Deliverable
D5.3 URBANAGE Ecosystem Prototype. Initial

Project Acronym:	URBANAGE	
Project title:	Enhanced URBAN planning for AGE-friendly cities through disruptive technologies	
Grant Agreement No.	101004590	
Website:	www.urbanage.eu	
Version:	1.0	
Date:	04/05/2022	
Responsible Partner:	ATC	
Contributing Partners:	ENG	
Reviewers:	Gorka Benguria (TEC), Celia Gilsanz (SANT)	
Dissemination Level:	Public	x
	Confidential – only consortium members and European Commission	

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101004590

Revision History

Revision	Date	Author	Organization	Description
0.1	18/03/2022	Athanasios Dalianis	ATC	ToC
0.2	23/03/2022	Athanasios Dalianis, Guiseppe Ciulla	ATC, ENG	Initial content for sections 3.1 and 3.2 and initial comments
0.3	04/04/2022	Athanasios Dalianis	ATC	Initial content for section 3.3
0.4	06/04/2022	Athanasios Dalianis	ATC	Initial content for section 3.4
0.5	13/04/2022	George Giotis, Ioannis Dematis	ATC	Updates in section 3.3
0.6	18/04/2022	George Giotis, Ioannis Dematis	ATC	Updates in section 3.4
0.7	19/04/2022	Athanasios Dalianis	ATC	Updates in section 3
0.8	20/04/2022	Maritini Kalogerini	ATC	Version ready for internal review
0.9	25/04/2022 26/04/2022	Gorka Benguria Celia Gilsanz	TEC SANT	Comments from reviewers
1.0	04/05/2022	Athanasios Dalianis	ATC	Final version

Table of Contents

Table of Figures

Table of Tables

List of abbreviations

Abbreviation	Explanation
API	Application Programming Interface
CI/CD	Continuous Integration / Continuous Deployment
CIM	City Information Model
CSV	Comma Separated Values
DML	Data Management Layer
DPPA	Descriptive Predictive Prescriptive Analytics
HTTP(S)	Hypertext Transfer Protocol (Secure)
IT	Information Technology
IoT	Internet of Things
JSON	JavaScript Object Notation
ML / DL	Machine Learning / Deep Learning
MQTT	Message Queuing Telemetry Transport
REST	Representational State Transfer
SQL	Structured Query Language
WP	Work Package
UI	User Interface

1 Executive Summary

This document summarizes the actions performed under Task 5.3. "URBANAGE Ecosystem (Continuous Platform Integration and User Interface)" up to now, as well as the activities to be performed during the Task's lifetime.

The URBANAGE architecture is comprised by various components and tools that follow the microservices approach. It is based on the user requirements and specifications coming from WP2 and WP6, and the output of the modules materialised in WP3 and WP4. Once these components have been developed, the integration phase begins and allows for the completion of a working integrated system.

Deliverable 5.3 "URBANAGE Ecosystem Prototype. Initial" describes the architecture and the initial implementation plan of URBANAGE ecosystem, as well as the main platforms and tools used for the deployment of the URBANAGE components. Moreover, a brief overview of the URBANAGE components installed on the development server based on M15 integration plan, is being provided. More specifically, in this document the components that are being described are: The Big Data storage, the Digital Twin System, the Big Data Analytics System and The Data Management system. Finally, a description of the implementation and deployment of an initial version of the User and Admin UIs of the Platform is being provided.

