOps approach was based on three environments, as depicted in Figure 1. The Development environment is owned by each developer, provided by each partner, and is where the software is developed. The Integration environment is provided by Tecnalia, and is where the code is compiled, merged, and tested. Finally, the Pilots environment is where the compiled software is deployed and executed.

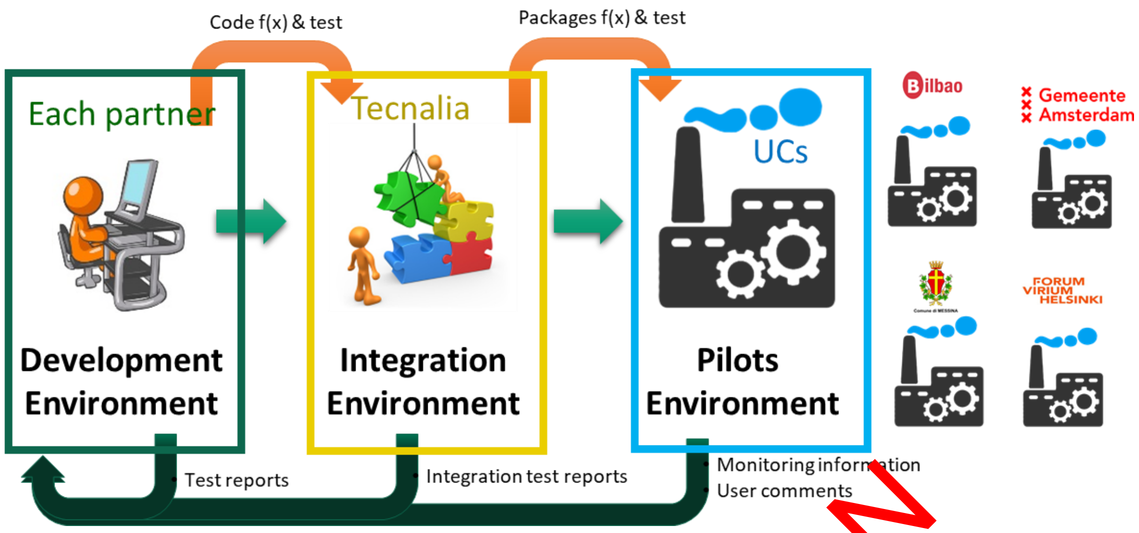


Figure 1. The three environments in URBANITE.

The city of Messina has deployed a local installation of the URBANITE Ecosystem, whereas the other municipalities have used the pilot environments provided by Tecnalia. The description of the local installation is reported in the ANNEX 1.

The different environments used for the continuous integration within the URBANITE Ecosystem are depicted in Figure 2.

Our solution uses Continuous Integration (CI), Continuous Deployment (CD) practices. The Continuous Integration practice includes the management of the software source code through a versioning control system and for this purpose, all the URBANITE components are available in a private GitLab repository. The Gitlab tool is also used for CI/CD in URBANITE. It provides the option of using branches and virtual environments, that have been mapped to the real environments described before. Thus, we have the *feature* branch, the *develop* branch, the *master* branch, and the *pilots* branch. At the same time, we have defined the environments develop, master, and one environment for each pilot.

The pipeline implemented ad-hoc for URBANITE is composed by the jobs Build, Deploy, Clean and Promote. The Build pipeline is triggered automatically at every push of the project in GitLab, and it automatizes the build of the project, the creation of the Docker image and its push to the Artifactory. If this previous pipeline succeeds, the second Deploy pipeline is triggered and will automatically deploy the component to the development environment.

The workflow is the following: when a developer uploads a new version of their components to the integration environment, the integration process starts compiling the code and testing it in a temporary environment (*Feature branch*). Afterwards, the code is merged in the *develop* environment, where the whole ecosystem is built and tested again. From this environment, in a further step, the developer can manually promote the code to the *demo Pilots* environments.

At any moment, the integrator can clone the actual version to the *Master* environment to maintain a stable version accessible, out of the integration up and downs.

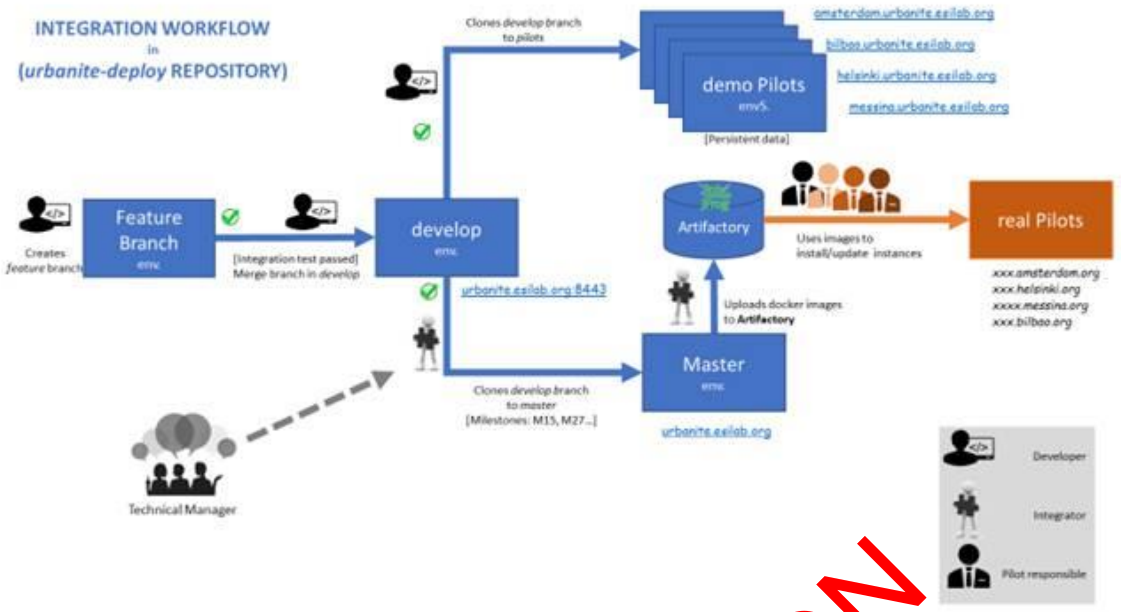

INTEGRATION WORKFLOW
in
(urbanite-deploy REPOSITORY)
Creates feature branch
Feature Branch
env.
[Integration test passed]
Merge branch in develop
develop
env.
urbanite.esilab.org:8443
Clones develop branch to pilots
demo Pilots
envS.
[Persistent data]
amsterdam.urbanite.esilab.org
bilbao.urbanite.esilab.org
helsinki.urbanite.esilab.org
messina.urbanite.esilab.org
Artifactory
Uses images to install/update instances
real Pilots
xxx.amsterdam.org
xxx.helsinki.org
xxxx.messina.org
xxx.bilbao.org
Uploads docker images to Artifactory
Master
env.
urbanite.esilab.org
Clones develop branch to master
[Milestones: M15, M27...]
Technical Manager
Developer
Integrator
Pilot responsible

Figure 2.URBANITE integration workflow

A brief description of these environments and their function follows (this can also be consulted in the README.md file of the integration repository[1], where the description of the environments, the components and their access points and the installation instructions are included).

- **FEATURE BRANCH:** Temporary environment that is created each time a developer wants to integrate a new version of his component. It just checks that the new version of the Urbanite platform builds without problems and is destroyed afterwards.

- **DEVELOP:** Environment that contains the last version of the components running together. Dedicated to test new features, interfaces, and communications among components. Available at urbanite.esilab.org:8443.

- **MASTER:** Contains a specific version of the platform, frozen for determined Milestones if needed. In those cases, the version should be available at urbanite.esilab.org.

- **DEMO PILOTS:** Four environments, one for each city, where the integrated platform is replicated and adjusted to the characteristics of the use cases. It is a previous step for testing the platform before setting it up in the infrastructure of the municipalities, and some of these municipalities can evaluate their use cases using these environments:

    - amsterdam.urbanite.esilab.org
    - bilbao.urbanite.esilab.org
    - helsinki.urbanite.esilab.org
    - messina.urbanite.esilab.org

- **REAL PILOTS**: the possible installation of the platform in each municipality's infrastructure. The municipalities that want to deploy the URBANITE Ecosystem in their

[1] https://git.code.tecnalia.com/urbanite/private/urbanite-deploy/-/blob/develop/README.md

own infrastructure for evaluating the use cases. At this point only the Messina municipality has expressed its wish to deploy this real pilot.

Apart from that, in order to support developers during the integration, we provide:

- A **Portainer** [4] instance that allows access to the logs and the console of every container in every environment.
- An **Artifactory** instance to store the images of the containerized components. These images will be used to deploy the final version of the platform in the real Pilots.

2.2.2 Architecture

The final version of the URBANITE Ecosystem-v3 is almost the same as the one reported in the deliverable D5.8 [2]. No more important changes have been defined with the exception of the new component "anonymization". Every pilot follows the same structure, based on this architecture and including its particular components, mostly related to the analysis and simulations provided and the data sources collected.

The particular details are explained in the D5.5 [3] and the components that make up the URBANITE Ecosystem are depicted in Figure 3. The Analytical framework includes different modules for each of the pilots and is described in the 2.2.3 section of this deliverable.

The new anonymization component is described in 2.2.3.1 section of this document.

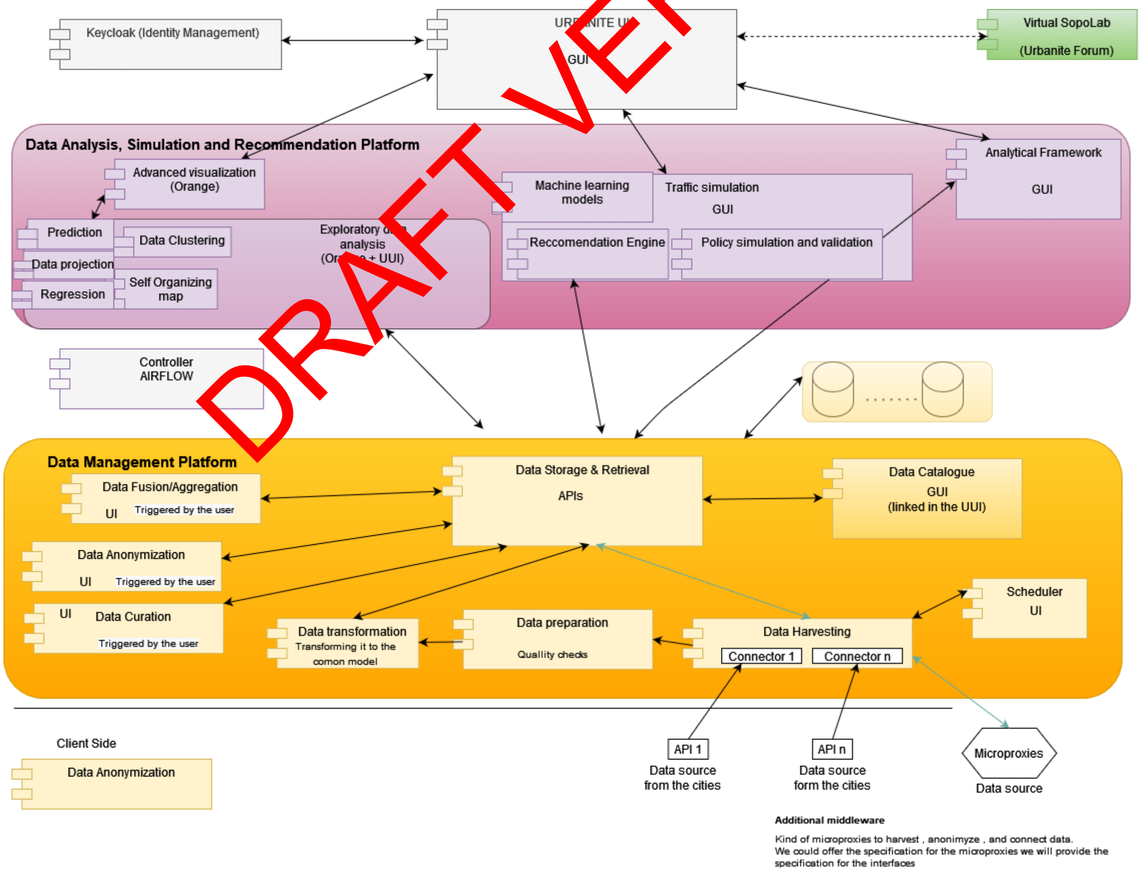


Figure 3.URBANITE Ecosystem-v3 Architecture

The Messina use case includes a particular process for exposing data to the URBANITE Ecosystem. The reason behind these additional components is to preserve the privacy of the data gathered by the municipality. The so-called Messina Edge components prepare and make available under a unique endpoint the data to be harvested by the URBANITE Platform, simplifying the connection with the latter.

The components that comprise this Edge and depicted in Figure 4 are:

- Data importer: this component collects data from the different sources related to the City of Messina.
- Messina data storage: this component stores the data collected by the Data importer.
- Data processor: this component exposes the APIs to access the stored data.
- GUI: provides a web portal offering information about the stored data and the specifications of the APIs to access the available data.

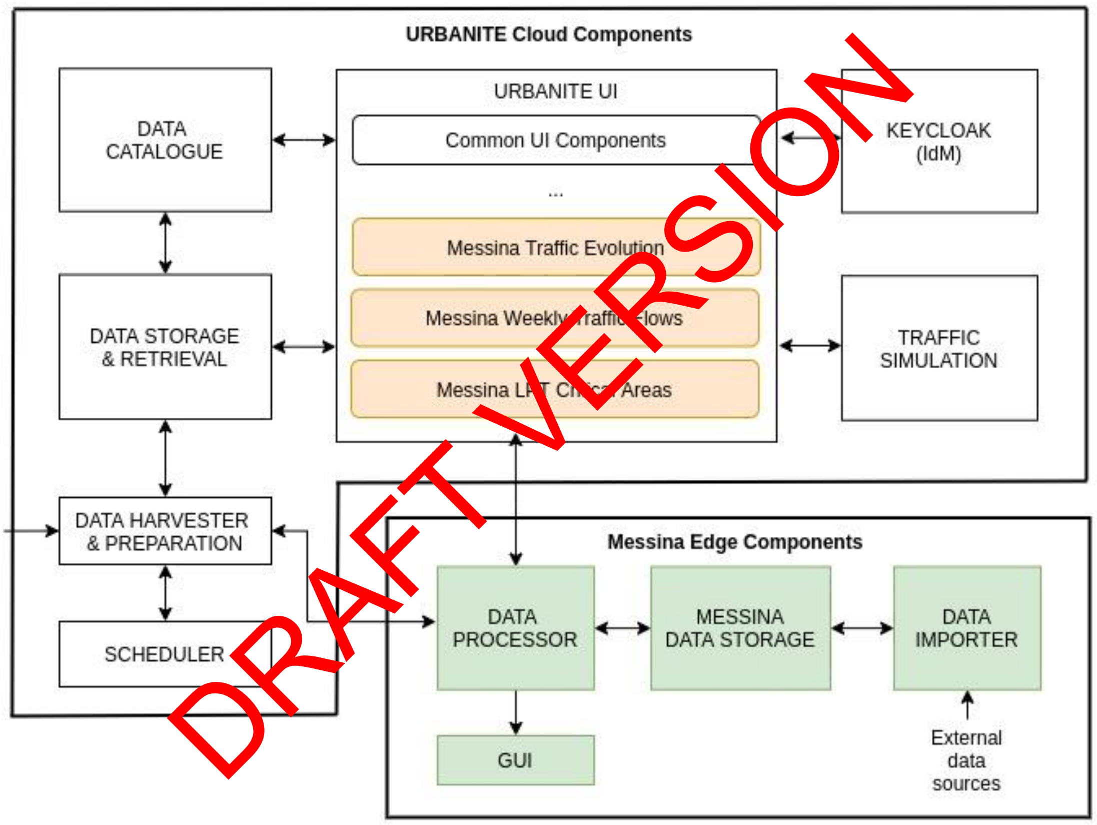


Figure 4 Deployed components of Messina use case

The interaction with the URBANITE Ecosystem is through three of the options under the Data Analysis option of the Messina use case implemented within the URBANITE UI and consists of a direct access to the data exposed by those Messina Edge Components. More details about the Messina use case are reported in D6.2 [10].

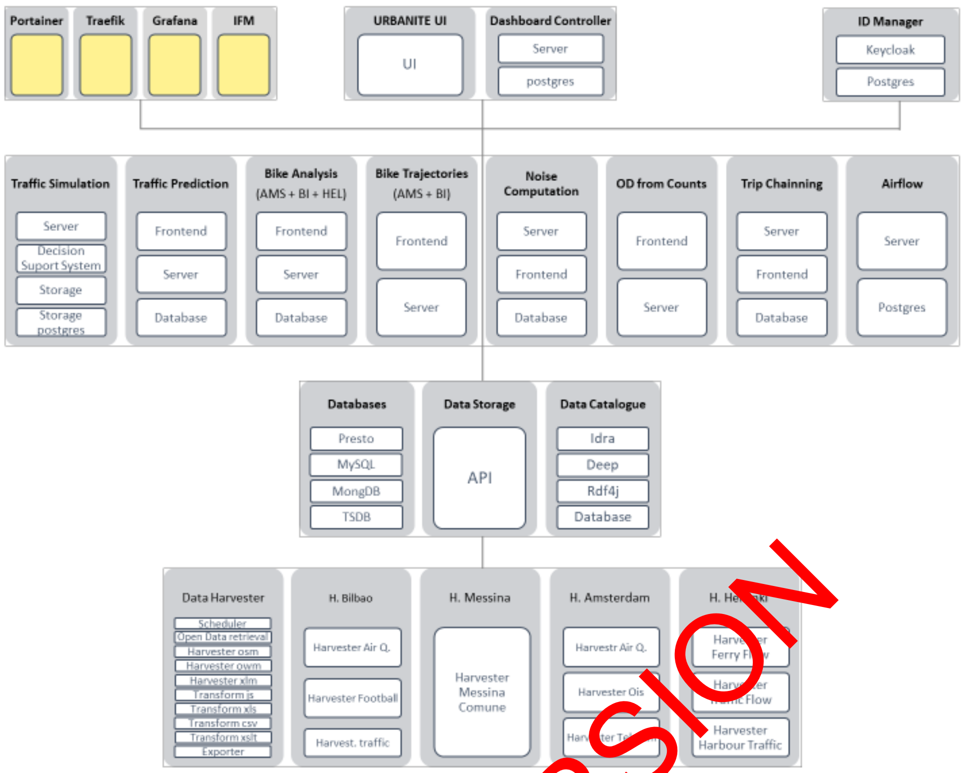


Figure 5.URBANITE Ecosystem-v3 containerized components

The integration process encapsulates all the components as containers for facilitating future deployments of the URBANITE Ecosystem in different installations. Every component is composed by one or more Docker containers and presents a REST interface to the rest of them.

The diagram with all the components and their corresponding containers is depicted in Figure 5.

2.2.3 Component description

The components integrated into the URBANITE Ecosystem have been implemented by the technical partners. The functionalities provided cover the requirements established by the deliverable D5.2 [4].

The technologies used by the URBANITE UI component are the base for creating the specific UIs of the different components, needed for interactions with the user. The UI development team provided a template with detailed descriptions in the readme file of the URBANITE UI repository. This template is built on NGX-Admin [11], an open source dashboard based on Angular [12], Nebular [13] with Eva Design System [14].

The different components are grouped by layers or platforms, attending to the nature of the functionality provided. This structure is depicted in Figure 3.

2.2.3.1 URBANITE Data Management Layer

The Data Management Platform gathers distinct software components that work together to deliver the key functionalities:

- data harvesting
- data preparation
- data transformation
- data curation
- data anonymisation
- data aggregation/fusion
- data storage.
- data catalogue

Detailed information on these components is available in their corresponding description deliverables D3.3 [5], D3.6 [6], and D3.8 [7]. The Data Management Layer follows a microservice architecture.

All the data processes follow a pipeline of steps, i.e., first to import data and metadata from endpoints on the web, by the harvesting module, second to check, clean and harmonise the different kinds of data and metadata by the data preparation, transformation, and curation components, and third, once the data and metadata are brought into a common format, to store in dedicated databases.

Additionally, the Scheduler component manages the regular intervals for downloading the data (and metadata), triggering the data importers which in turn download the data.

These components are not accessible from the URBANITE GUI and need to be run from a particular UI. The data collected by this layer are used by different components to run their analysis and simulations. Moreover, through the data catalogue, metadata describing the datasets collected are presented to the users.

The available datasets related to the use case and stored in every pilot are described in the D3.3 [5]. Besides, from the time of writing of D3.3., new datasets have been included which are:

- Noise simulations for each city: map layer that shows the level of noise in the city areas and variation along the day, time and weather (calculated by WP4). These data is transformed into heatmaps and visualized in the URBANITE UI.

- Ferry stations in Amsterdam.

- 30 km roads in Amsterdam

- Telraam Traffic data (car traffic, heavy traffic, pedestrian traffic, bike traffic): Available from 1st Oct 2022 till present.