2 Introduction

Urban planning is a complicated process that comprises utility structures, distribution chains, communication networks, infrastructure, and several other features and procedures. Urban planners are adopting urban planning software and services to handle such intricate processes and conceptualize urban designs and plan layouts. Via URBANAGE activities, the consortium implements a framework for decision making in the field of urban planning. Special attention will be paid to the building of a decision-support Ecosystem that will be the basis of the framework and that will integrate Big Data Analysis, modelling and simulation techniques with Artificial Intelligence (AI) algorithms, adapted visualization methods through Urban Digital Twins and gamification for enhanced engagement purposes. More specifically, URBANAGE will develop an Ecosystem that improves the quality of decision making on issues related to urban planning for age-friendly cities, by harnessing the collective intelligence of users. The Ecosystem will integrate a dynamic iterative approach and correlations among multiple variables from multiple and varied data sources, in order to better tackle the complexity and interrelated nature of urban systems. URBANAGE project will deal with the social and political components of urban systems, the potential benefits, risks, and impact of implementing a long-term sustainable framework for data-driven decision-making with aim to lead to more sustainable decisions and cross-sectoral strategic actions.

On the one hand, the technical components to build an intelligence data management framework that can support advanced data analysis and simulation capabilities, are being developed under WP3. These components include a) Data management components that will be in charge of collecting, aggregating and harmonising different types of data, coming from various sources in the cities, b) Artificial Intelligence algorithms and simulation engines that will be able to analyse the collected data and provide predictions support decision-making processes, and c) Big Data analytics components able to analyse a large set of data to extract knowledge providing visual dashboards for the end users.

In parallel, on the other hand, the URBANAGE Digital Twin that is being developed under WP4 is an extensible platform that can be supported through solution accelerators. This platform will integrate the solution accelerators and will allow the interaction with them through a user interface.

Following the development of these components, the integration phase will begin and allow for the accomplishment of a functional integrated system and more specifically, the building of a reliable and manageable core system that is replicable and extendable. WP5 will provide the core integration activities of the project. The user requirements and specifications have been designed and gathered under WP2 & WP6 activities, while the output of the modules is being implemented under WP3 & WP4.

3 The URBANAGE Ecosystem

In this section we provide a quick reminder of the URBANAGE architecture, as well as details about the status of the deployment and installation of the components and supporting tools that comprise the URBANAGE ecosystem.

3.1 Reference to defined architecture and implementation plan

As described in deliverable 5.1 (D5.1) [1], the URBANAGE architecture is comprised by various components and tools that follow the microservices approach and interact harmonically in order to realise the business logic of the project and tackle the pilot user needs.

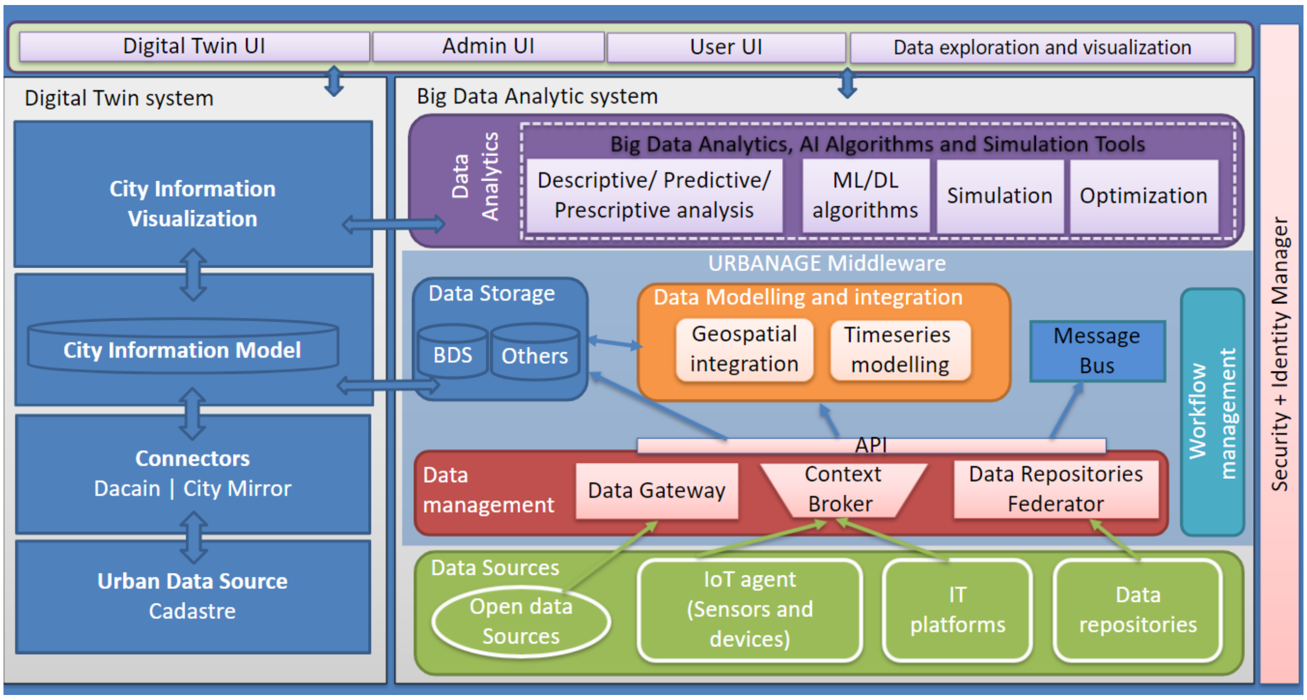


Figure 1: URBANAGE Architecture

The URBANAGE components are being implemented and integrated in releases following the agile principles. The current release of the URBANAGE ecosystem is part of the M15 milestone and according to the implementation plan described in D5.1, involves components of the Data Management layer, the Big Data Analytics tools, and the Digital Twin System. In the following sections we provide more details about the specific components and tools deployed and integrated for this release.

3.2 Deployment

The following table presents in a nutshell the main platforms and tools used for the deployment of the URBANAGE components. More specifically we provide the category of interest, the relevant tool as well as the deployed version of the tool. These technologies have been described in more detail in D5.2 [2], so in the current document we just give a reminder and update the information regarding the server specs and the tool versions.

Regarding the development server, for the purposes of the project, we have selected the dedicated server AX101 [3] provided by Hetzner [4] with 16 cores / 32 threads, 128GB RAM DDR4 and 2 x 3.84 TB NVMe SSD.

Table 1: Deployment tools

Category	Tool	Version
Code repository	Gitlab	Gitlab cloud, free plan
Component images repository	Gitlab registry	Gitlab cloud, free plan
Component deployment	Gitlab pipelines	Gitlab cloud, free plan
Cloud services provision	Hetzner dedicated server	AX101
Component packaging	Docker	20.10.14
Component orchestration	Docker compose	2.4.1

In Figure 2 we present the deployment diagram of the URBANAGE ecosystem for the current release.