The main functionalities of the Data Catalogue are related to the discovery of datasets managed by the Data Storage and Retrieval. It also allows the federation with other existing data catalogues. The Idra tool is integrated into the platform and adapted to interact with the API exposed by the Data Storage and Retrieval. Furthermore, it is able to schedule the update period to check the availability of new datasets or updates related to datasets already available.

At the time of writing the final deliverables in WP3, the Data Management Layer did not contain an anonymization module as all the data managed by the platform was already anonymized and there was no sensitive data. However, afterwards, the Messina pilot managed to obtain datasets of bike GPS locations identified by a SIM number. These data are considered sensitive data, as a user's home, working place and behaviour, can be identified based on GPS trajectories in a time period. In order to anonymize these bike trajectories, WP3 developed a new component that is able to anonymize GPS trajectories in 3 ways.

- Anonymization component (new component)

This component provides an API that exposes 3 methods as REST Web Services. The first method "anonymize", is valid for any data model. Given a property in the model, the process transforms the value in that property to a hash value. This method can be used e.g., to anonymize identifiers or as in the case of the bike trajectories, to anonymize the SIM value.

POST/anonymize/{city}/{model}/{property}/{value}

The second method, removePointsOutOfArea, removes from the repository all the GPS points that are located outside the area that is provided as input polygon.

POST/removePointsOutOfArea/{city}/{model}/{property}/{value}

The third method, generateAnonymousVehicleTripFromVehiclePoints, anonymizes the trajectories by removing the start and end GPS locations of the trajectory whose distance to the initial and end points is smaller in meters than the input distance "d" value.

POST/generateAnonymousVehicleTripFromVehiclePoints/{city}

This anonymization process is run on the harvested data so the information stored in the data storage has been anonymized using a distance d=50m. The ebikes anonymized trajectories in Messina are available at:

https://messina.urbanite.esilab.org/data/getTData/gtfsShape/messina?filters=%7BalternateName%3A%22ebiketrajectories%22%7D

2.2.3.2 *URBANITE data analysis, simulation, and recommendation layer*

The components that are integrated in the URBANITE Ecosystem v3 are:

- Analytical Framework

Within the context of the WP4, some modules for data analysis have been developed to help the end user to make decisions and define urban policies in each pilot. The details of them are reported in the WP4 deliverables.

Table 12: Data Analysis Components per pilot.

Data Analysis component	Data	AMSTERDAM	BILBAO	HELSINKI	MESSINA
Traffic Prediction	Traffic Counts		X	X	
Global Traffic Prediction	Traffic Counts		X	X	
Noise Computation	Simulated Data	X	X	X	X

Bike OD Matrix	Bike Rentals	X	X	X	
Bike Trajectories	Bike Trajectories		X		
Bike Data	Ring-ring	X			
Bus OD Matrix	Bus Smart Card		X		
Traffic OD Matrix	Traffic Counts		X		
Traffic Evolution	Traffic				X
Weekly Traffic Flows	Traffic				X
LPT Critical Areas	Traffic				X

Messina Analysis

This paragraph illustrates the analyses carried out thanks to the in-house data made available by the Municipality of Messina itself.

The architectural unit developed in the CED Municipality of Messina, has the capability to operate as URBANITE module at the EDGE (so called the Messina Edge components), useful for increasing the functionalities of URBANITE services guaranteeing Security and Privacy at the Edge, because the module offers aggregate data by API, and does not expose specific Municipality sensitive data.

In the case of the traffic evolution services the dashboard and its graphs *(Messina Traffic Evolution and Weekly Traffic Flows*) are displayed in an aggregated manner, following the just mentioned approach, in which the component accomplished in collaboration between the Municipality of Messina and Alma Digit called **EdgeTrafficMessina** has been deployed. Indeed, in this scenario this module does not expose Municipality data coming from in city deployed cameras. The original data source was 'cleaned' and 'filtered' of information not needed for the particular analysis due to user privacy concerns (removing of sensitive data).

In particular, all information related to:

- number plates
- timestamp
- images/videos.

This information, in fact, is part of the meta-data collection provided by the platform already own by the Municipality od Messina named "Mesm@rt" (see Figure 6) and to which the accomplished edge component connects via REST-API.

The only thing left for analysis was the counting by the time, of events related to the transit of vehicles in front of a camera on a given road section.

At the beginning, in the first testing periods, the analyses were based on data collected via API from the here.com service.

Then, the Municipality, together with Alma Digit, started gathering information from both the Urbanite project cameras and the ones already existing in the municipal area, thus creating, together with the analyses, its own historical source of city traffic data.

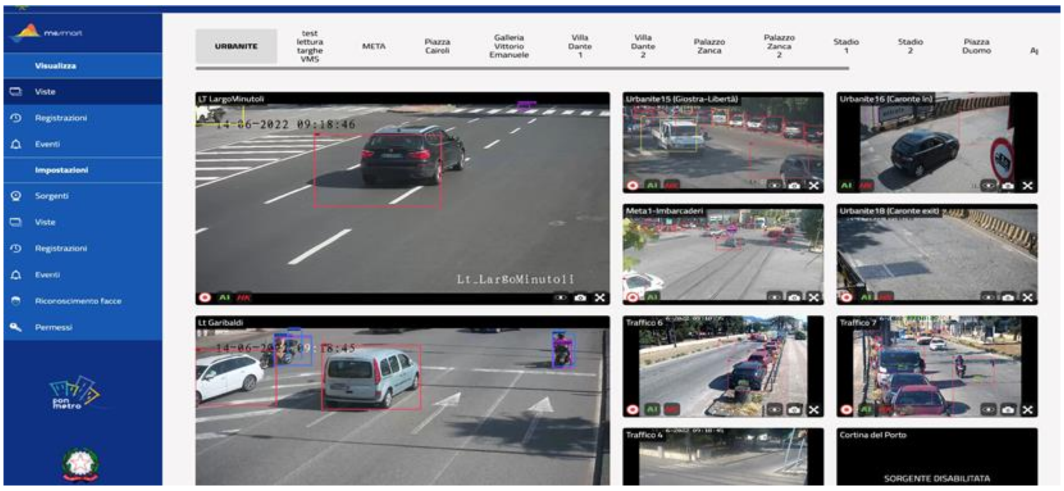


Figure 6. Data-source - Mesm@rt web platform.

- Traffic Simulation

The traffic simulation component represents the proposed policies as simulations and simulates both the baseline and proposed scenarios. The outcomes of the simulation can be analysed by the decision support system.

The machine learning models use the simulations for learning models and making connections between traffic patterns and the represented policies.

The integrated version of the traffic simulation module supports basic traffic simulation and some KPI estimations.

- Decision Support System (DSS)

As part of the Simulation module, the DSS uses multi-attribute decision analysis (MADA) methodology to analyse and compare different policy proposals.

  - KPIs were defined for each pilot and calculated based on simulation results.
  - Decision models were created for each pilot, utilising the estimated KPIs.
  - DEXi, an open source MADA toolkit was integrated.

- Exploratory Data Analysis

The exploratory data analysis component includes the libraries for prediction and regression, self-organising map, clustering, and projection. The main function is to provide interactive, and visualisation supported data exploration and analysis for presenting the user with a powerful data analysis toolkit.

- Recommendation engine

The engine provides two types of recommendations, when selecting two simulations to compare:

- General recommendation, information about the overall mobility policy quality, whether the scenario simulation is better, worse, or same, when compared to a baseline scenario.

- Specific recommendations regarding which KPIs should be improved in order to improve the mobility policy quality, achieved by performing a +/- 1/2 analysis.

2.2.3.3 *URBANITE Virtual SoPoLab (VSPL)*

The main aim of the VSPL is to enable and facilitate on-line collaboration among users, following co-creation principles. A URBANITE Forum has been integrated, supported by the digital platform for citizen participation Decidim.

2.2.3.4 *Integrated URBANITE UI*

The URBANITE UI is the main interaction and entry point between the URBANITE Platform and the end users. It is conceived as an integration framework at the UI level and acts as a wrapper of the different components of the platform that expose a web-based user interface. The aim of the URBANITE UI is to provide a homogeneous environment where the users can access and discover the functionalities offered by the platform.

From the perspective of an end-user, the URBANITE UI is composed of three main elements: a central panel that provides the user interface of the accessed functionality, a left column that provides the menu of the available functionalities, and a top bar that provides a button to resize the left columns.

More functionalities related to the customization of the dashboard and the sharing of them have been included in this version of the ecosystem.

2.2.3.5 *Identity/Authorization Management*

The Identity/Authorization Management is the tool for managing users and permissions. It offers a login page, integrated into the general URBANITE UI.

The administration console included allows to configure the realms, the registration of users and client applications, the management of their roles and the assignments to the users.

This component is based on Keycloak, an open-source Identity and Access management tool, compliant with OpenID Connect, Oauth2 and SAML.

2.2.3.6 *Controller*

The Controller component manages and executes the workflows orchestrating the different steps in implementing a data processing pipeline. For this purpose, the Controller interacts with the other components and services provided by the URBANITE platform, which are linked through the definition of "workflows". The Controller is based on Apache Airflow.

3 Delivery and usage

3.1 Package information

All components are encapsulated as docker containers.

3.2 Installation instructions

To deploy the URBANITE Ecosystem in an easy way we provide a docker compose configuration file, so that the user can install everything in one step. Alternatively, the user can also build the Docker images for each component separately, compiling the respective Docker file included in each module directory.

Installation requirements

- To have Docker tool installed in your machine and accessible.
- To have Git installed.
- We recommend running the URBANITE Ecosystem in a powerful machine, because the project is composed by 48+ Docker containers (minimum: 8 CPU; 64GB RAM; 200GB free storage depending on the datasets used).

Getting started

1. Clone the GitLab repository[2] of the project in your computer.
2. Navigate to the main root directory of the project.
3. Define the required environment variables (see .env) file, e.g.
    - export HTTPS_PORT=8443
    - export SERVER_HOST=192.168.56.1.nip.io
    - ...
4. Run in the console the command docker-compose up
   This will automatically deploy all the component containers in your *localhost* domain. This deployment may take some minutes.
5. Access to the local URBANITE UI web page in the different pilots, with a browser:
    - https://amsterdam.urbanite.esilab.org
    - https://bilbao.urbanite.esilab.org
    - https://helsinki.urbanite.esilab.org
    - https://messina.urbanite.esilab.org

3.3 User Manual

This release (v3) is the final version of the URBANITE Ecosystem and will be evaluated within the context of the activities planned in the WP6.

The instructions for using the Ecosystem are similar for every pilot. Therefore, this manual will explain the common functionalities using the Bilbao pilot as example, and the corresponding pilot for the pilot-specific functionalities.

The complete manual is included in the ANNEX II of this document.

[2] The project software will be made available in its public repository (https://git.code.tecnalia.com/urbanite/public) at the end of the project, as the licenses of the software components are still in discussion and parts of them could be proprietary.

3.4 Licensing information

The license under which this prototype is delivered is under discussion. This decision will be reported in the D7.9 deliverable where the exploitation strategy and business plan will be reported.

3.5 Download

The code is uploaded and available in the project GitLab repository:

https://git.code.tecnalia.com/urbanite/releases

This release can be checked accessing to the different deployed pilots, where a stable version is available for each of them:

https://amsterdam.urbanite.esilab.org

https://bilbao.urbanite.esilab.org

https://helsinki.urbanite.esilab.org

https://messina.urbanite.esilab.org

4 Conclusions

This document reports the status of the final version of the URBANITE Ecosystem. It contains the description of the components from a functional and technical point of view, covering all the requirements and including a complete user manual.

This URBANITE Ecosystem v3 will be used by the municipalities to evaluate the use cases defined within the context of WP6. These sessions will involve different stakeholders selected by the WP6 partners and will be supported by the technical partners.

Although this version is considered the final version of the URBANITE Ecosystem, minor changes could come up from the suggestions and comments made by the stakeholders during the evaluation sessions.

The license under the URBANITE Ecosystem, is still under study by some of the components' owners. The common license is a decision to take as part of the tasks of WP7 and will be taken by the whole consortium considering the licences of the individual components that compose this integrated version.

5 ANNEX I: Instructions for replicating the URBANITE Ecosystem in Messina

5.1 Introduction

This document is aimed to provide the instructions to install the URBANITE framework in a new environment, more concretely in the pilots or Use Cases of URBANITE. The pilots are four: Messina, Bilbao, Amsterdam, and Helsinki.

The intended reader is the technical staff in charge of the implementation of this new environment and the installation and configuration of the URBANITE Ecosystem.

As the starting point of the installation, URBANITE provides the software developed in the project, whose source files are already dockerized, the corresponding docker images and the installation scripts.

The source files and scripts are provided in the git repository of the project: https://git.code.tecnalia.com/urbanite/private/urbanite-deploy.

The docker images are stored in the docker repository of the project in Artifactory: https://artifact.tecnalia.com:443/artifactory/optima-urbanite-docker-dev-local/. Credentials must be provided by Tecnalia.

As a result, no compilation is needed to install the software. Only the docker images are downloaded and installed in the new environment.

5.1.1 Some definitions

These are the common words used trough the document, that are defined here for clarity:

Development environment: refers the environment used during the project to develop, deploy, and test the URBANITE framework.

Pilot environment, new environment refers to the environment to be set in or by the municipality that wants to install the URBANITE framework in its premises and play with it.

Data: refers to the data that is used by the different components of URBANITE. Each component can use a different type of data and managed it differently. Some components fill the URBANITE databases provided by the Storage component; some use this data directly from the Storage; others use already pre-processed data stored in files; etc.

UI: acronym of User Interface, refers to the UI of the URBANITE framework. This refers to the integrated UI that is provided to interact with the different tools that compose the URBANITE framework.

Git: refers to the git repository used during the project to support the development of URBANITE. It is actually a GitLab instance provided by Tecnalia.

Artifactory: a Docker repository that contains the Docker images of the containers of the URBANITE components. The image is an executable package of software that includes everything needed to run an application. It is provided by Tecnalia.

5.2 Pre-requirements

Hardware:

We recommend running the URBANITE framework in a powerful machine (hereinafter target server), given that the project is composed by many Docker containers (42+)

- Minimum: 8 CPU; 64Gb RAM; 200GB free storage (this highly depends on the datasets to be used)

Software:

These are the tools needed for the installation:

- Docker server and docker CLI (command line interface) installed in the target server.
- Git installed in the target server.

Network:

The target server must have internet connection in order to download the required docker images, access git repositories, etc.

The machine must have a domain (hereinafter target server domain) name (internal or external) associated to the target server. NOTE: for testing purposes nip.io can be used (https://nip.io/)

The target server may have a public IP with https port (recommended scenario) or not. These will drive to two different configurations of the traefik and the certificate (hereinafter public scenario and private scenario)

Credentials:

- URBANITE UI: urbanite/urbanite for testing purposes.
- Git: credentials/public repo

5.3 Installation instructions [MESSINA pilot]

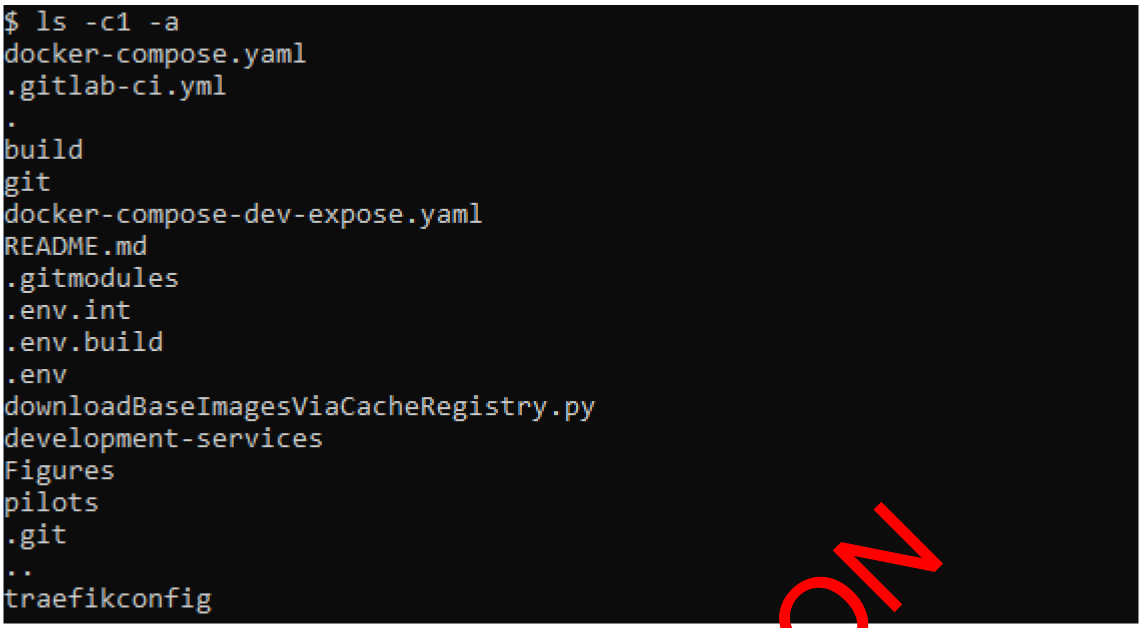
To deploy the URBANITE Ecosystem in an easy way, we provide docker-compose configuration files so that the user can install everything in one step. These files start the initialization of all the required components in a background task. Alternatively, the user can also build separately the docker images for each component, compiling the respective docker files that are included with each module.

5.3.1 Download the platform setup code

The platform setup code uses the docker compose technology to specify the components that will be deployed and the configuration that they will use.

Clone the GitLab repository https://git.code.tecnalia.com/urbanite/private/urbanite-deploy of the project in the target server. The root folder shoud look like this:

```
git clone --depth
1https://oauth:exWRbCo7MxyNy1BFD3ov@git.code.tecnalia.com/urbanite/pri
vate/urbanite-deploy.git
cd urbanite-deploy
git checkout 86a718df8bbd21ebb0a3e5170
```



```
$ ls -c1 -a
docker-compose.yaml
.gitlab-ci.yml
.
build
git
docker-compose-dev-expose.yaml
README.md
.gitmodules
.env.int
.env.build
.env
downloadBaseImagesViaCacheRegistry.py
development-services
Figures
pilots
.git
..
traefikconfig
```

Figure 7. Root folder.

5.3.2 (Optional) generate certificate for private scenario

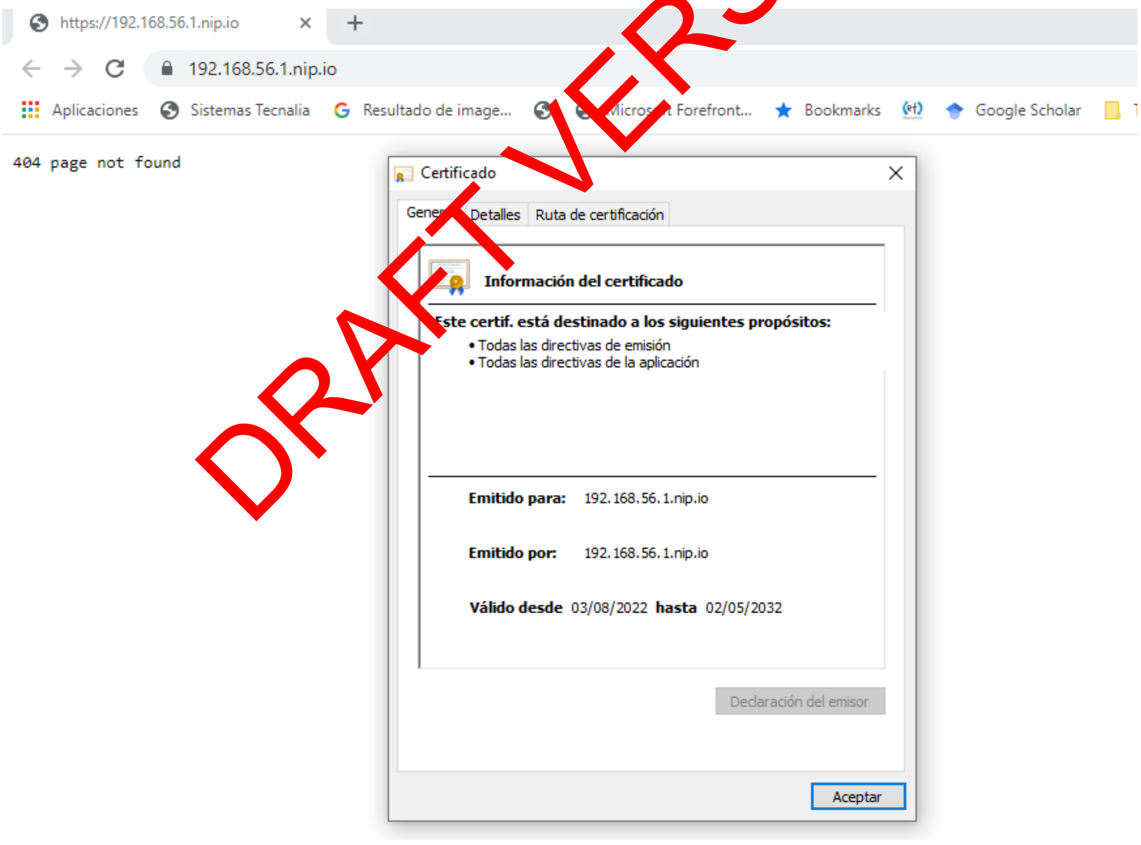
This does not apply if you have a public IP scenario where we can obtain the certificate from the let's encrypt service (https://letsencrypt.org/). Here we include an example of certificate creation.

```
echo being in the root of the cloned platform setup code
export SERVER_HOST=192.168.56.1.nip.io
export SUBJ=\
/C=IT\
/ST=Messina\
/L=Messina\
/O=Messina\
/OU=Messina\
/emailAddress=info@messina.org\
/CN=$SERVER_HOST
export SSL_DIR="$(pwd)/traefikconfig/certs"

openssl req \
-x509 \
-newkey rsa:4096 \
-sha256 \
-days 3560 \
-nodes \
-keyout $SSL_DIR/certificate.key.pem \
-out $SSL_DIR/certificate.crt.pem \
-subj "$SUBJ" \
-config <(
cat <<-EOF
[req]
distinguished_name=dn
x509_extensions=v3_req
[dn]
C=IT
ST=Messina
L=Messina
O=Messina
OU=Messina
emailAddress=info@messina.org
CN=$SERVER_HOST
[v3_req]
subjectAltName=@alt_names
[ alt_names ]
DNS.1=$SERVER_HOST
DNS.2=*.$SERVER_HOST
EOF
)
openssl x509 -in "$SSL_DIR/certificate.crt.pem" -text -noout
```

Note: if you want to trust in this autogenerated certificate in windows

```
certutil -addstore -user -f "Root"
git\deploy\traefikconfig\certs\certificate.crt.pem
```



404 page not found

Información del certificado

Este certif. está destinado a los siguientes propósitos:
- Todas las directivas de emisión
- Todas las directivas de la aplicación

Emitido para: 192.168.56.1.nip.io

Emitido por: 192.168.56.1.nip.io

Válido desde 03/08/2022 **hasta** 02/05/2032

Figure 8. Details of the certificate.

5.3.3 Config the. env

This file provides the environment variables needed by URBANITE. Those must be injected in the machine's OS. The deployment process gets the values from this file if they are not manually injected in the OS as environment variables (using exports).

The most important one to be adapted is the SERVER_HOST, that should match the domain name of the target server. In the private scenario the SERVER_HOST must be the same as the one used to generate the certificate.

At the time of writing, these are the variable set (the actual values that can be adapted to the new environment are also shown):

```
cat << 'EOF' > .env
SERVER_HOST=192.168.56.1.nip.io
HTTPS_PORT=443
TZ=Madrid

BIKETRAJECTORIESFRONTEND=
URBANITEUIPILOT=MESSINA
NOISECOMPUTATIONCITY=MESSINA
NOISECOMPUTATIONCITYIMAGE=-messina
TRAFFICPREDICTIONDBIMAGE=
TRAFFICPREDICTIONIMAGE=
BIKEANALISISCITY=Bilbao
PILOT=MESSINA

# these password are url sensible therefore do not use url special characters
i.e. [@:/]
PASSWORD_MYSQL_ROOT=admin
PASSWORD_MYSQL_PRESTO=admin
PASSWORD_POSTGRES_KEYCLOAK=admin
PASSWORD_POSTGRES_DASHBOARD=admin
PASSWORD_KEYCLOAK=admin
HARVESTER_MESSINA_COMUNE_TOKEN=admin
PASSWORD_POSTGRES_SIMULATION=admin
PIVEAU_CLUSTER_CONFIG={"pipeRepositories":{"system":{"uri":"https://git.code.t
ecnalia.com/urbanite/private/wp3-data-management/harvester/harvesting-
pipelines.git","username":"gitlab-
user","token":"UcC_[illegible]AMx[illegible]nZ47CEyQbt","branch":"master"}}}
PASSWORD_IFM_HASH=$$apr1$$krW7sm24$$8tJDxOywuuTfXRgyauAb5/
DOCKER_REGISTRY_PREFIX=optima-urbanite-docker-dev.artifact.tecnalia.com/
PROJECT_NAME=urbanite
COMPOSE_PROJECT_VERSION=latest
INIT_DATA_URL_DOWNLOAD_LINK=https://portal.ijs.si/nextcloud/s/m8zkQGAwGipS329/
download?path=%2F&files=messina
SIMULATION_CITY=messina
    # TRAFFICPREDICTIONDBIMAGE: -messina
    # TRAFFICPREDICTIONIMAGE: -messina
    TRAFFICPREDICTIONCITY: Messina
    #BIKETRAJECTORIESFRONTEND: Messina

#
https://docs.docker.com/compose/reference/envvars/#compose_file#compose_projec
t_name
# these are docker-compose related environment variables
COMPOSE_PROJECT_NAME=urbanite-messina-master

TRAEFIK_CONSTRAINTS="Label(`com.docker.compose.project`, `urbanite-messina-
master`)"
# With public IP -> public scenario
# COMPOSE_FILE=docker-compose.yaml:pilots/messina/docker-
compose.yaml:development-services/docker-compose-ifm.yaml:development-
```

```
services/docker-compose-expose.yaml:development-services/docker-compose-
redirect-http.yaml:development-services/docker-compose-traefik-acme.yaml

# Without public IP -> private scenario
COMPOSE_FILE=docker-compose.yaml:pilots/messina/docker-
compose.yaml:development-services/docker-compose-ifm.yaml:development-
services/docker-compose-expose.yaml:development-services/docker-compose-
redirect-http.yaml:development-services/docker-compose-traefik-certs.yaml

EOF
```

5.3.4 Deploy the platform

Deploy the platform following setup code and the customised .env.

```
docker login optima-urbanite-docker-dev.artifact.tecnalia.com -u
urbanite.fordevelopers@gmail.com -p
AKCp8mZwMpVc8rQpW774AQTBs5c4rMjAzGhr8gm2y7AUr1DCcXkPSf5M3wopqKvf2dBbK1
jtX
docker-compose up -d
```

5.3.5 Using data from demo pilots

The database servers used in the storage components (MySQL, MongoDB, OpenTSDB) are installed by the installation process described before.

If some of the data already in these databases in the pilot demo servers (https://messina.urbanite.esilab.org/) is required by a component of a pilot site, a manual procedure has to be followed to copy the data from the development environment and insert it in the databases created in the pilot environment. In that case the data must be requested to TECNALIA that will provide a tar.gz of the data volume.

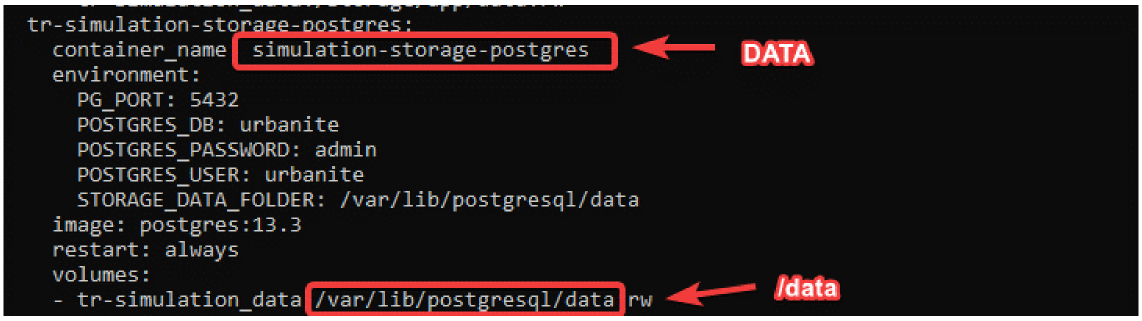
This table list the datasets that may require manual data insertion:

- Traffic Simulation
- Noise Computation
- Traffic Evolution
- Weekly Traffic Flows
- LPT Critical Areas
- Dashboard Controller
- Urbanite UI
- Data Catalogue
- Data Storage

Procedure to back up the volume to be executed by TECNALIA at the pilot demo server:

```
docker run --rm --volumes-from DATA -v $(pwd):/backup busybox tar -
czvf /backup/backup.tar.gz /data
```

being DATA the name of the container see the list above and "/data" the appropriate persistence folder inside that container. Both values (DATA and /data) can be seen in the docker-compose config.



```
tr-simulation-storage-postgres:
  container_name  simulation-storage-postgres
  environment:
    PG_PORT: 5432
    POSTGRES_DB: urbanite
    POSTGRES_PASSWORD: admin
    POSTGRES_USER: urbanite
    STORAGE_DATA_FOLDER: /var/lib/postgresql/data
  image: postgres:13.3
  restart: always
  volumes:
  - tr-simulation_data /var/lib/postgresql/data rw
```

Figure 9. Details of DATA and /data in the docker-compose configuration file.

Procedure to restore the data to be executed at the target server

```
docker stop DATA
docker run --rm --volumes-from DATA -v $(pwd):/backup busybox tar xzvf
/backup/backup.tar.gz -C /data
docker start DATA
```

NOTE: consider that the data restored may contain domain related information that will be pointing to the pilot target domain messina.urbanite.esilab.org

5.3.6 Particular configuration of components

There are components that require additional actions to be configured, actions that are not covered by the docker-compose up.

5.3.6.1 Idmanager/Keycloak

After the deployment it is necessary to define the users to be able to enter the urbanite ui (https://target_server_domain/ i.e. https://192.168.56.1.nip.io/) in order to define the users, access the keycloak ui i.e. (https://target_server_domain/auth i.e. https://192.168.56.1.nip.io/auth login with admin/Pa55w0rd). You can also use the URBANITE customized Keycloak (https://messina.urbanite.esilab.org/auth)

NOTE: if you have problems accessing the keycloak try with the "incognito" mode of your browser to get rid of the cookies.

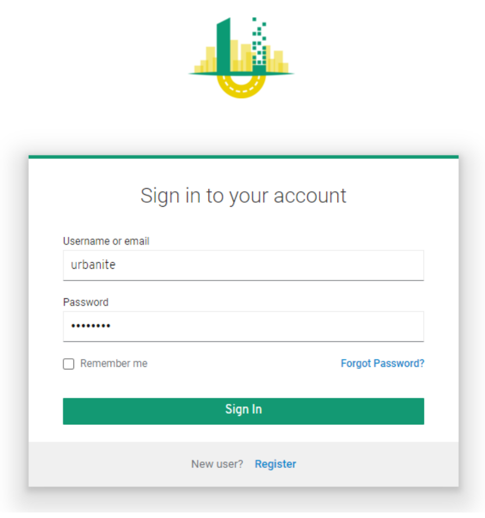
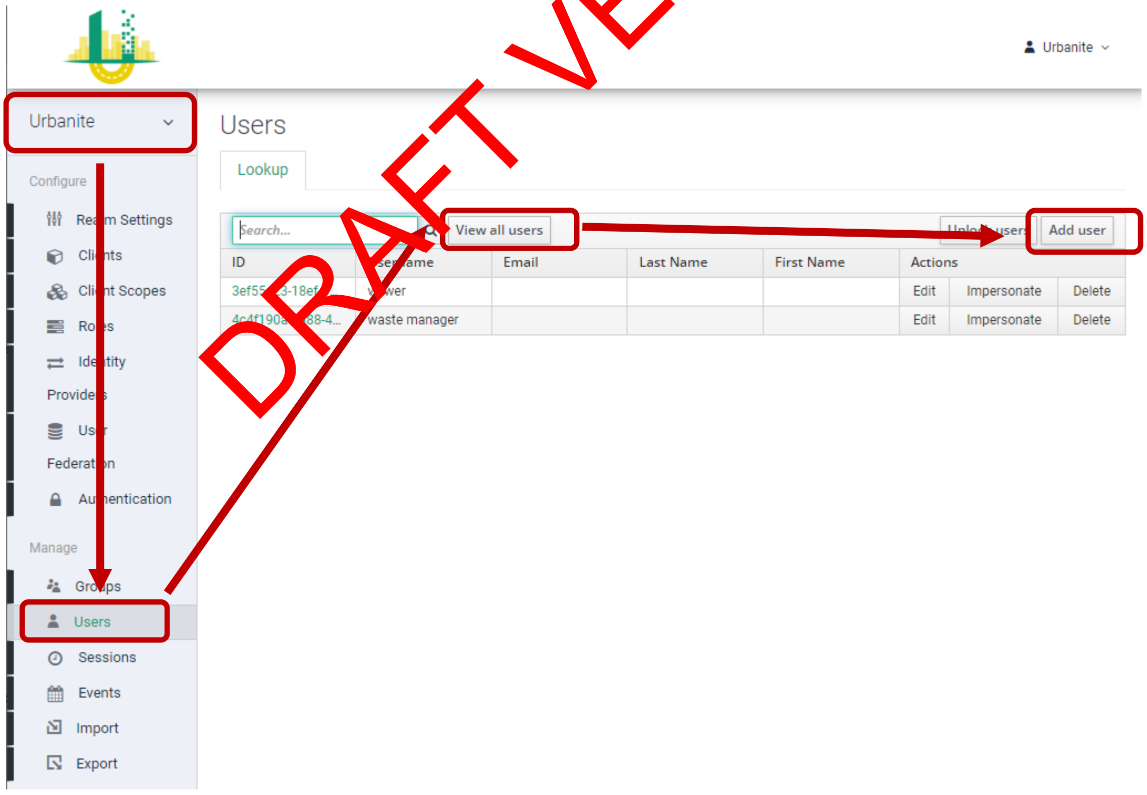


Figure 10. URBANITE Keycloak login page.

There:

- Make sure you are in urbanite realm
- Got to users
- Click view to make sure that you are not adding an existing user
- Click add user



ID	Username	Email	Last Name	First Name	Actions		
3ef55…3-18ef	viewer				Edit	Impersonate	Delete
4c4f190a…88-4…	waste manager				Edit	Impersonate	Delete

Figure 11. URBANITE Keycloak adding users.

5.3.6.2 Harvesting tools

The **Harvesters** are processes that gather data from public/private data sources and inject them in the Storage component. Every harvester has been implemented following common rules, whose pipeline is configured by a json file (that is now stored in the project git), and finally is registered in and started by the **Scheduler**. Once started, the Harvester can continue gathering data until the specified limits are met or indefinitely as new data is encountered in the data source.

To start a pipeline using the Harvesters installed in the new environment, it has to be scheduled manually by creating a so-called trigger. This can be done either via a TUI or the API. Both interfaces can show all available pipelines and triggers and offer the ability to create new triggers. The TUI is available at https://target_server_domain/harvester/scheduler/shell.html, i.e. https://192.168.56.1.nip.io/harvester/scheduler/shell.html. The API description with explanations and examplkes for different triggers can be accessed at https://target_server_domain/harvester/scheduler/api, i.e. https://192.168.56.1.nip.io/harvester/scheduler/api.

Available commands for the TUI are:

Command	**Description**
`pipes`	List available pipes.
`show <pipe-id>`	Show specific pipe.
`trigger <pipe-id>`	Show triggers of specific pipe.
`launch <pipe-id>`	Start specific pipe immediately.

NOTE: **PIVEAU_CLUSTER_CONFIG** is an environment variable in .env file, that is used by the Scheduler component. Its value points now to a repository (*harvesting-pipelines.git*) in *git.code.tecnalia.com*, that stores the pipeline files. Although this will continue working this way, this variable should be ideally configurable and point to a repository somewhere in the new environment, which contains all pipeline files, that will be used.

5.4 Access to the UI

Once all is installed and configured, the URBANITE UI platform should be accessible with a web browser in https://target_server_domain i.e., https://192.168.56.1.nip.io

6 ANNEX II: URBANITE Ecosystem. User manual.

The entry point to the Ecosystem is the URBANITE UI, available at these URLs:

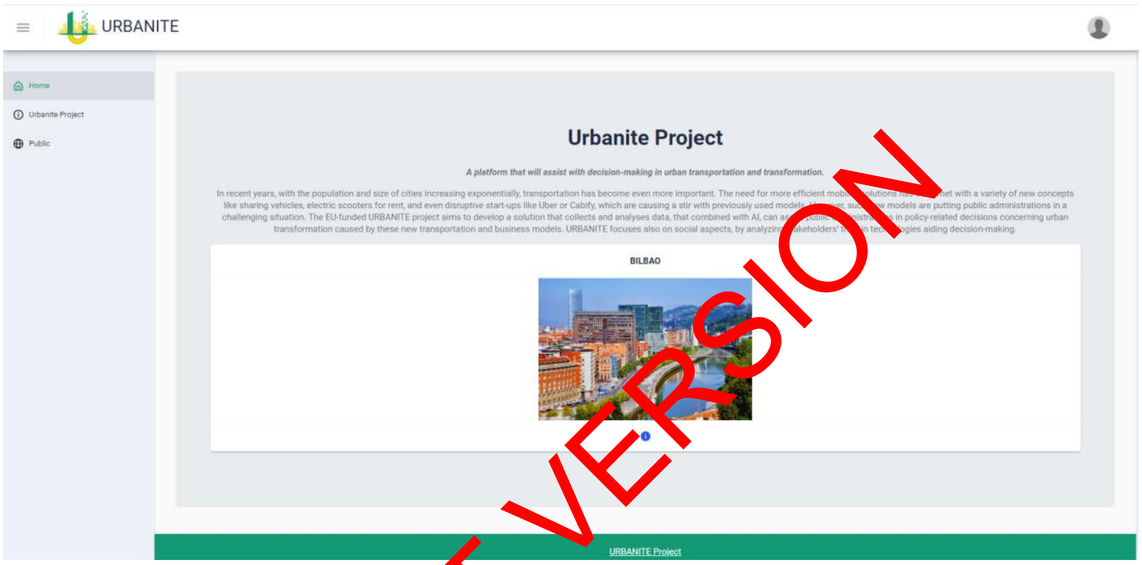
https://amsterdam.urbanite.esilab.org/

https://bilbao.urbanite.esilab.org/

https://helsinki.urbanite.esilab.org/

https://messina.urbanite.esilab.org/

and the Interface presented is the public view of the URBANITE Ecosystem:



Urbanite Project

A platform that will assist with decision-making in urban transportation and transformation.

In recent years, with the population and size of cities increasing exponentially, transportation has become even more important. The need for more efficient mobi[illegible]olutions [illegible]et with a variety of new concepts like sharing vehicles, electric scooters for rent, and even disruptive start-ups like Uber or Cabify, which are causing a stir with previously used models. [illegible] su[illegible]w models are putting public administrations in a challenging situation. The EU-funded URBANITE project aims to develop a solution that collects and analyses data, that combined with AI, can as[illegible] public [illegible]istra[illegible]s in policy-related decisions concerning urban transformation caused by these new transportation and business models. URBANITE focuses also on social aspects, by analyzing [illegible]akeholders' t[illegible] in tec[illegible]ogies aiding decision-making.

Figure 12 URBANITE public page

The user needs to introduce the credentials (urbanite, urbanite) to enter the Ecosystem by accessing the icon on the right top corner of the screen.