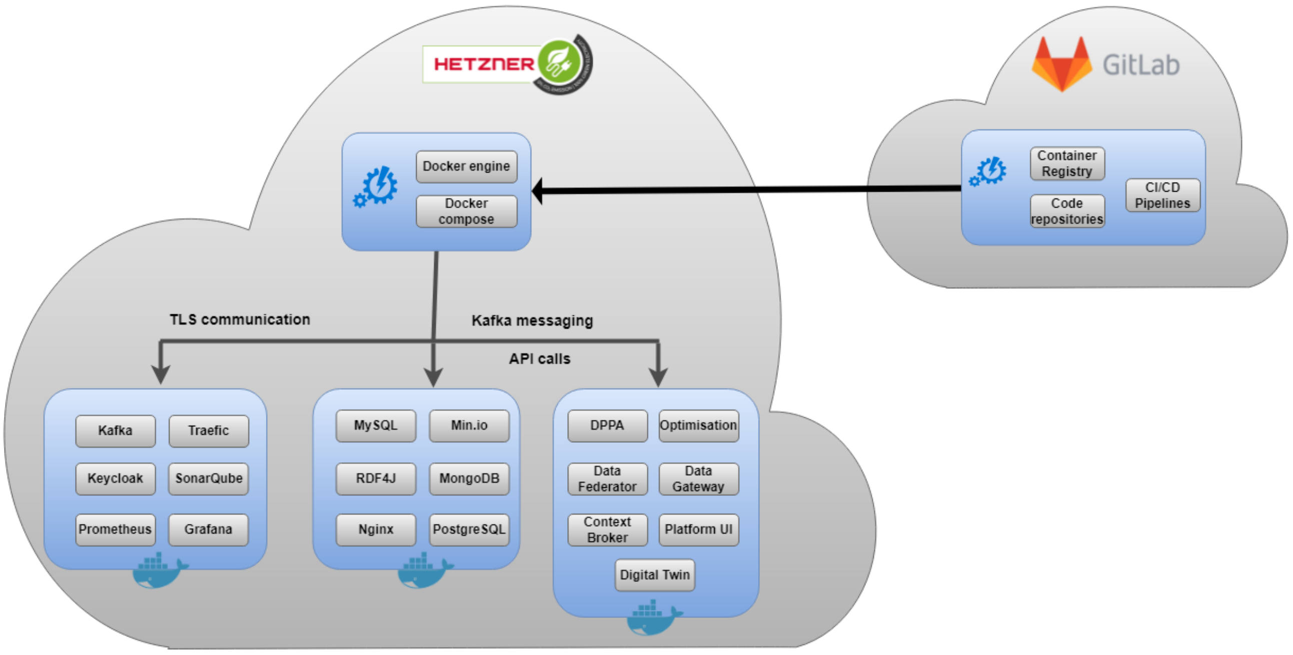


Figure 2: URBANAGE Ecosystem deployment diagram

3.3 Supporting tools deployed

In this section we provide a quick overview of the supporting tools that are deployed in the URBANAGE server and will aid the URBANAGE ecosystem in areas like communication, user authentication, monitoring etc.

The following table presents in a nutshell the category of interest, the relevant tool as well as the deployed version of the tool. For every tool we give a quick description of its purpose and some details about its deployment, like the Docker image used, its dependencies, the relevant GitLab repository, its public endpoints (if any) and screenshots for its UI (if any).

Table 2: Main URBANAGE supporting tools

Category	Tool	Version
API Gateway	Traefik	v2.6
Identity Manager	Keycloak	17.0.1-legacy
Message Bus	Kafka	2.8.1
Monitoring	Prometheus, Grafana	2.34 , 7.4.5 respectively
Code Quality	SonarQube	7.7 community

3.3.1 API Gateway

In microservices architectures, where many services are deployed in a number of virtual or physical nodes and multiple instances of the same service can exist, an API Gateway is a necessity.

An API Gateway is a component that intervenes between a web client and the backend APIs, acting as a reverse proxy that forwards the request to the appropriate microservice, usually after proper authorization.

The API Gateway thus provides a single point of access to UIs and in general external applications, decreases the complexity of implementation and allows security measures, and functionalities such as load balancing and service discovery to be applied more easily. For the purposes of the project, we are going to use Traefik [5].

Traefik is a reverse proxy and load balancer that aids in the deployment of microservices. It is capable of handling large, highly complex deployments across a wide range of protocols, and it comes with a powerful set of middleware that enhance its capabilities to include features like certificate management, https entrypoint, load balancing and API gateway.