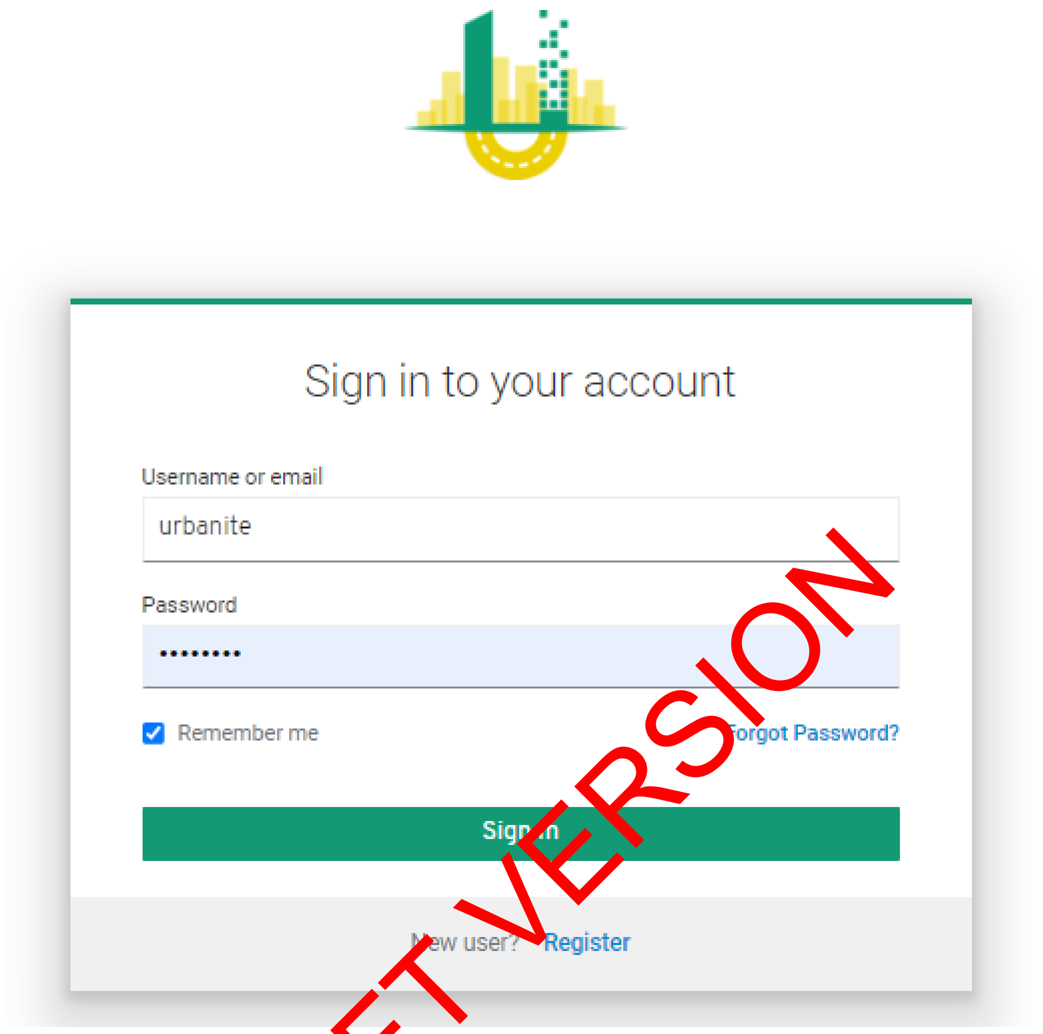
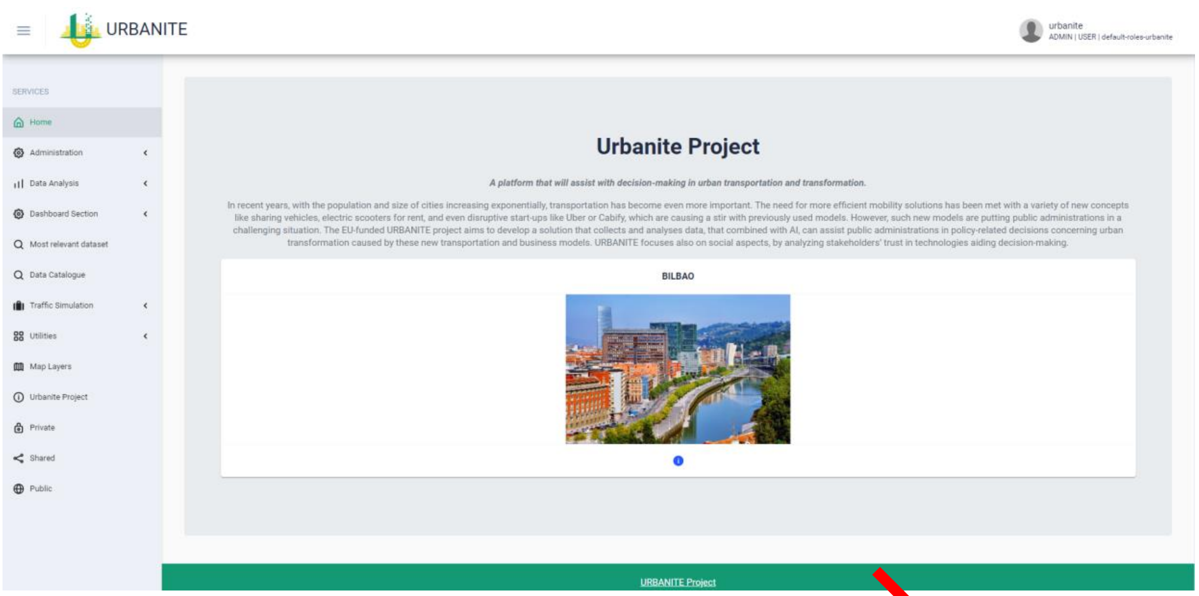


Figure 13.URBANITE login page

The URBANITE UI presents two main sections: a central panel that provides the information of the selected functionality, and a left column with the menu of the available functionalities.

The main features provided are placed in the left side of the page:



Urbanite Project

A platform that will assist with decision-making in urban transportation and transformation.

In recent years, with the population and size of cities increasing exponentially, transportation has become even more important. The need for more efficient mobility solutions has been met with a variety of new concepts like sharing vehicles, electric scooters for rent, and even disruptive start-ups like Uber or Cabify, which are causing a stir with previously used models. However, such new models are putting public administrations in a challenging situation. The EU-funded URBANITE project aims to develop a solution that collects and analyses data, that combined with AI, can assist public administrations in policy-related decisions concerning urban transformation caused by these new transportation and business models. URBANITE focuses also on social aspects, by analyzing stakeholders' trust in technologies aiding decision-making.

BILBAO

Figure 14.URBANITE UI home page

There are two kinds of functionalities, the general ones, more related to the administration ad utilisation of the Ecosystem, and the core functionalities in URBANITE where the simulations, analysis and recommendations are placed.

The general functionalities are:

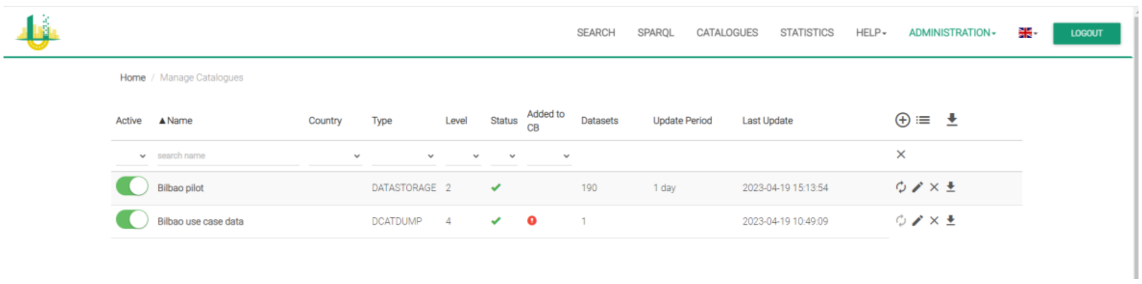
- Administration
- Dashboard section
- Most relevant Datasets
- Data Catalogue
- Utilities
- Map layers
- Urbanite Project
- Private
- Shared
- Public

The core URBANITE functions are presented under the Data Analysis and Traffic Simulation options.

6.1 Administration

The administration section gathers two options related to the general aspects of the data and the user management: Data catalogue Administration and IDM Administration.

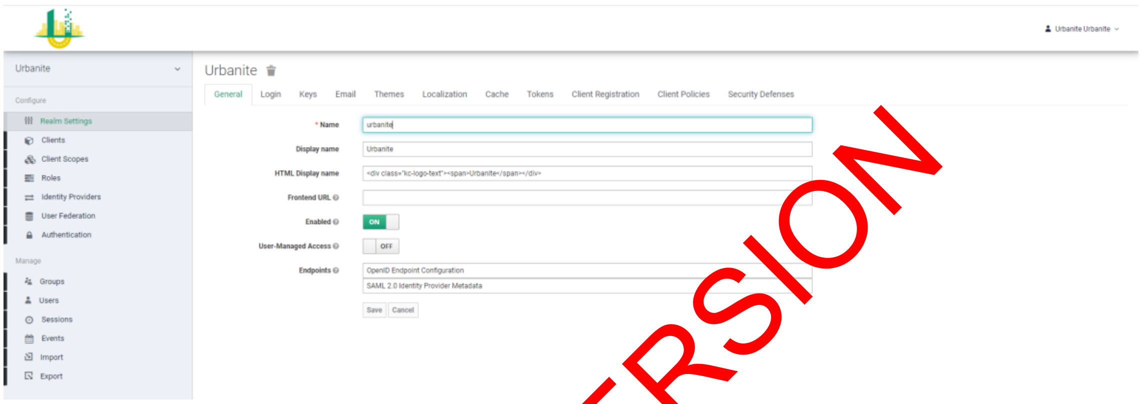
The Data Catalogue administration allows the creation of new data catalogues, the activation of them and a series of utilities related to the datasets.



Active	Name	Country	Type	Level	Status	Added to CB	Datasets	Update Period	Last Update
ON	Bilbao pilot		DATASTORAGE	2	✓		190	1 day	2023-04-19 15:13:54
ON	Bilbao use case data		DCATDUMP	4	✓	●	1		2023-04-19 10:49:09

Figure 15.Administration: Data catalogue Administration.

The IDM Administration is linked to a customized Keycloak version in order to manage the users and the access to the URBANITE Ecosystem.



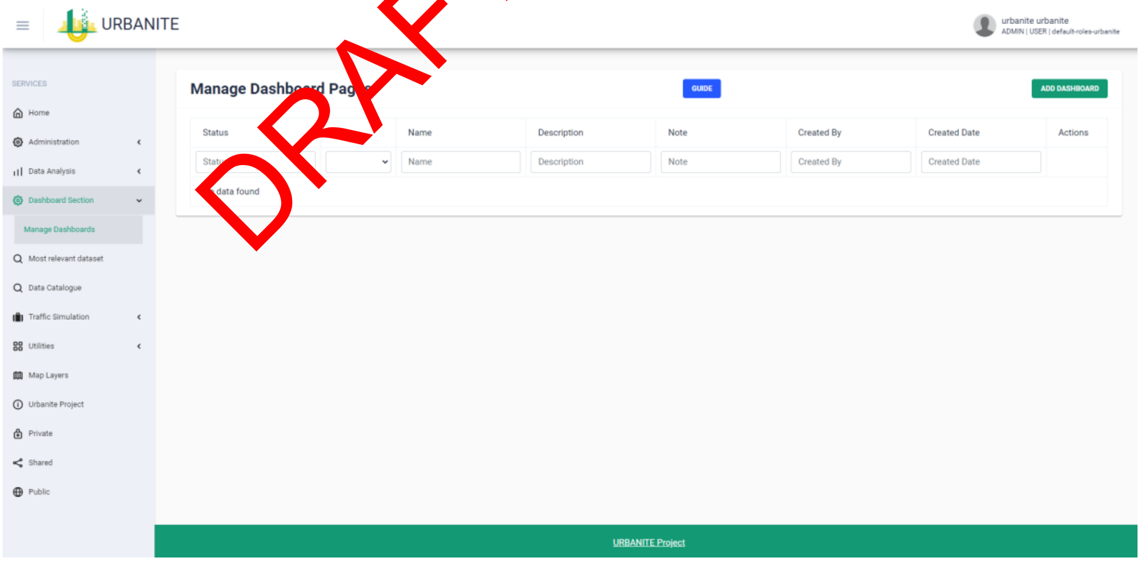
Urbanite

* Name	urbanite
Display name	Urbanite
HTML Display name	<div class="kc-logo-text"><span>Urbanite</span></div>
Frontend URL	
Enabled	ON
User-Managed Access	OFF
Endpoints	OpenID Endpoint Configuration
	SAML 2.0 Identity Provider Metadata

Figure 16.Administration: IDM Administration.

6.2 Dashboard section

In this section the user can manage dashboards and can customize them.



Manage Dashboard Pages

Status	Name	Description	Note	Created By	Created Date	Actions
No data found						

Figure 17.Manage Dashboard Pages.

The management consists of different actions related to the dashboards:

- ✓ Edit the dashboard information
- ✓ Delete the dashboard

- ✓ Clone the dashboard
- ✓ Preview the dashboard
- ✓ Share the dashboard
- ✓ Edit the dashboard content

Selecting the ADD DASHBOARD button, a new page is presented:

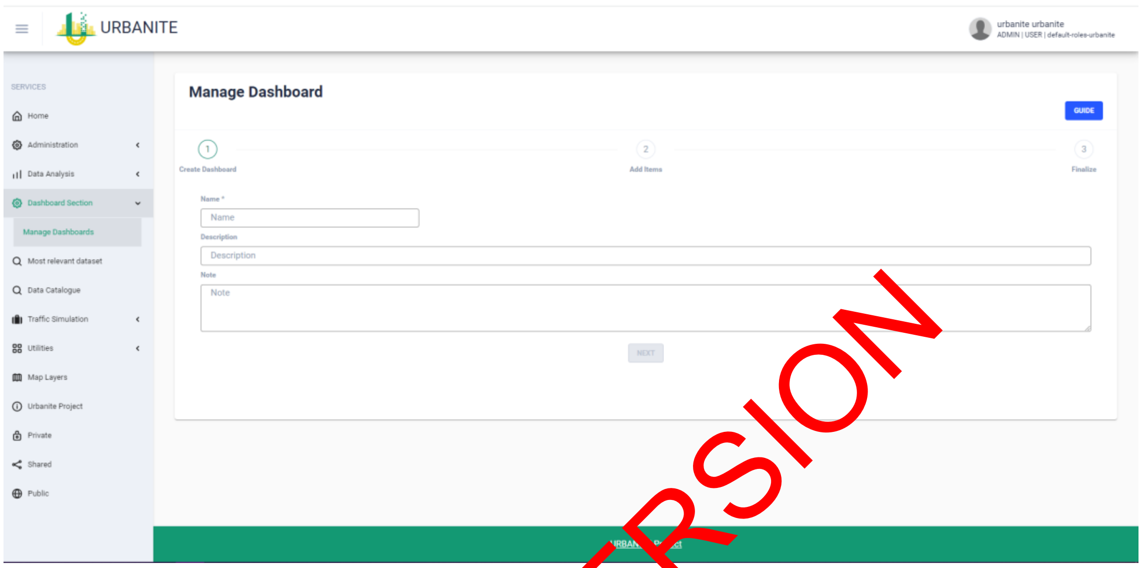


Figure 18. Manage Dashboard

The fields to be fulfilled are Name, description and Notes. The fields marked with * are mandatory.

Along all the process, a button guide is available at the top of the page where a description of the steps is shown.

The next step is for including components in the Dashboard.

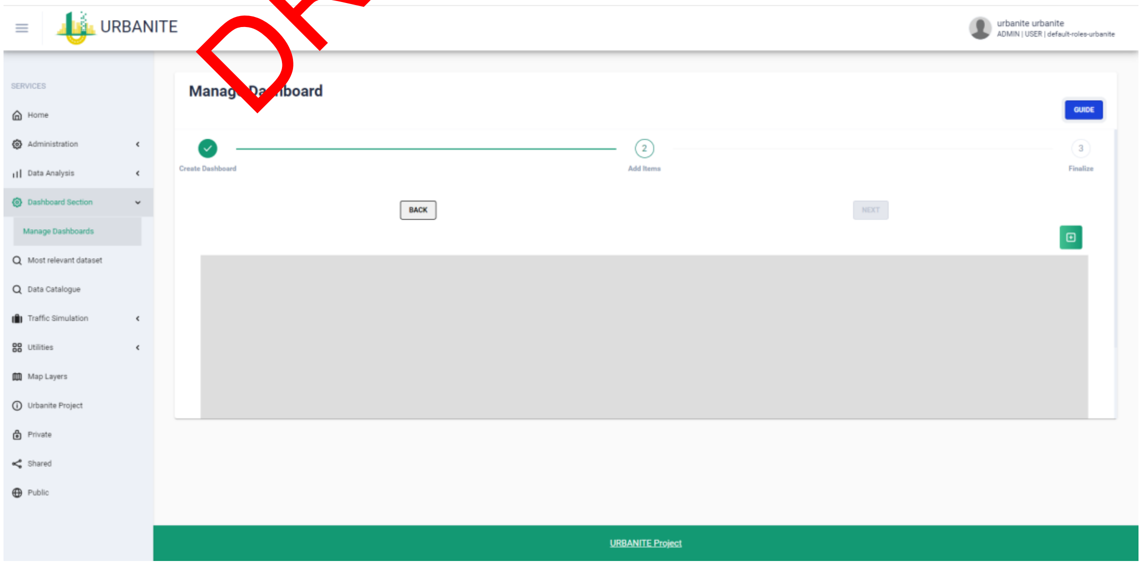
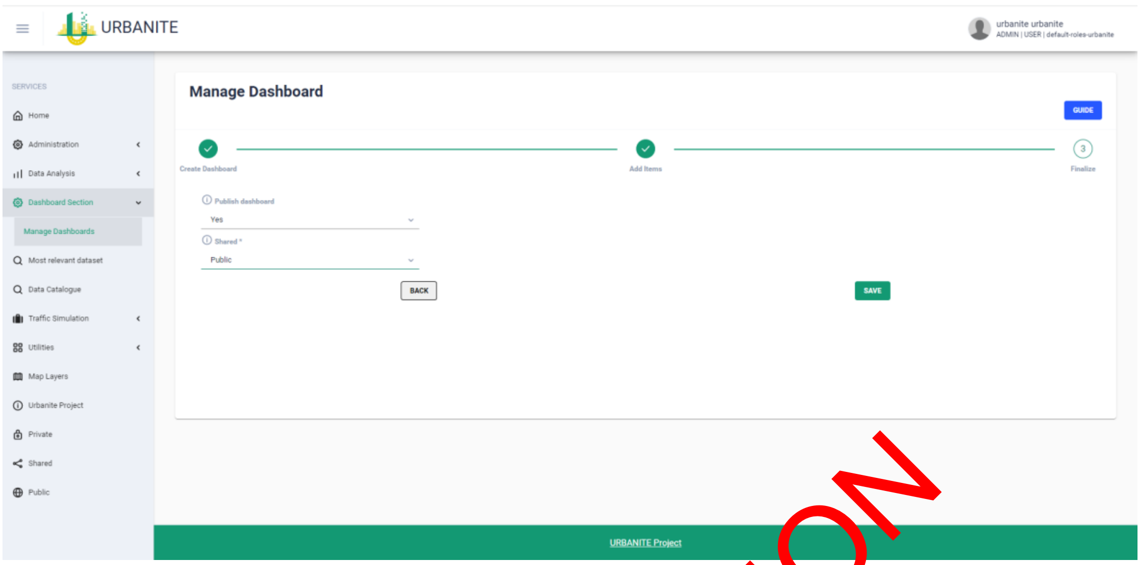


Figure 19. Manage Dashboard

The third step allows to set the dashboard as published or as a draft. If you set it as published you can choose to make it: private, public (available even for not registered user) or shared with someone or a group or a role (a window will appear in the next step to let you choose).



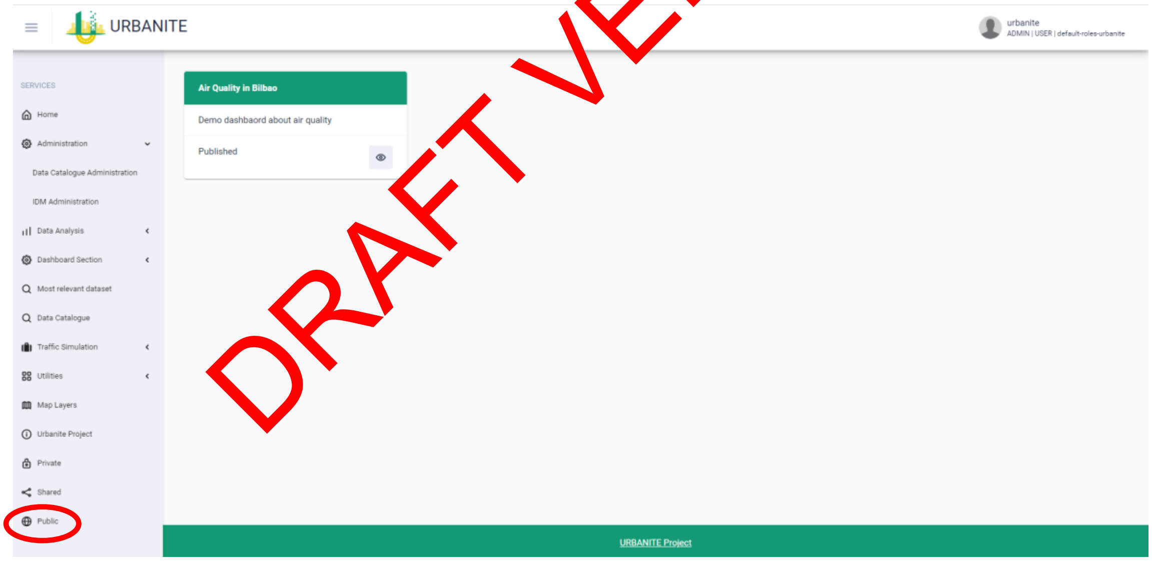
Manage Dashboard

Publish dashboard: Yes

Shared *: Public

Figure 20. Manage Dashboard

Once the dashboard is created, it will be visible in the corresponding section, depending on the characteristics of the new dashboard.



Air Quality in Bilbao

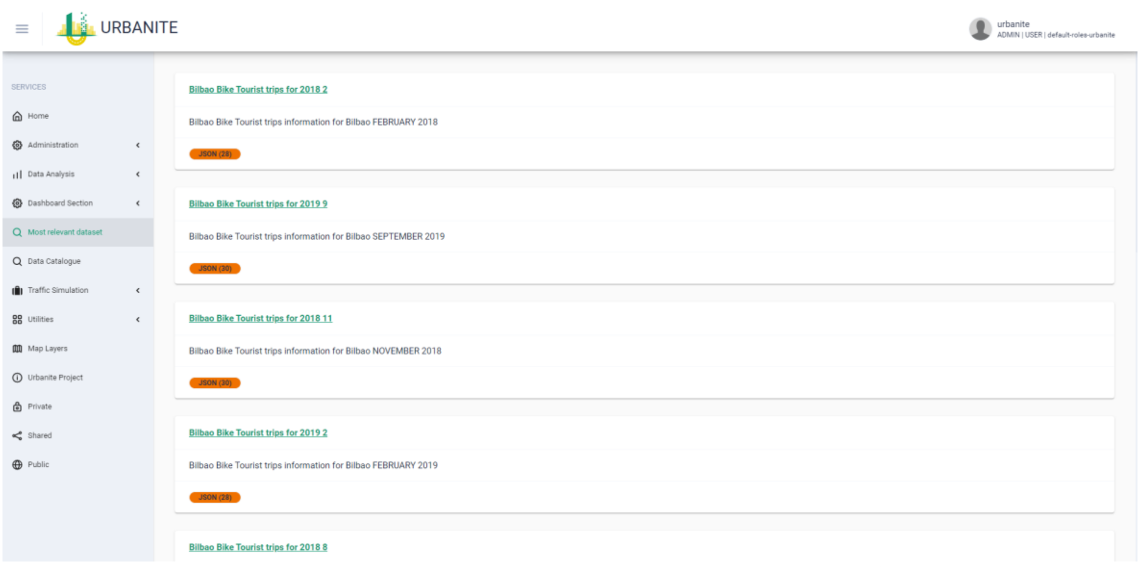
Demo dashbaord about air quality

Published

Figure 21. Public Dashboard

6.3 Most relevant Datasets

This option shows a list of the datasets following the configuration assigned when creating.



Bilbao Bike Tourist trips for 2018 2

Bilbao Bike Tourist trips information for Bilbao FEBRUARY 2018

JSON (28)

Bilbao Bike Tourist trips for 2019 9

Bilbao Bike Tourist trips information for Bilbao SEPTEMBER 2019

JSON (30)

Bilbao Bike Tourist trips for 2018 11

Bilbao Bike Tourist trips information for Bilbao NOVEMBER 2018

JSON (30)

Bilbao Bike Tourist trips for 2019 2

Bilbao Bike Tourist trips information for Bilbao FEBRUARY 2019

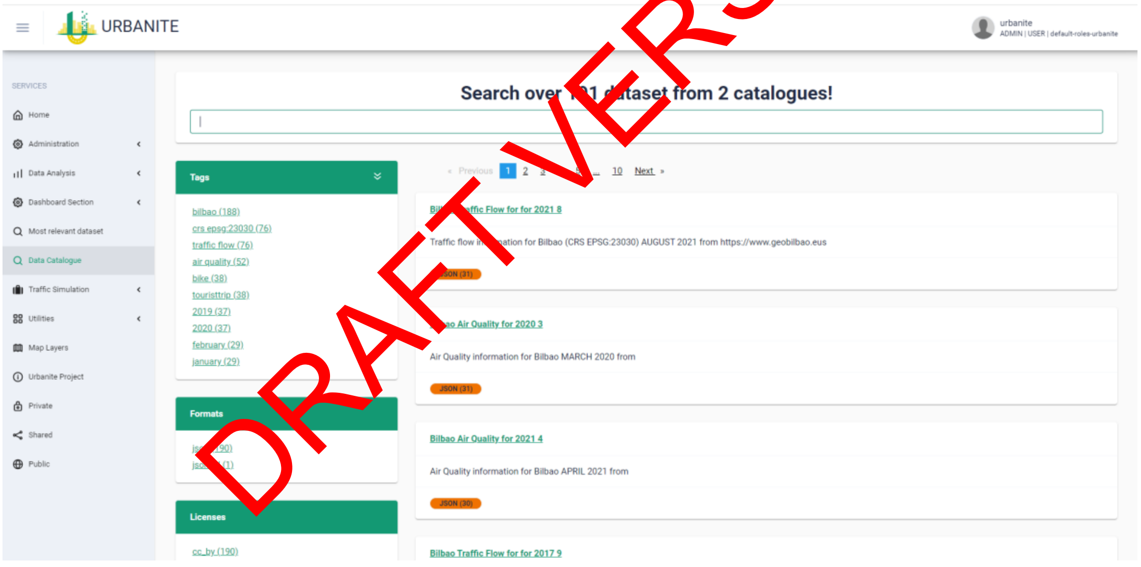
JSON (28)

Bilbao Bike Tourist trips for 2018 8

Figure 22. Most relevant dataset

6.4 Data Catalogue

This option presents to the user the datasets available, and it allows to search among them selecting a predefined criterion.



Search over [illegible] dataset from 2 catalogues!

Tags

bilbao (188)
crs epsg:23030 (76)
traffic flow (76)
air quality (52)
bike (38)
touristtrip (38)
2019 (37)
2020 (37)
february (29)
january (29)

Formats

[illegible] (190)
[illegible] (1)

Licenses

cc_by (190)

[illegible]affic Flow for for 2021 8

Traffic flow [illegible]ation for Bilbao (CRS EPSG:23030) AUGUST 2021 from https://www.geobilbao.eus

[illegible]SON (31)

[illegible]o Air Quality for 2020 3

Air Quality information for Bilbao MARCH 2020 from

JSON (31)

Bilbao Air Quality for 2021 4

Air Quality information for Bilbao APRIL 2021 from

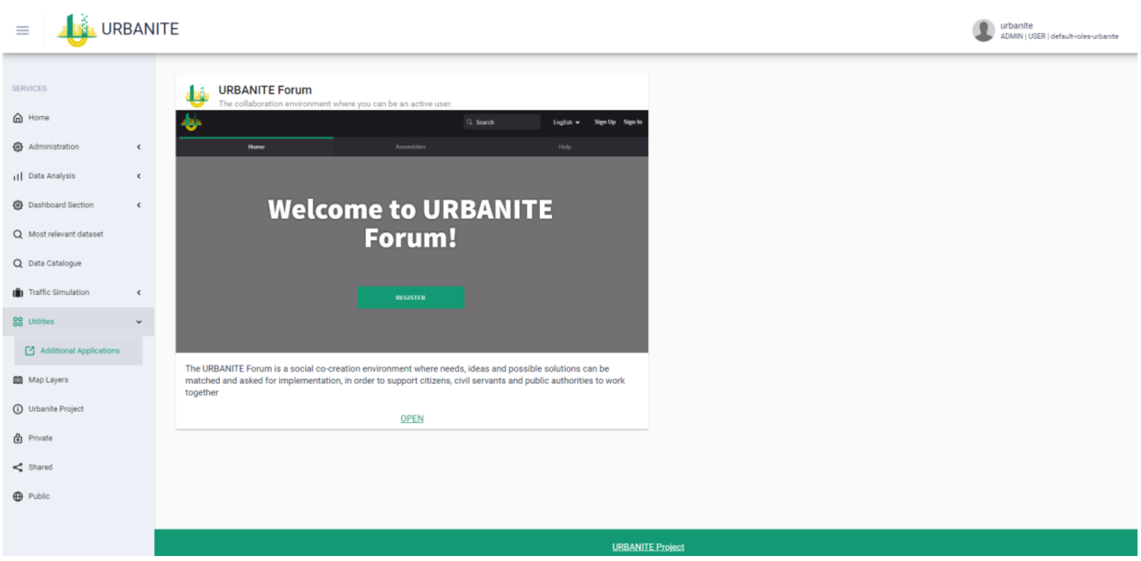
JSON (30)

Bilbao Traffic Flow for for 2017 9

Figure 23.Bilbao Data Catalogue

6.5 Utilities

The URBANITE Forum is included as an additional application. The URBANITE GUI allows to integrate external tools to be accessible to the users from the URBANITE GUI.



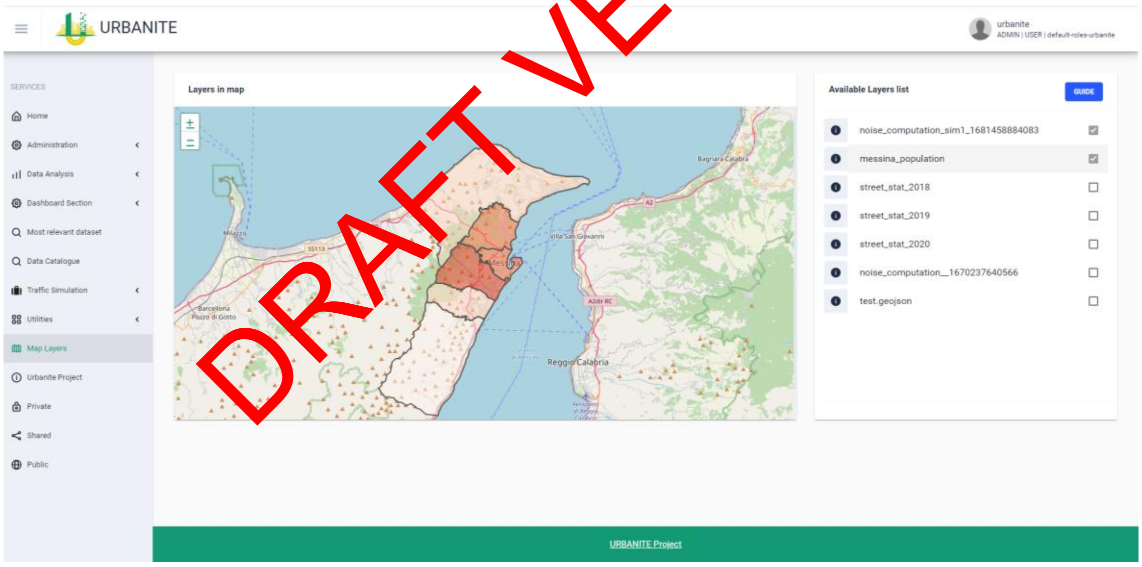
The URBANITE Forum is a social co-creation environment where needs, ideas and possible solutions can be matched and asked for implementation, in order to support citizens, civil servants and public authorities to work together

Figure 24.Utilities: URBANITE Forum as Additional Application

6.6 Map Layers

In order to obtain information about a specific layer you can click the info button beside the layer you're interested at.

If you want to visualize a specific layer, click on it. If you want to hide a certain layer that is already selected, click on it again to deselect it.

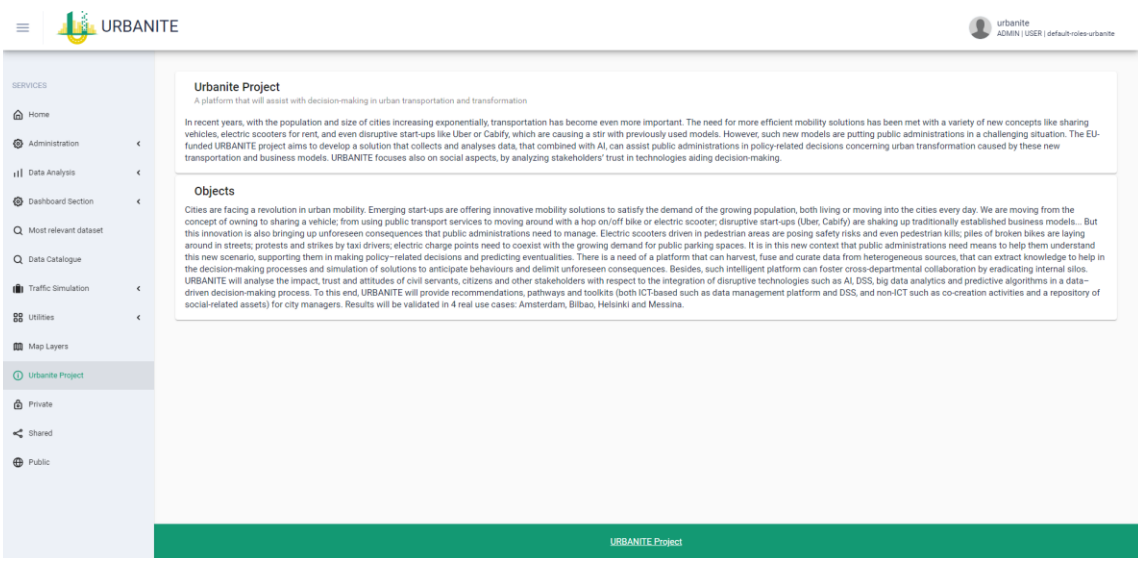


●	noise_computation_sim1_1681458884083	✓
●	messina_population	✓
●	street_stat_2018	☐
●	street_stat_2019	☐
●	street_stat_2020	☐
●	noise_computation__1670237640566	☐
●	test.geojson	☐

Figure 25. Map layers

6.7 Urbanite Project.

This option shows general information of the project and a link to its public page (https://urbanite-project.eu/)



Urbanite Project

A platform that will assist with decision-making in urban transportation and transformation

In recent years, with the population and size of cities increasing exponentially, transportation has become even more important. The need for more efficient mobility solutions has been met with a variety of new concepts like sharing vehicles, electric scooters for rent, and even disruptive start-ups like Uber or Cabify, which are causing a stir with previously used models. However, such new models are putting public administrations in a challenging situation. The EU-funded URBANITE project aims to develop a solution that collects and analyses data, that combined with AI, can assist public administrations in policy-related decisions concerning urban transformation caused by these new transportation and business models. URBANITE focuses also on social aspects, by analyzing stakeholders' trust in technologies aiding decision-making.

Objects

Cities are facing a revolution in urban mobility. Emerging start-ups are offering innovative mobility solutions to satisfy the demand of the growing population, both living or moving into the cities every day. We are moving from the concept of owning to sharing a vehicle; from using public transport services to moving around with a hop on/off bike or electric scooter; disruptive start-ups (Uber, Cabify) are shaking up traditionally established business models... But this innovation is also bringing up unforeseen consequences that public administrations need to manage. Electric scooters driven in pedestrian areas are posing safety risks and even pedestrian kills; piles of broken bikes are laying around in streets; protests and strikes by taxi drivers; electric charge points need to coexist with the growing demand for public parking spaces. It is in this new context that public administrations need means to help them understand this new scenario, supporting them in making policy–related decisions and predicting eventualities. There is a need of a platform that can harvest, fuse and curate data from heterogeneous sources, that can extract knowledge to help in the decision-making processes and simulation of solutions to anticipate behaviours and delimit unforeseen consequences. Besides, such intelligent platform can foster cross-departmental collaboration by eradicating internal silos. URBANITE will analyse the impact, trust and attitudes of civil servants, citizens and other stakeholders with respect to the integration of disruptive technologies such as AI, DSS, big data analytics and predictive algorithms in a data–driven decision-making process. To this end, URBANITE will provide recommendations, pathways and toolkits (both ICT-based such as data management platform and DSS, and non-ICT such as co-creation activities and a repository of social-related assets) for city managers. Results will be validated in 4 real use cases: Amsterdam, Bilbao, Helsinki and Messina.

Figure 26. URBANITE Project

6.8 Private, Shared and Public

Options in the menu related to the Dashboard management function.

6.9 Data Analysis

Bilbao pilot provides most of the modules developed for data analytics. All of them include a help button with information about how to use the specific analysis in order to obtain the desired results.

6.9.1 Traffic prediction

The component is divided in two different columns: the column of the left-hand side has several controls to visualize data including a menu, two tables and a calendar, and the column of the right-hand side has a map and area to visualize charts. Now we describe these components. The controls of the left column include the following:

- **Labels Switch:** On the top left side, its purpose is to show in the map the id of each sensor.
- **Filter Switch:** On the top centre side, its purpose is to show, in the models table, only the models of the sensor selected in the map.
- **Models Table:** The models table show information about the characteristics and the status of the models that are or have been trained. In the bottom left side of the table there are two buttons that allow create new models and delete them. In addition, on the bottom right side there is a button to reload the data of the table. This last button is important because immediately after you submit the creation of the model the new model does NOT show up in the table, the user should press the reload button to be able to see it.
- **Calendar:** The calendar is used to choose the day to be predicted, or the starting day of the period to be predicted. This is used once the models have been trained, i.e., their status is zero. By default, the selected day corresponds to the current day. In addition, in the calendar. Notice that is a prediction that involves weather is required only few days in the future are susceptible to be predicted. In that particular case the not allowed days are grey out showing that it cannot be selected.

- **Loop Table:** Shows statistical information about the data from the Loop sensor chosen. Below this table there are two buttons:
    - **Visualize:** Allows to visualize the data associated with the selected sensor to be used for training, be aware that visualizing all these data takes generally a very long time.
    - **Download:** Allows to download to your computer the result of a prediction that is shown in the Chart area.

On the map of the right-hand side the sensors with data available for the use case selected are show with blue dots. In order to start with the process to create a new model a sensor has to be chosen by clicking on this blue dot. Once the loop sensor is selected the available data is shown in the graph area below the map. The historic traffic stored within the platform is shown in blue, the temperature in green and the precipitation in cyan (for the time being the temperature and the precipitation are not used for the training of models). In addition, other information related the flow measurements are shown in the table on the left-hand side, the aforementioned Loop Table.

Create/Train Model

If a loop sensor is selected (by clicking on it in the map) the training of a model can be initiated by pressing the **New Model** button located inside the model table. This opens up a new window. In the case no sensor is selected the process will involve all the sensors and will take a long time. Once the loop sensor is selected information related the flow measurements are shown in the table on the bottom left-hand side. When the **New Model** button is pressed there is a window opened, in the window the following information can be seen/specified:

- **id:** the identification label for the loop sensor chosen in the map (in the case one loop is selected on the map).
- **Num Features:** number of features considered to create the model:
    - **1**: Considers only the time of the day, Hour of the Day (HoD). We call it the *slot* of the day, this corresponds to the time interval of the day, considering the measuring period of the given sensor.
    - **2**: In addition to the previous considers the day of the week (DoW).
    - **3**: In addition to the previous considers if the given day is a school day and/or a Holiday or not (SaH).
    - **4**: In addition to the previous considers the meteorological variables (MET).
    - **5**: In addition to the previous considers events.

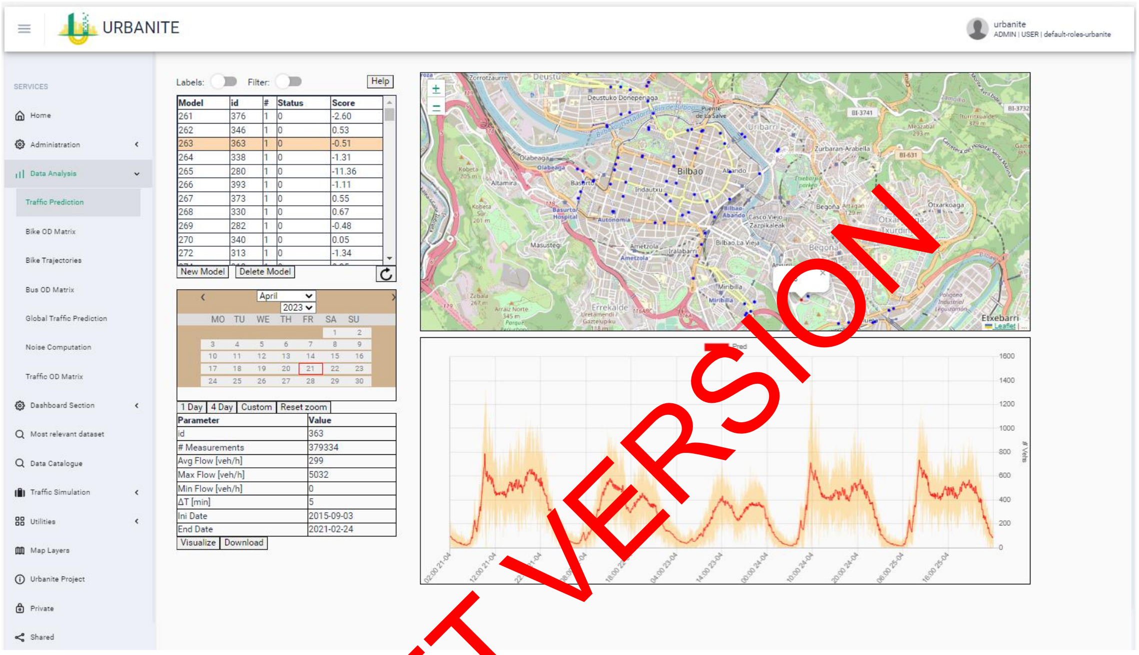
Once this information is properly set, the creation of the predicting model is started by pressing the **Submit** button. The information related to the models that have been created is shown in the **Models Table** as it can be seen in the following image: The information that is shown for each model is:

- **Model**: Unique model Id that identifies each model.
- **id:** Id for the Loop sensor associated to the model.
- **#:** Number of features.
- **Status**: Status of the training phase:
    - **0:** completed
    - **-1:** training still in process

- **Score**: Score, quality, of model, using the test data set. Only available if the Type is equal to 2 (Random Forest). This value is better the closer to 1.