Table 3: API Gateway

Docker image	Dependencies	GitLab repository	Endpoints
traefik:v2.6	-	https://gitlab.com/urbanageeu/api-gateway.git	https://ecosystem.urbanage.eu/dashboard/

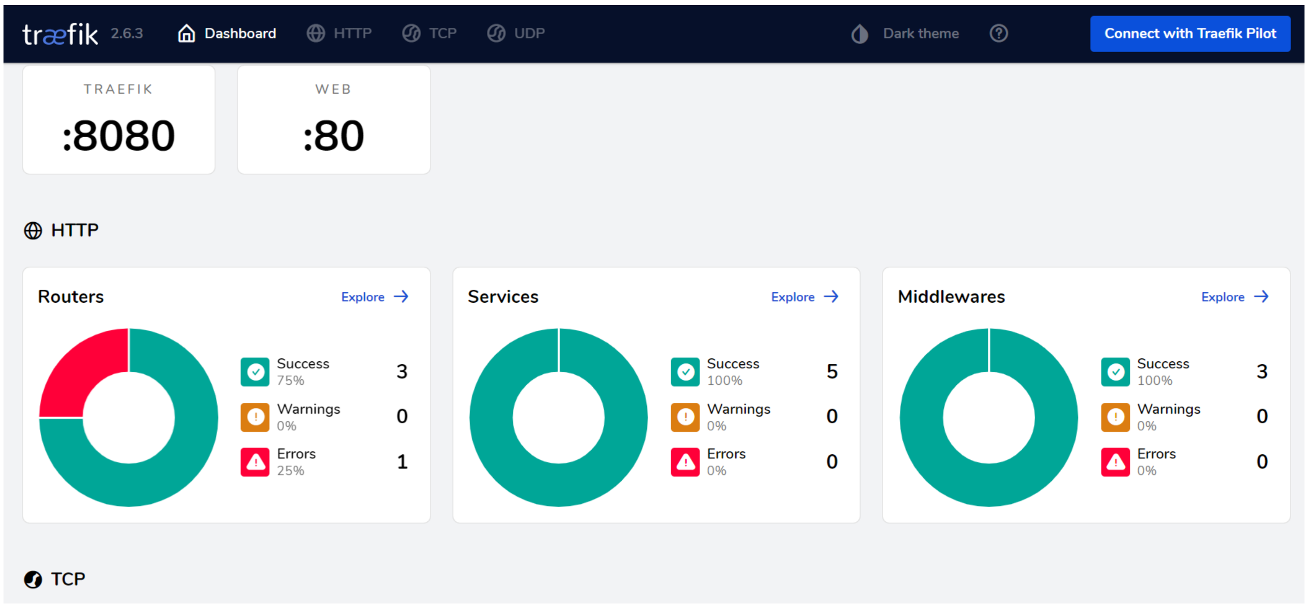


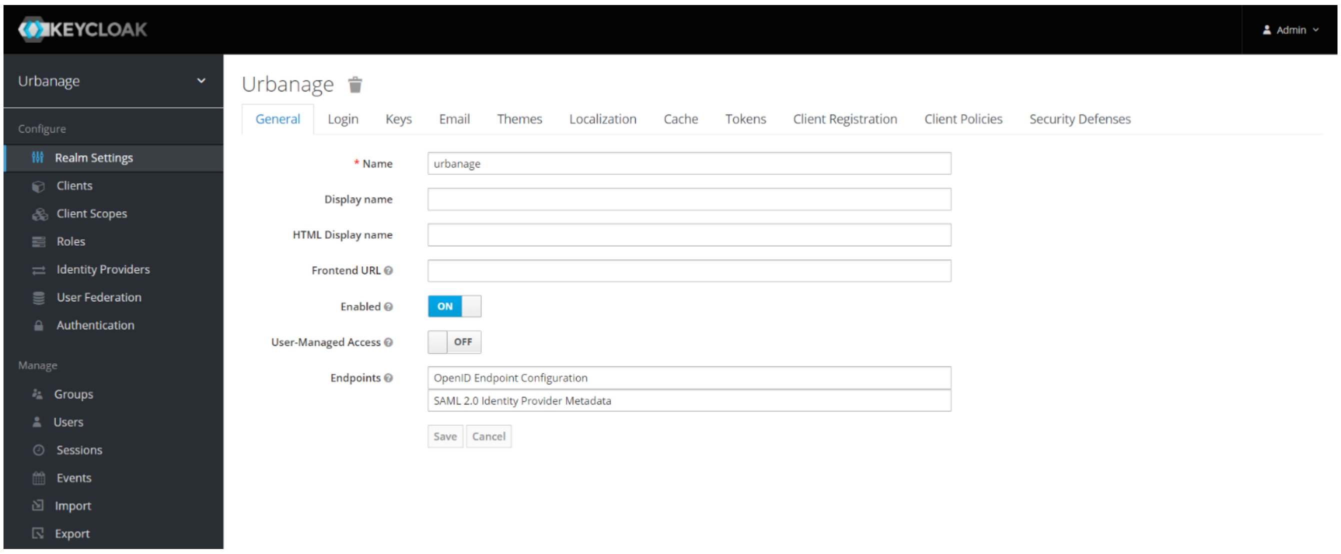
Figure 3: Traefik dashboard

3.3.2 Identity Manager

The Identity Manager is responsible to store securely the user account data and to provide authentication and authorization services to the platform. The realization of the Identity Manager will be based on Keycloak.
Keycloak [6] is an open-source identity and access manager for application and services, that provides several features for the project like centralized management, standard protocols like OAuth 2.0, SAML 2.0 etc, social login, single sign on (SSO) etc, giving us a variety of options to tackle the project needs in this area.

Table 4: Identity Manager

Docker image	Dependencies	GitLab repository	Endpoints
quay.io/keycloak/keycloak:17.0.1-legacy	PostgreSQL	https://gitlab.com/urbanageeu/urbanage-tools.git	https://ecosystem.urbanage.eu/auth/



Urbanage

* Name	urbanage
Display name	
HTML Display name	
Frontend URL	
Enabled	ON
User-Managed Access	OFF
Endpoints	OpenID Endpoint Configuration SAML 2.0 Identity Provider Metadata

Figure 4: Keycloak dashboard

3.3.3 Message Broker

The Message Broker allows the different URBANAGE components to communicate and exchange data in an asynchronous way.

Apache Kafka [7] is an open-source Massage Broker, with characteristics like stream processing, highly scalable architecture, high availability and throughput, as well as a large ecosystem of open-source tools and plugins around it e.g., MQTT support. All these characteristics make Apache Kafka a perfect option for the URBANAGE project.