Perform Prediction

Once the status of a model is equal to zero the model can be used in order to perform predictions. The prediction can be performed for a full day, for 4 days or the user can choose a custom period. The result of the prediction is shown in the graph below the map showing the actual prediction and the confidence interval associated with it. The prediction is performed for the day selected in the calendar or starting that day if the period is longer than one day.



Model	id	#	Status	Score
261	376	1	0	-2.60
262	346	1	0	0.53
263	363	1	0	-0.51
264	338	1	0	-1.31
265	280	1	0	-11.36
266	393	1	0	-1.11
267	373	1	0	0.55
268	330	1	0	0.67
269	282	1	0	-0.48
270	340	1	0	0.05
272	313	1	0	-1.34

MO	TU	WE	TH	FR	SA	SU
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Parameter	Value
id	363
# Measurements	379334
Avg Flow [veh/h]	299
Max Flow [veh/h]	5032
Min Flow [veh/h]	0
ΔT [min]	5
Ini Date	2015-09-03
End Date	2021-02-24

Figure 27.Data Analysis: Traffic Prediction.

6.9.2 Bike OD Matrix

The component is divided in two different columns: the column of the left-hand side has several controls to visualize data including a menu, two tables and a calendar, and the column of the right-hand side has a map and area to visualize charts. Now we describe these components. The controls on the left column include the following:

- **Models Table:** The models table, located on the top of the left column, show information about the characteristics and the status of the models that are or have been trained. In the downside of the table there is one button that allows to create new models. In addition, on the bottom-right side there is a button to reload (update) the data of the table. This last button is important because immediately after you submit the creation of a new model this one does NOT show up in the table, the user should press the reload button to be able to see it.
- **Calendar:** The calendar is used to choose the day to be predicted, or the starting day of the period to be predicted. This is used once the models have been trained. By default, the selected day corresponds to the current day. Notice that if a prediction that involves weather is required only a few days (4 days) in the future are susceptible to be predicted. In that particular case the not allowed days are grey out showing that it cannot be selected.

- **Matrix Output:** Shows the predicted matrix at a future instance of time, actual hour can be chosen by a slider that can be found immediately above this matrix.

On the map of the right-hand side 3 buttons can be found:

- **Load Points:** Allows to visualize the bike stations on the map.
- **Create Voronoi:** Generate Voronoi areas from the set of points previously loaded, by default these are the locations of the bike stations.
- **Load Areas:** This allows the user to specify the zoning for the OD Matrix computation by providing a geojson. Notice that the geojson Features should have an integer "zone_id" property that is used for identifying each of the zones. These "zone_id" properties should start in 1 and should be in consecutive order (i.e., no gaps are allowed). Right now, this button only allows the visualization of the areas, it does not allow to upload new areas.

Create/Train Model

The training of a model can be initiated by pressing the **New Model** button located inside the model table. This opens up a new window with two menus one for the areas available in the platform another one for specifying the number of features to be consider for the model:

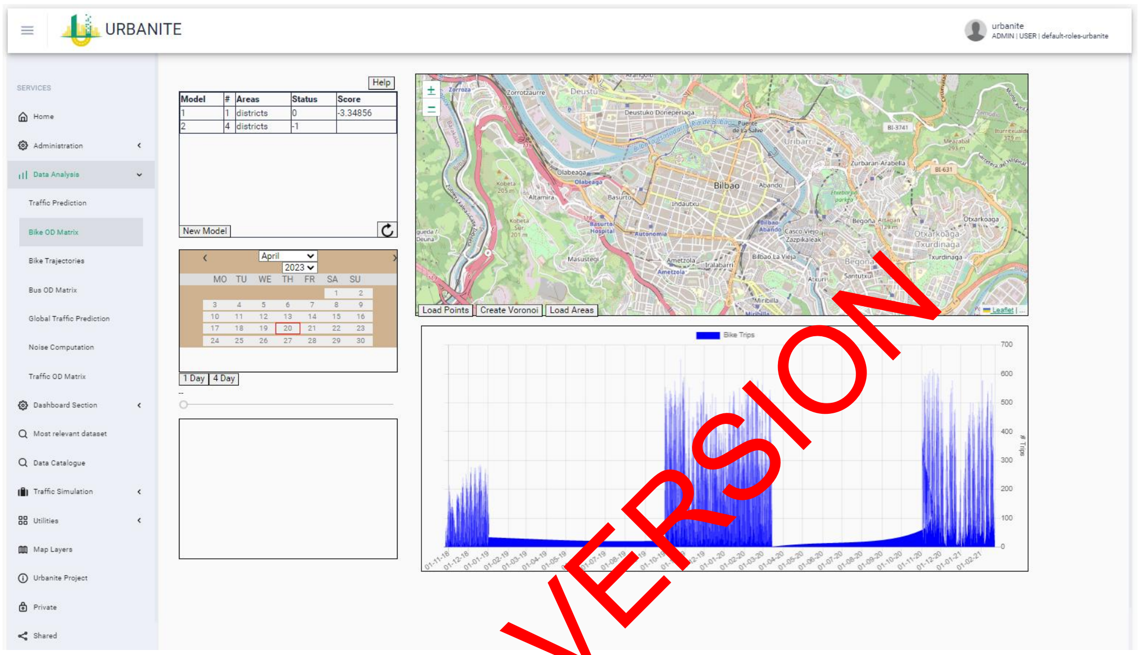
- **Areas:** The identification of the zoning to be used for the computation. Right now, the only available zoning are the "districts".
- **Num Features:** number of features considered to create the model:
  - **1**: Considers only the time of the day, Hour of the Day (HoD). We call it the *slot* of the day, this corresponds to the time interval of the day, considering the measuring period of the given sensor.
  - **2**: In addition to the previous considers the day of the week (DoW).
  - **3**: In addition to the previous considers if the given day is a school day and/or a Holiday or not (SaH).
  - **4**: In addition to the previous considers the meteorological variables (MET).
  - **5**: In addition to the previous considers events.

Once this information is properly set, the creation of the predicting model is started by pressing the **Submit** button. The information related to the models that have been created is shown in the **Models Table**. The information that is shown for each model is:

- **Model**: Unique model Id that identifies each model.
- **#:** Number of features.
- **Areas:** The name of the zoning used for the computation.
- **Status**: Status of the training phase:
  - **0:** completed
  - **-1:** training still in process
- **Score**: Score, quality, of model, using the test data set. This value is better the closer to 1.

Perform Prediction

Once the status of a model is equal to zero the model can be used in order to perform predictions. The prediction can be performed for a full day, for 4 days or the user can choose a custom period. The result of the prediction is shown both in matrix form below the calendar for an instant of time and in the graph below the map showing the actual prediction of the confidence interval associated with the prediction. The prediction is performed for the day selected in the calendar or starting that day if the period is longer. The result is shown in matrix form on the left bottom part of the module. The matrix has as many rows and columns as areas in your zoning geojson, for districts zoning it is 8. Moving the slider on top of the matrix the time of the prediction can be changed, hour by hour, and by clicking on one of the cells of the matrix the prediction for the whole time period will be shown on the chart on the bottom right-hand side.



Model	#	Areas	Status	Score
1	1	districts	0	-3.34856
2	4	districts	-1	

Figure 28. Data Analysis: Bike OD Matrix

6.9.3 Bike trajectories

The component shows the most popular locations for bikes travelling within the city. The locations are painted with different colours from white (less popular) to dark red (more popular). In the case there is no information the pointsopenUploadModalata used are the trajectories obtained from the bicycle rental service of the city. Each trajectory has been matched to points in the road network of the city. This module has two modes of operation:

- **Per Day:** The points are aggregated by days and each day can be chosen with the slider on top of the map.
- **Filter:** The day within a week (Monday, Tuesday, ...) can be chosen using the menu and the part of the day (morning or afternoon)

Amsterdam Use Case

In the case of Amsterdam use case, the data used is provided by Ring-Ring. The data is aggregated by the day of the week (morning, Tuesdays, ...) and the part of the day (morning, afternoon) This module has three modes of operation:

- **# Trajs:** The most popular locations are shown in darker red

- **Speed:** The points with larger speeds are shown in darker blue, we notice that the map matching procedure in this case breaks the finite difference approximation used to estimate the speed making this result unreliable.
- **Safety Index:** Computed by using the static characteristics of the roads and paths within the city, i.e., this index does not consider the actual number of bikers on a given road.

Other Functionality

In both use cases they are buttons that allows to visualize this help, to upload the geojson that is being shown into the URBANITE data Storage, to download the points, and to download the values shown into your computer.

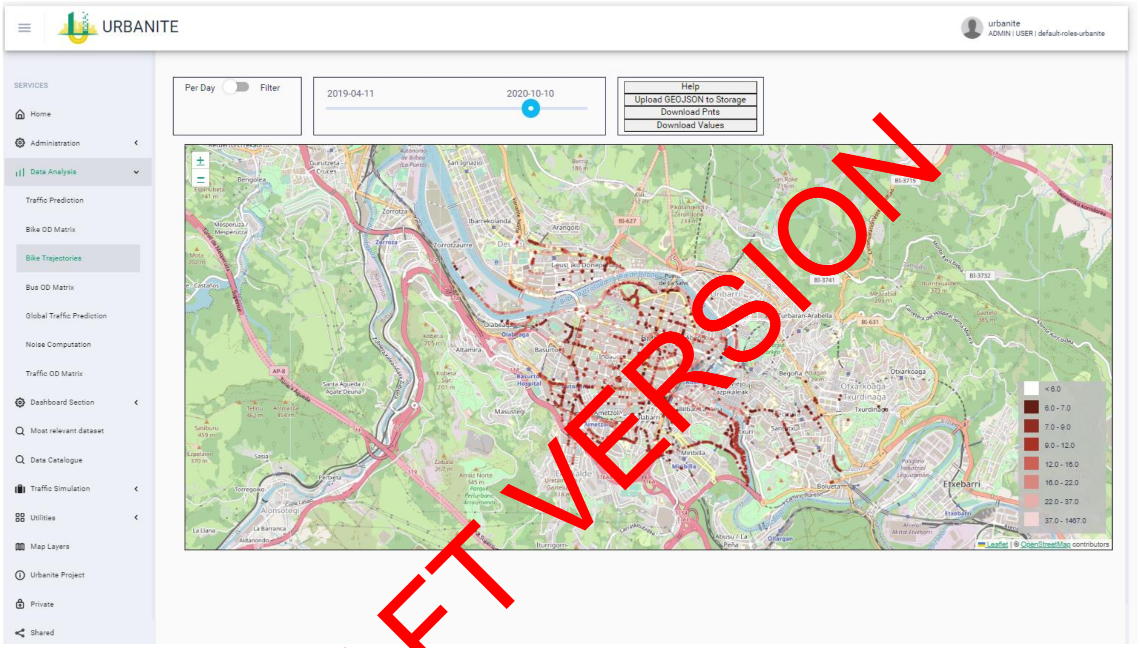


Figure 29.Data Analysis: Bike Trajectories

6.9.4 Bus OD Matrix

This component shows the results of applying the Trip-Chaining algorithm to the Smart Card data for the bus service Bilbobus of the Bilbao Use Case. The algorithm computes the OD-matrixes, specifically it produces estimates of where the passengers that get into the bus at a given stop get off from it.

The module has two different modes of operation: Per Line and Per Trip.

The **Per Line** mode shows the results aggregated per each of the lines. So, selecting one bus line from the menu on the left and pressing the button "Get Line" it shows the results. In the map on the top the different bus stops for each for the selected bus line are shown. On the bottom a chord diagrams shows with the yellow dots denoting the bus stops. By placing the mouse on top of one of the points the chord diagram shows the results of where the passengers have disembarked. The same data are also shown in a table on the left of the component.

It is worth mentioning that we have added fake bus stops with negative values denoting different reasons why the algorithm does not provide a reasonable answer. The values are aggregated over the whole set of the data which in our case is one month (Sept. 2021).

The **Per Trip** mode shows the results on a whole trip basis, allowing changes within this trip. This implies that potentially every stop of every bus line is potentially connected with every other one. In this mode, an initial bus stop needs to be selected from the map and the results are shown on the table on the left of the components. The bus stops that are more popular are colored from gray, no trip ends, to yellow, small number of ends, to darked tones of red, many number of ends.

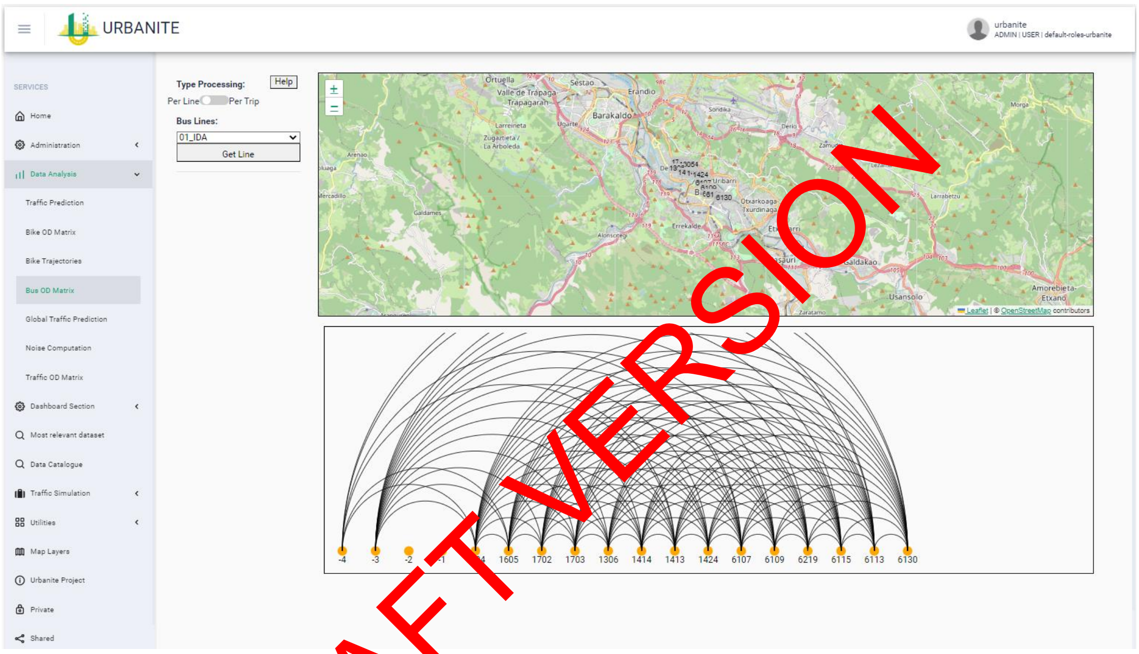


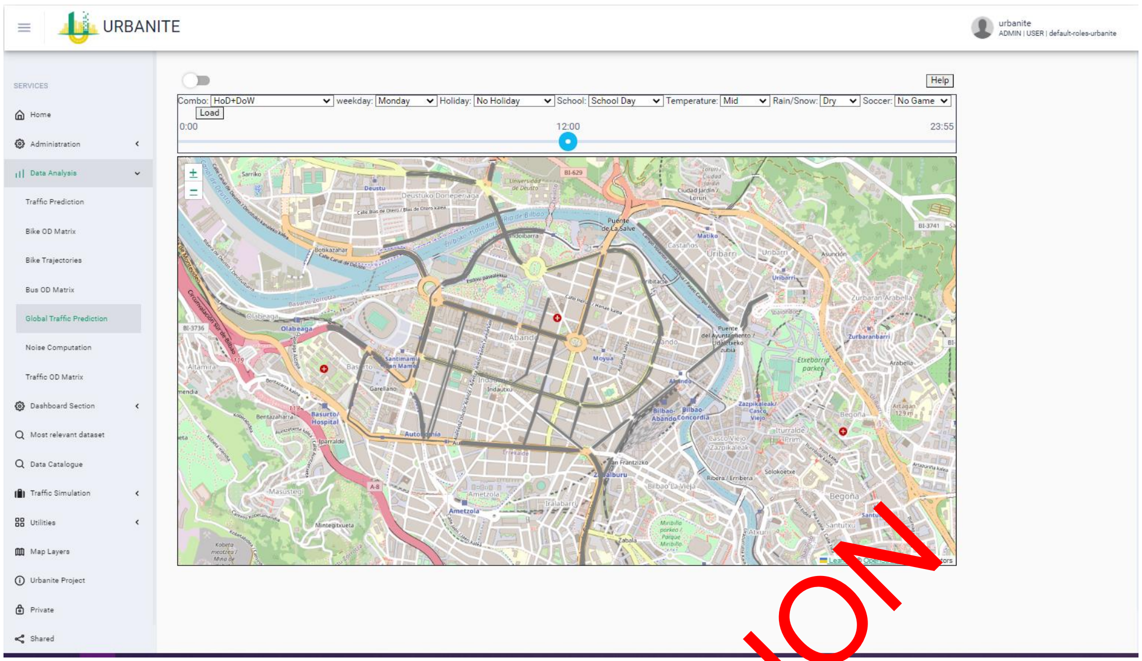
Figure 30.Data Analysis: Bus OD Matrix.

6.9.5 Global Traffic Prediction

This component allows to visualize predictions of specific conditions for the whole city. Specifically, the same combination of features that are available in the Traffic Prediction module can be chosen within this module and in order to be able to use this module, the corresponding models need to be trained in the traffic prediction module.

Once the type of prediction is chosen by means of the "Combo" menu, the specific values for the features can be chosen with the corresponding menus on top of the component, when the values of the features are chosen the predictions are obtained from the models by pressing the "Load" button.

The information can be visualized globally by changing the slider, which allows to change the hour of the day when the prediction is performed. Different predicted intensities are shown with different colors in the map. Alternatively, the prediction can be seen at a given location by clicking in the specific sensor on the map.



Combo: HoD+DoW weekday: Monday Holiday: No Holiday School: School Day Temperature: Mid Rain/Snow: Dry Soccer: No Game

Figure 31.Data Analysis: Global Traffic Prediction.

6.9.6 Noise computation

The component is divided in two different parts the controls on the top side and a map to show the results on the bottom. This module has two different modes of operation:

Map: In this case of operation the noise computation is produced using only the characteristics of the roads and paths. From these static characteristics a typical flow of vehicles is estimated. To start a computation a map needs to be selected with the menu that is place on the right of the "Map:" label. If the menu is empty a new Map can be created by pressing the "New" button besides the menu.

Once the "New" button is pressed a pop-up window is opened. A map is created choosing a rectangular area with the control that is placed on the top right-hand side of the new map within the popup. To complete the creation of the new Map a name has to be chosen by the user on the text input on the bottom of the popup window.

Traffic Sim:

This mode of operation a given traffic simulation can be chosen in order to perform the noise simulation but for now on that feature does not work and any computation is performed considering typical sensible traffic flows.

Creation of Computation

The creation of a new computation is completed by selecting either a map or a traffic simulation and providing a name for the computation and pressing the button Create. This starts the process, and the evolution can be noticed in the menu on the top right side denoted by "Computations". The computations completed are denoted by [0], computations that are still in process are labelled by [-3], [-2], [-1].

Once a computation is fully created, i.e., with [0] label in the computations menu, the result can be seen on the map by selecting it on the menu and pressing the "show" button. Once the noise is printed in the map it can be download by pressing the "Download" button or saved into the central URBANITE data storage pressing the "Upload GEOJSON to Storage". At any time while using the component the information of the menus can be updated pressing the "Update" button.

Note: There is a menu to show the results in heatmap way but at the time this Help is written is still not usable, in order to use it the obtained geojson needs to be uploaded to the URBANITE data storage and process afterwards.

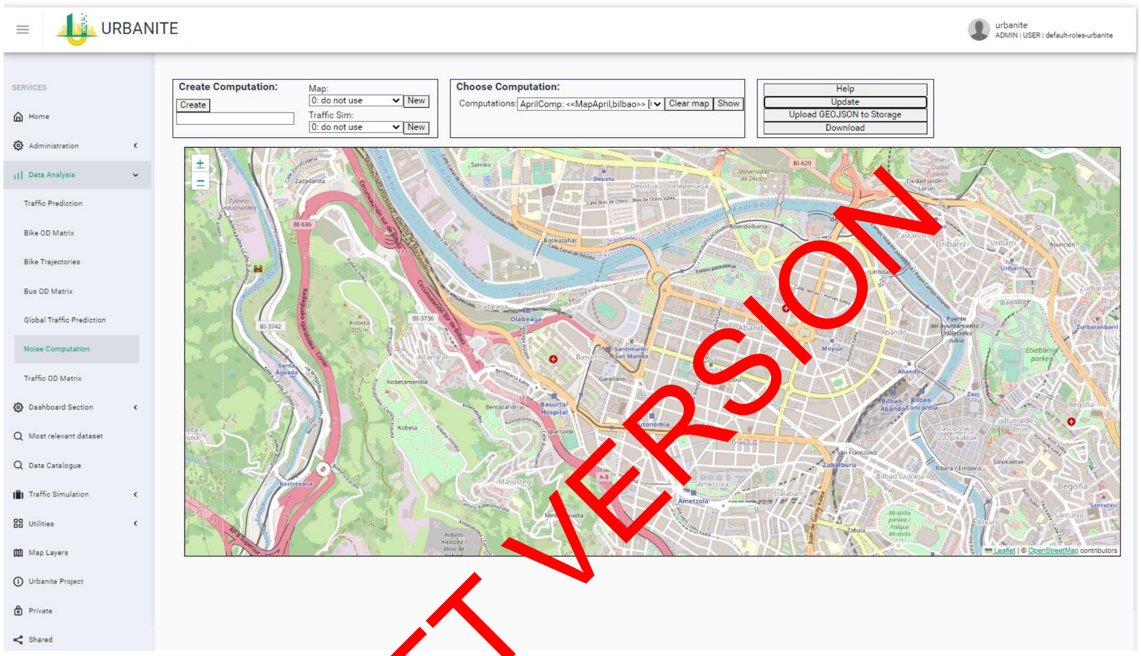


Figure [illegible]2.Data Analysis: Noise Computation

6.9.7 Traffic OD Matrix

This component shows the results obtained from an inverse problem solution to generate the OD Matrix. In this case, the module is not yet operative, i.e the sensors or areas cannot be change live, but only shows the results obtained from solving the inverse problem for a specific case.

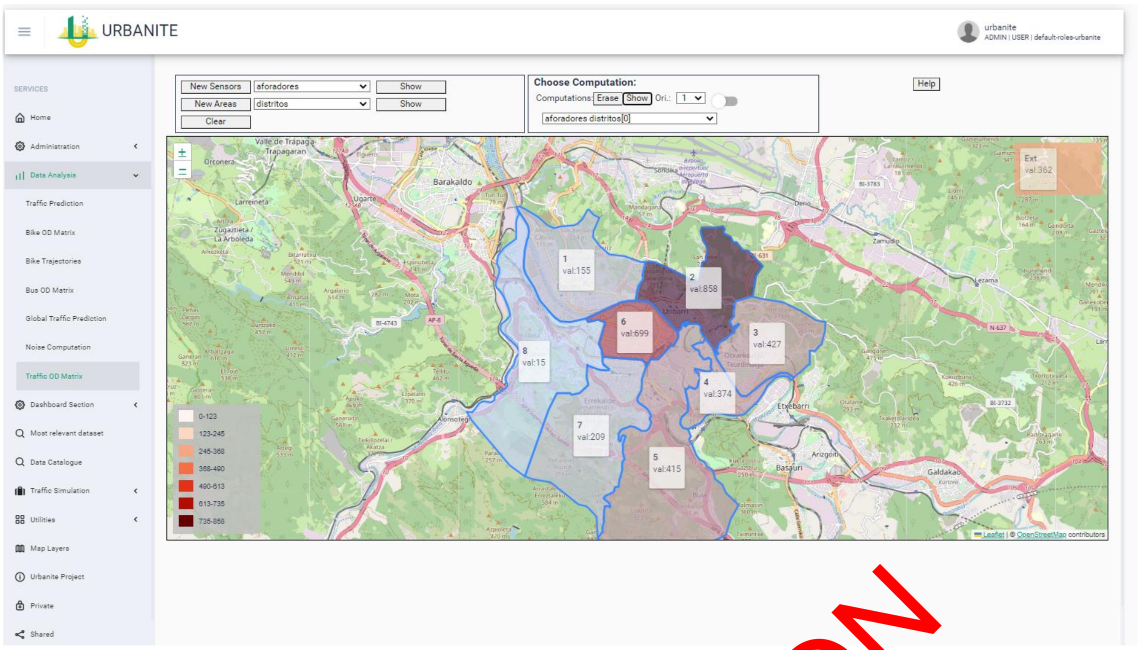


Figure 33.Data Analysis: Traffic OD Matrix

6.9.8 Bike Data

The component shows the most popular locations for bikers travelling within the city. The locations are painted with different colors from white (less popular) to dark red (more popular). In the case there is no information the pointsperUploadModalata used are the trajectories obtained from the bicycle rental service of the city. Each trajectory has been matched to points in the road network of the city. This module has two modes of operation:

- **Per Day:** The points are aggregated by days and each day can be chosen with the slider on top of the map.
- **Filter:** The day within a week (Monday, Tuesday, ...) can be chosen using the menu and the part of the day (morning or afternoon)

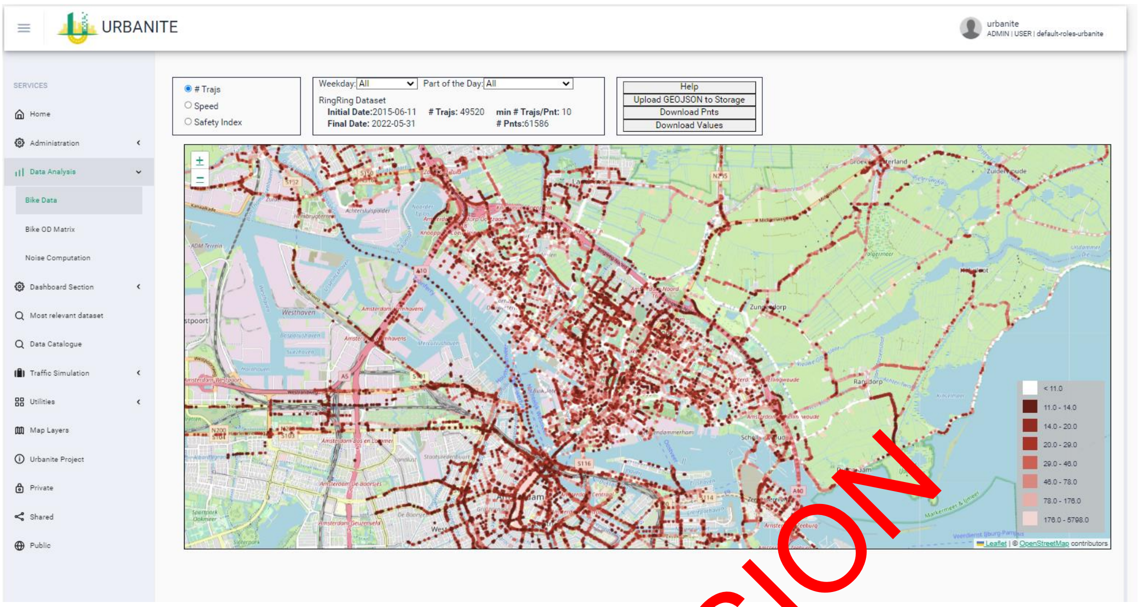
Amsterdam Use Case

In the case of Amsterdam use case, the data used is provided by Ring-Ring. The data is aggregated by the day of the week (morning, Tuesdays, ...) and the part of the day (morning, afternoon) This module has three modes of operation:

- **# Trajs:** The most popular locations are shown in darker red
- **Speed:** The points with larger speeds are shown in darker blue, we notice that the map matching procedure in this case breaks the finite difference approximation used to estimate the speed making this result unreliable.
- **Safety Index:** Computed by using the static characteristics of the roads and paths within the city, i.e. this index does not consider the actual number of bikers on a given road.

Other Functionality

In both use cases they are buttons that allows to visualize this help, to upload the geojson that is being shown into the URBANITE data Storage, to download the points, and to download the values shown into your computer.

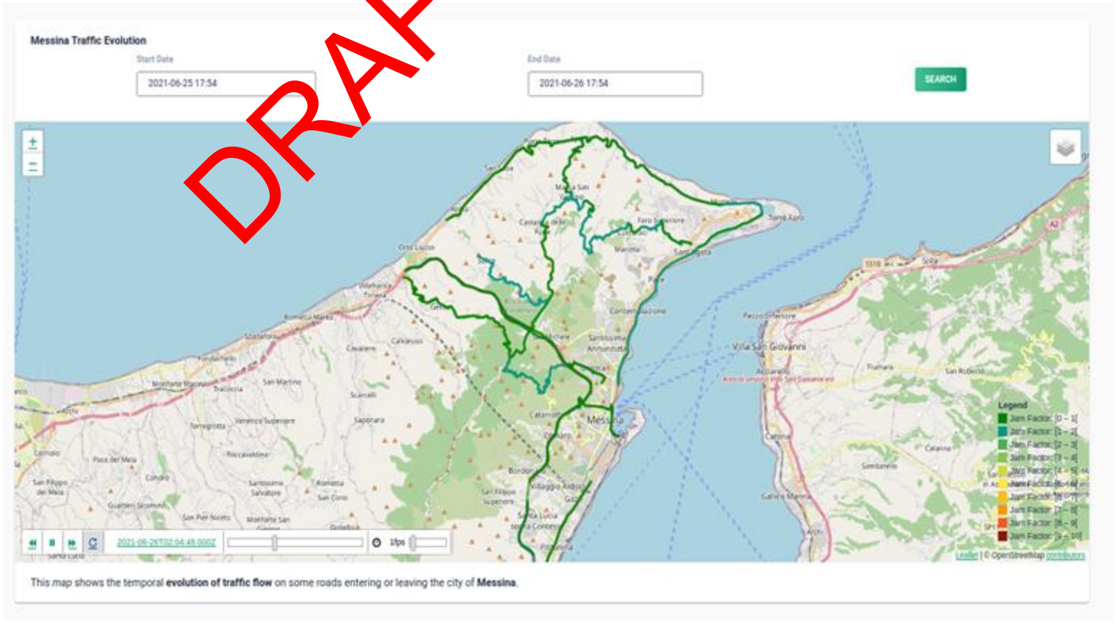


RingRing Dataset
Initial Date: 2015-06-11 # Trajs: 49520 min # Trajs/Pnt: 10
Final Date: 2022-05-31 # Pnts: 61586

Figure 34.Data Analysis: Bike Data

6.9.9 Messina Traffic Evolution

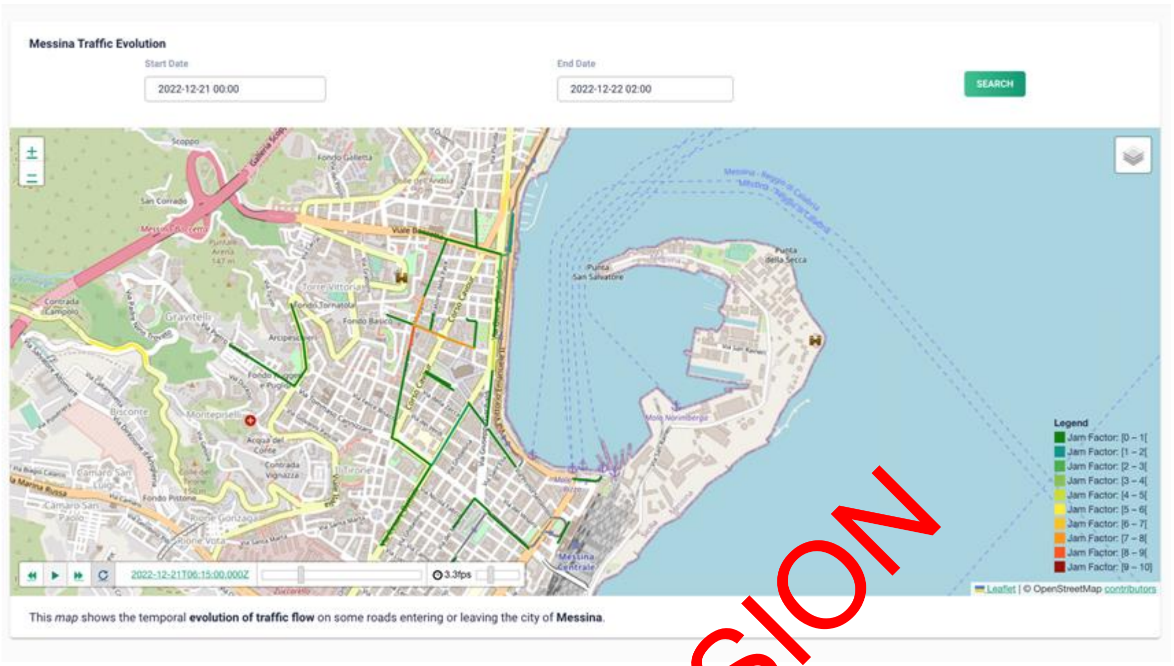
In the first graphical representation it is possible to see the temporal evolution of the traffic flow on some roads entering or leaving the city of Messina previously accomplished with the Public Service here.com with its REST-API offering.



This map shows the temporal **evolution of traffic flow** on some roads entering or leaving the city of **Messina**.

Figure 35.Data Analysis: Messina Traffic Evolution (Source: HereMap API)

In the second graphical representation (Figure 36) we can see the last version of the temporal evolution of traffic flow, built with the data collected from the Messina Municipality cameras.


Messina Traffic Evolution

Start Date: 2022-12-21 00:00

End Date: 2022-12-22 02:00

SEARCH

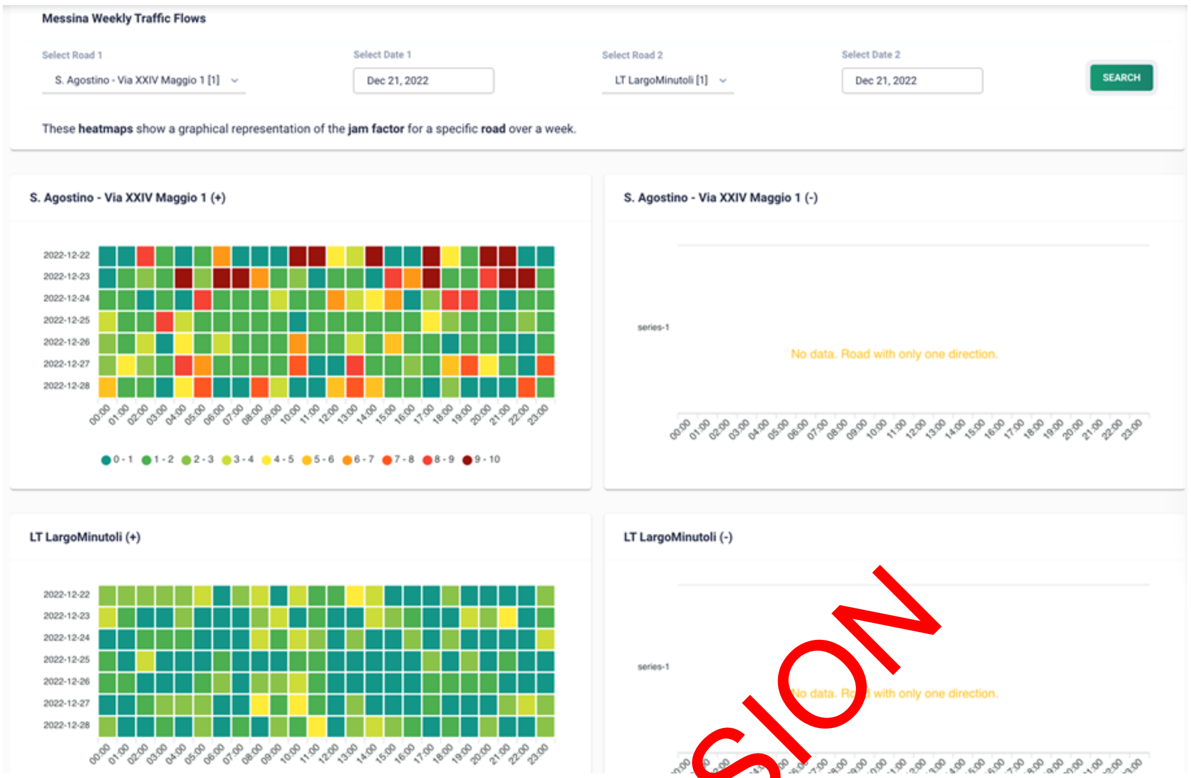
This map shows the temporal **evolution of traffic flow** on some roads entering or leaving the city of **Messina**.

Figure 36.Data Analysis: Messina Traffic Evolution (Source Source: Messina Urbanite cameras)

The new routes are smaller, but they represent a seed of beginnings for the City of Messina, which will be able to implement new ones or extend existing ones.

6.9.10 Weekly Traffic Flows

In the second type of analysis, the dashboard is depicting the Jam Factor values (that provide information about congestion level at all road segments) that are aggregated by time slots on individual days of the week, per individual road section. From this picture it is possible to quickly recognize the period of the day where the roads are particularly congested.


Messina Weekly Traffic Flows

Select Road 1: S. Agostino - Via XXIV Maggio 1 [1]
Select Date 1: Dec 21, 2022
Select Road 2: LT LargoMinutoli [1]
Select Date 2: Dec 21, 2022

These **heatmaps** show a graphical representation of the **jam factor** for a specific **road** over a week.

S. Agostino - Via XXIV Maggio 1 (+)

S. Agostino - Via XXIV Maggio 1 (-)

No data. Road with only one direction.

LT LargoMinutoli (+)

LT LargoMinutoli (-)

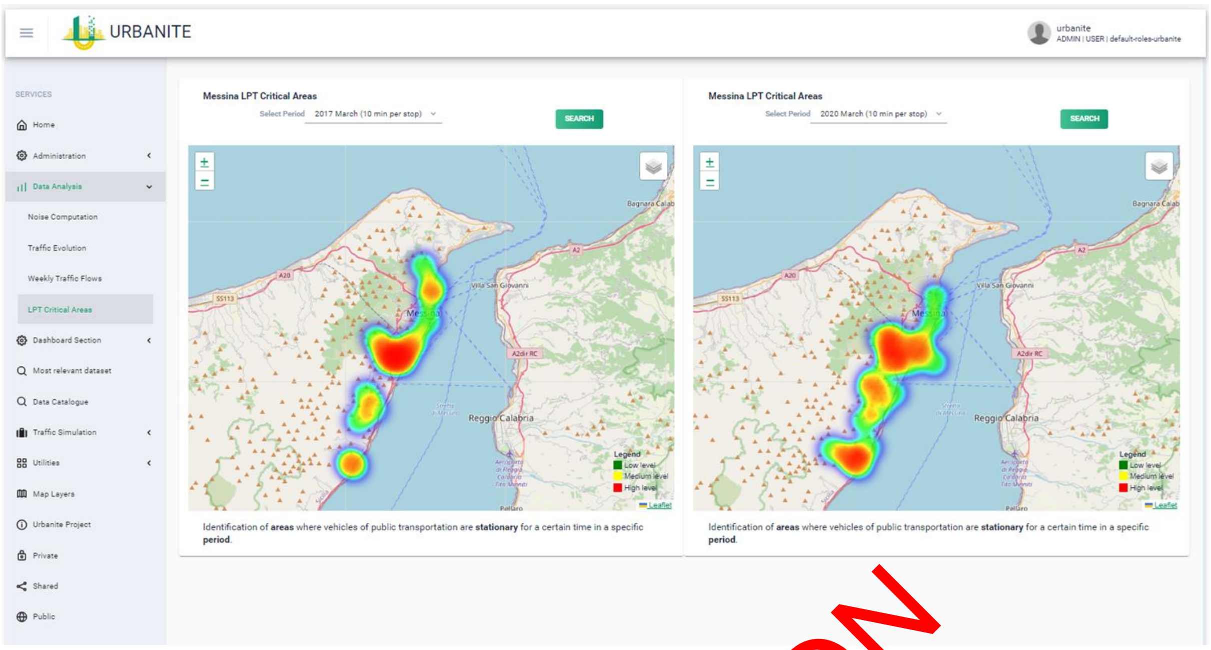
No data. Road with only one direction.

Figure 37.Data Analysis: Weekly Traffic Flows

The graph in the right part of the dashboard in Figure 37 is used in the case of bidirectional in road traffic.

6.9.11 LPT Critical Areas

This part analyses historical data for the identification of areas where vehicles of public transportation are stationary for a certain time in a specific period. It is possible to see at the same time two of these selected periods.



Identification of **areas** where vehicles of public transportation are **stationary** for a certain time in a specific **period**.

Identification of **areas** where vehicles of public transportation are **stationary** for a certain time in a specific **period**.

Figure 38.Data Analysis: LPT Critical Areas

6.10 Traffic Simulation

This option covers four functionalities:

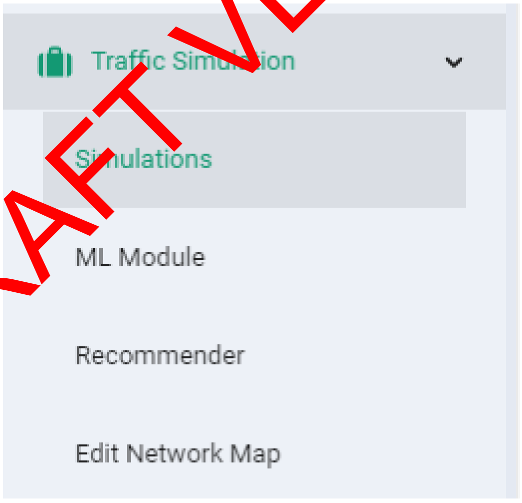
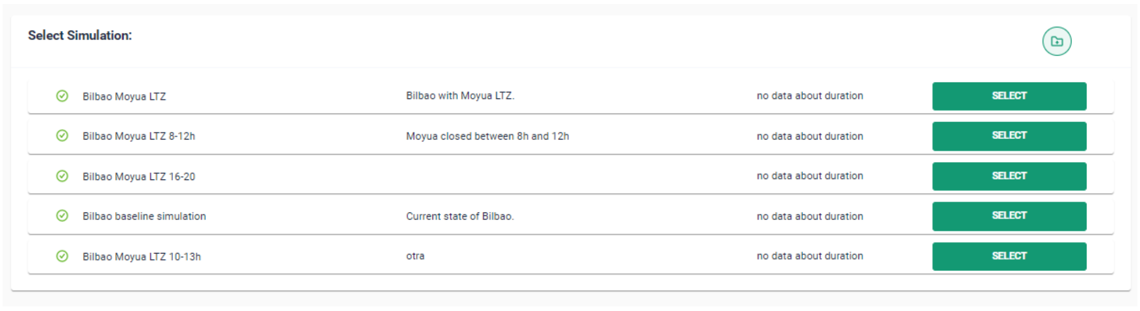


Figure 39.Traffic Simulation menu

6.10.1 Simulations

This functionality covers certain activities related to the simulation process. The component is divided into several sections, each of one allows to see the oucomes of the selected simulation at the beginning as well as analysis and comparison among them and the KPIs calculated.

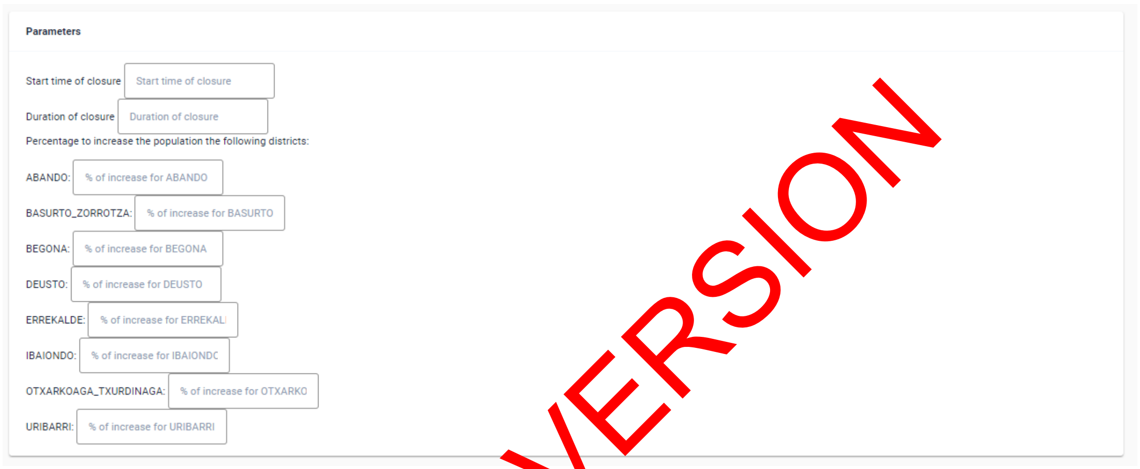


Select Simulation:

✓ Bilbao Moyua LTZ	Bilbao with Moyua LTZ.	no data about duration	SELECT
✓ Bilbao Moyua LTZ 8-12h	Moyua closed between 8h and 12h	no data about duration	SELECT
✓ Bilbao Moyua LTZ 16-20		no data about duration	SELECT
✓ Bilbao baseline simulation	Current state of Bilbao.	no data about duration	SELECT
✓ Bilbao Moyua LTZ 10-13h	otra	no data about duration	SELECT

Figure 40.Traffic Simulation: Select Simulation section

The new simulation created can be configured with the different values of the input, depending on the pilot.



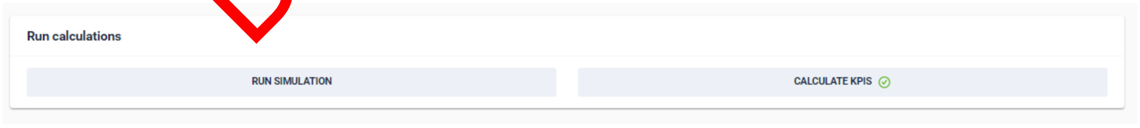
Parameters

Start time of closure: Start time of closure
Duration of closure: Duration of closure
Percentage to increase the population the following districts:
ABANDO: % of increase for ABANDO
BASURTO_ZORROTZA: % of increase for BASURTO
BEGONA: % of increase for BEGONA
DEUSTO: % of increase for DEUSTO
ERREKALDE: % of increase for ERREKALI
IBAIONDO: % of increase for IBAIONDC
OTXARKOAGA_TXURDINAGA: % of increase for OTXARKO
URIBARRI: % of increase for URIBARRI

Figure 41.Traffic Simulation: Parameters section

Once the simulation has been selected or created based on the existing ones, the run simulation button allows to obtain the outcomes needed for the following steps. The KPIs can also be calculated in this section.

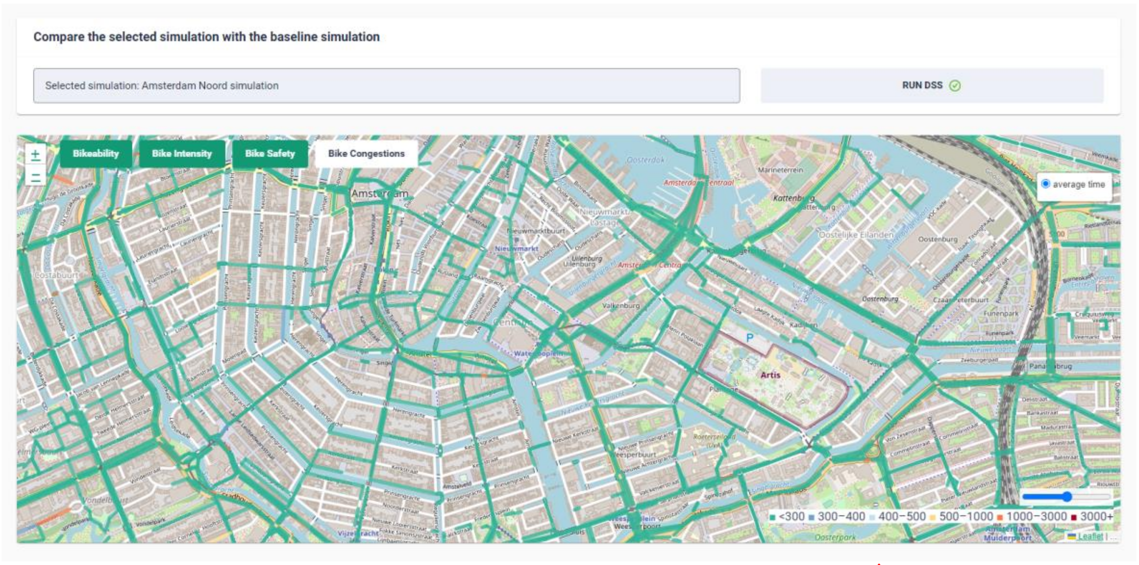
If there are some previously run simulations, the data can be used for the rest of the sections.



Run calculations

Figure 42.Traffic Simulation: Run calculations section

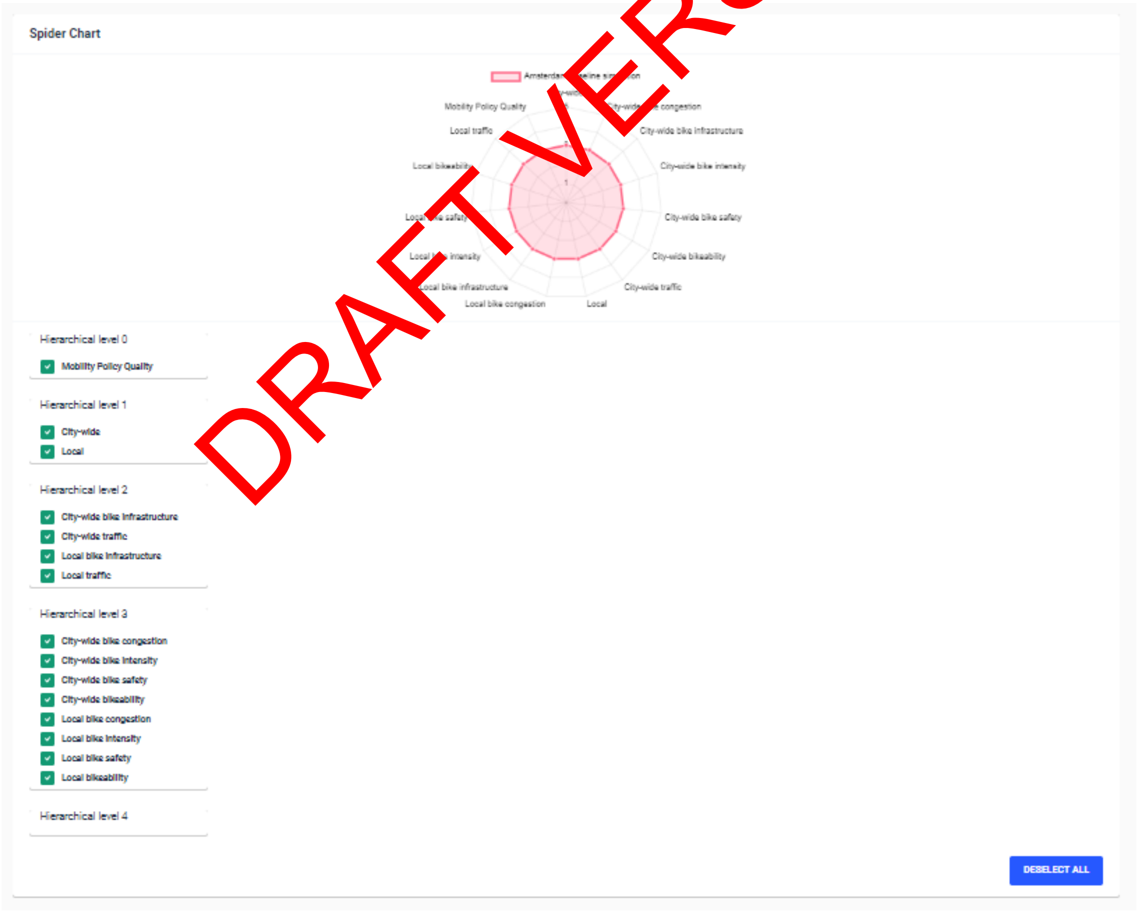
The DSS module allows to compare the selected simulation with the baseline. The results are shown on the map.


Compare the selected simulation with the baseline simulation

Selected simulation: Amsterdam Noord simulation

Figure 43.Traffic Simulation: Comparison section

The spider chart section shows the decision analysis results. The KPIs and aggregated KPIs are listed, allowing the selection of specific KPIs to compare.


Spider Chart

Hierarchical level 0

- ✓ Mobility Policy Quality

Hierarchical level 1

- ✓ City-wide
- ✓ Local

Hierarchical level 2

- ✓ City-wide bike infrastructure
- ✓ City-wide traffic
- ✓ Local bike infrastructure
- ✓ Local traffic

Hierarchical level 3

- ✓ City-wide bike congestion
- ✓ City-wide bike intensity
- ✓ City-wide bike safety
- ✓ City-wide bikeability
- ✓ Local bike congestion
- ✓ Local bike intensity
- ✓ Local bike safety
- ✓ Local bikeability

Hierarchical level 4

Figure 44.Traffic Spider chart section

The scenario comparison graph shows the result of the comparison in a different graphical presentation.

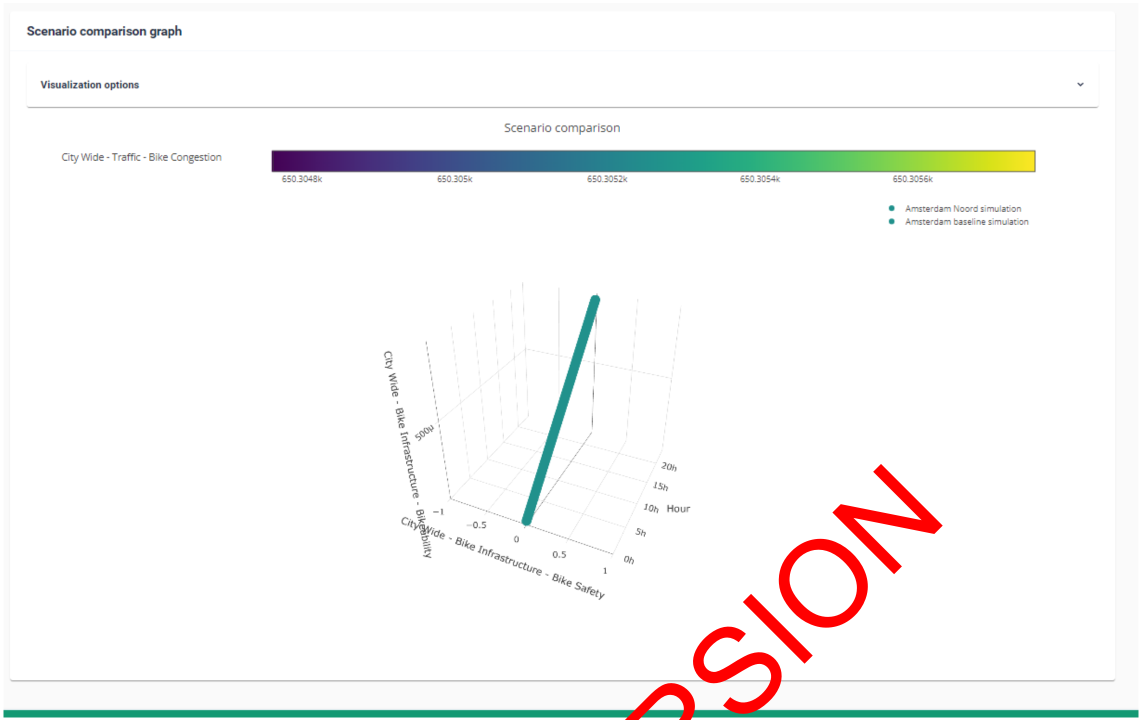


Figure 45.Scenario comparison graph section

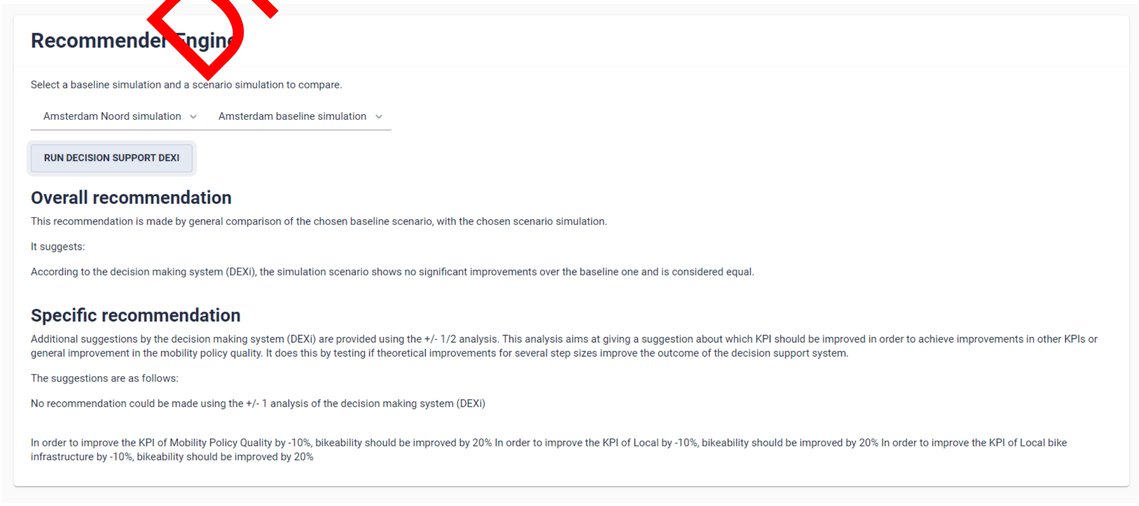
6.10.2 ML Module

Not ready

6.10.3 Recommender

This module makes some recommendations based on the outcomes of the two previously compared simulations.

There are two kinds of recommendations, overall and specific.



Recommender Engine

Select a baseline simulation and a scenario simulation to compare.

Amsterdam Noord simulation ⌄ Amsterdam baseline simulation ⌄

RUN DECISION SUPPORT DEXI

Overall recommendation

This recommendation is made by general comparison of the chosen baseline scenario, with the chosen scenario simulation.

It suggests:

According to the decision making system (DEXi), the simulation scenario shows no significant improvements over the baseline one and is considered equal.

Specific recommendation

Additional suggestions by the decision making system (DEXi) are provided using the +/- 1/2 analysis. This analysis aims at giving a suggestion about which KPI should be improved in order to achieve improvements in other KPIs or general improvement in the mobility policy quality. It does this by testing if theoretical improvements for several step sizes improve the outcome of the decision support system.

The suggestions are as follows:

No recommendation could be made using the +/- 1 analysis of the decision making system (DEXi)

In order to improve the KPI of Mobility Policy Quality by -10%, bikeability should be improved by 20% In order to improve the KPI of Local by -10%, bikeability should be improved by 20% In order to improve the KPI of Local bike infrastructure by -10%, bikeability should be improved by 20%

Figure 46.Traffic Simulations: Recommender

6.10.4 Edit Network Map

This functionality allows to add lines modifying the layout of the map.

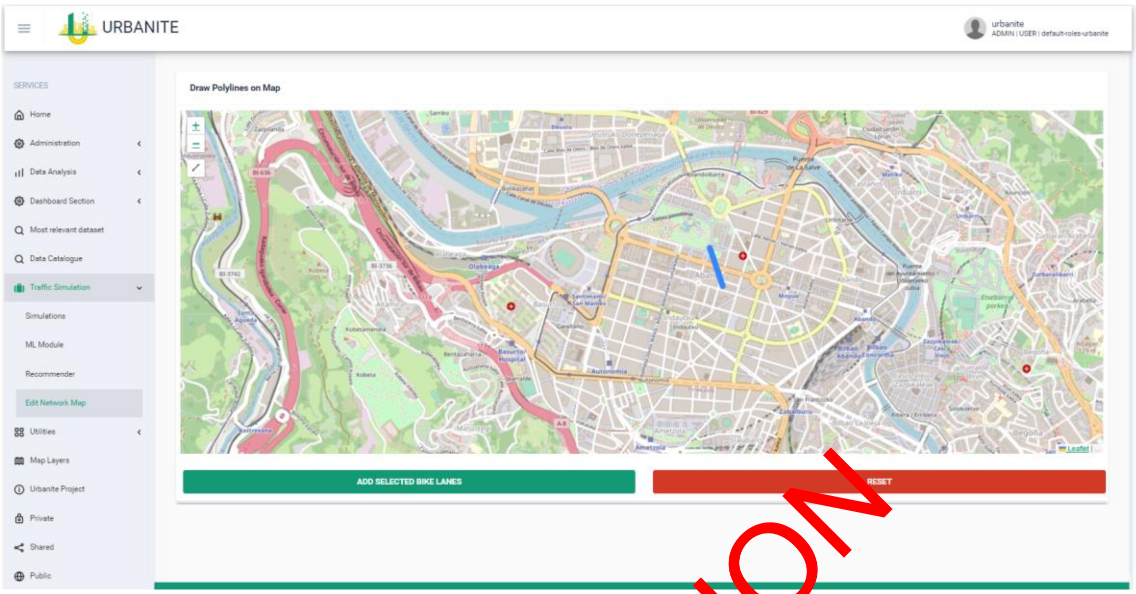


Figure 47.Traffic Simulations: Edit Network Map

7 References

[1] URBANITE Ecosystem, «D5.7 URBANITE Ecosystem-v1».

[2] URBANITE Consortium, «D5.8 URBANITE Ecosystem-v2».

[3] URBANITE Consortium, «D5.5 URBANITE Detailed architecture-v2,» 2022.

[4] URBANITE Consortium, «D5.2 Detailed requirements specification-v2,» 2021.

[5] URBANITE Consortium, «D3.3 Data Harvesting module and connectors implementation,» 2022.

[6] URBANITE Consortium, «D3.6 Data Curation module implementation,» 2022.

[7] URBANITE Consortium, «D3.8 Data Aggregation and Storage module implementation,» 2022.

[8] URBANITE Consortium, «D5.3 Integration strategy,» 2020.

[9] «Portainer,» [En línea]. Available: https://www.portainer.io/.

[10] URBANITE Consortium, «D6.2 URBANITE Use cases implementation».

[11] Akveo, «ngx-admin,» [En línea]. Available: https://akveo.github.io/ngx-admin/.

[12] Google, «Angular,» [En línea]. Available: https://angular.io/.

[13] Akveo, «Nebular,» [En línea]. Available: https://akveo.github.io/nebular/.

[14] Akveo, «Eva Design,» [En línea]. Available: https://eva.design/.

[15] URBANITE Consortium, «D4.2 Implementation of Strategies and algorithms for data modelling and visualizations,» 2022.