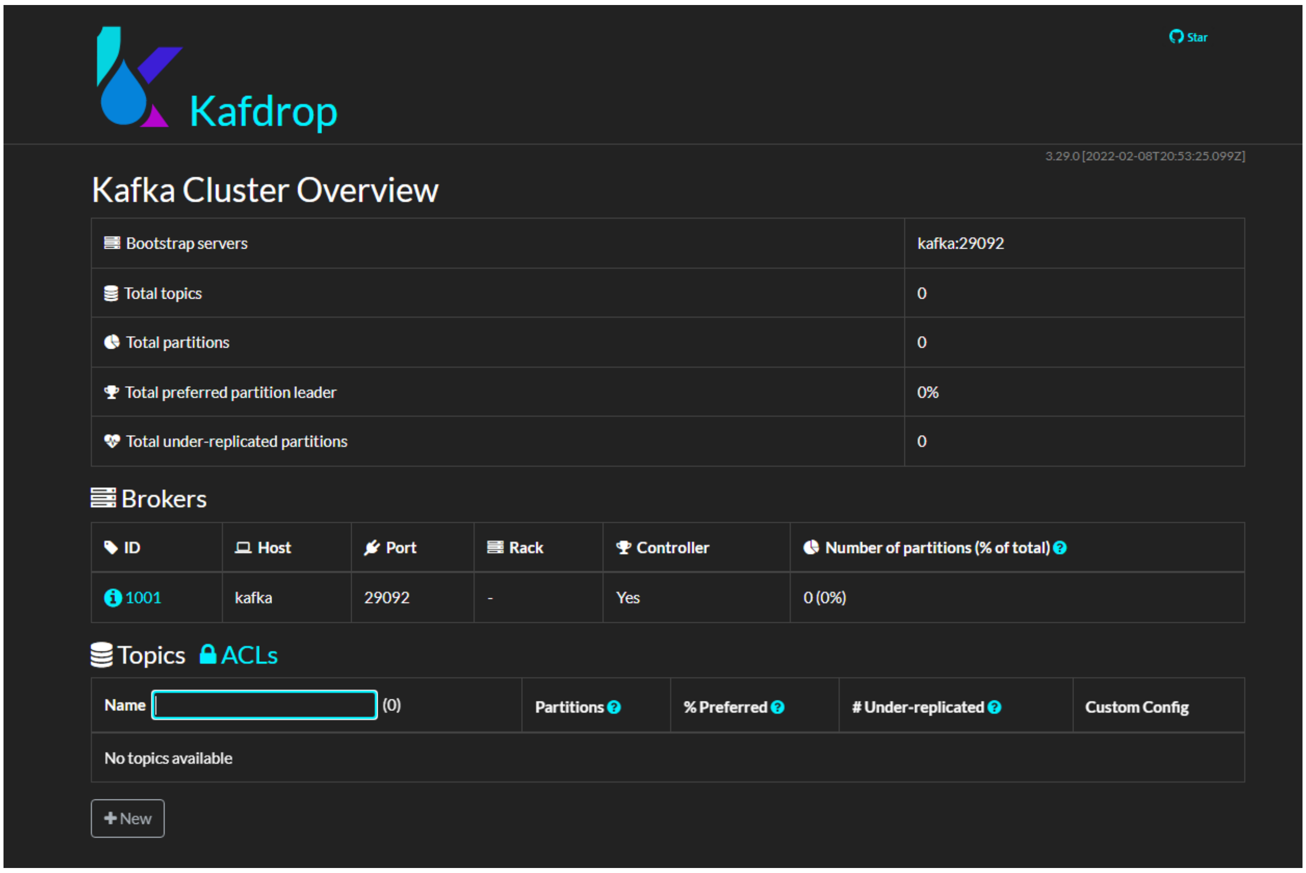
Apache Kafka is used in a master - worker mode, thus it is possible to create a cluster of Kafka Brokers, each one being able to support thousands of message queues that are called event topics. Every component that needs to exchange data through Kafka, is required to send its data to specific topics in a certain format and listen to specific topics to retrieve them.

ZooKeeper [8] is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. ZooKeeper is used in distributed systems for service synchronization and as a naming registry. When working with Apache Kafka, ZooKeeper is primarily used to track the status of nodes in the Kafka cluster and maintain a list of Kafka topics and messages.

On top of Kafka, we have deployed kafdrop [9] which is a web UI for viewing various aspects of Kafka like consumer groups, topics, consumers and messages.

Table 5: Message Broker

Docker image	Dependencies	GitLab repository	Endpoints
wurstmeister/kafka:2.13-2.8.1	Zookeeper	https://gitlab.com/urbanageeu/urbanage-tools.git	-
wurstmeister/zookeeper:latest	-	https://gitlab.com/urbanageeu/urbanage-tools.git	-
obsidiandynamics/kafdrop:3.29.0	Kafka	https://gitlab.com/urbanageeu/urbanage-tools.git	https://ecosystem.urbanage.eu/kafdrop/



Kafka Cluster Overview

Bootstrap servers	kafka:29092
Total topics	0
Total partitions	0
Total preferred partition leader	0%
Total under-replicated partitions	0

Brokers

ID	Host	Port	Rack	Controller	Number of partitions (% of total)
1001	kafka	29092	-	Yes	0 (0%)

Topics ACLs

Name (0)	Partitions	% Preferred	# Under-replicated	Custom Config
No topics available				

Figure 5: Kafdrop dashboard

3.3.4 Monitoring

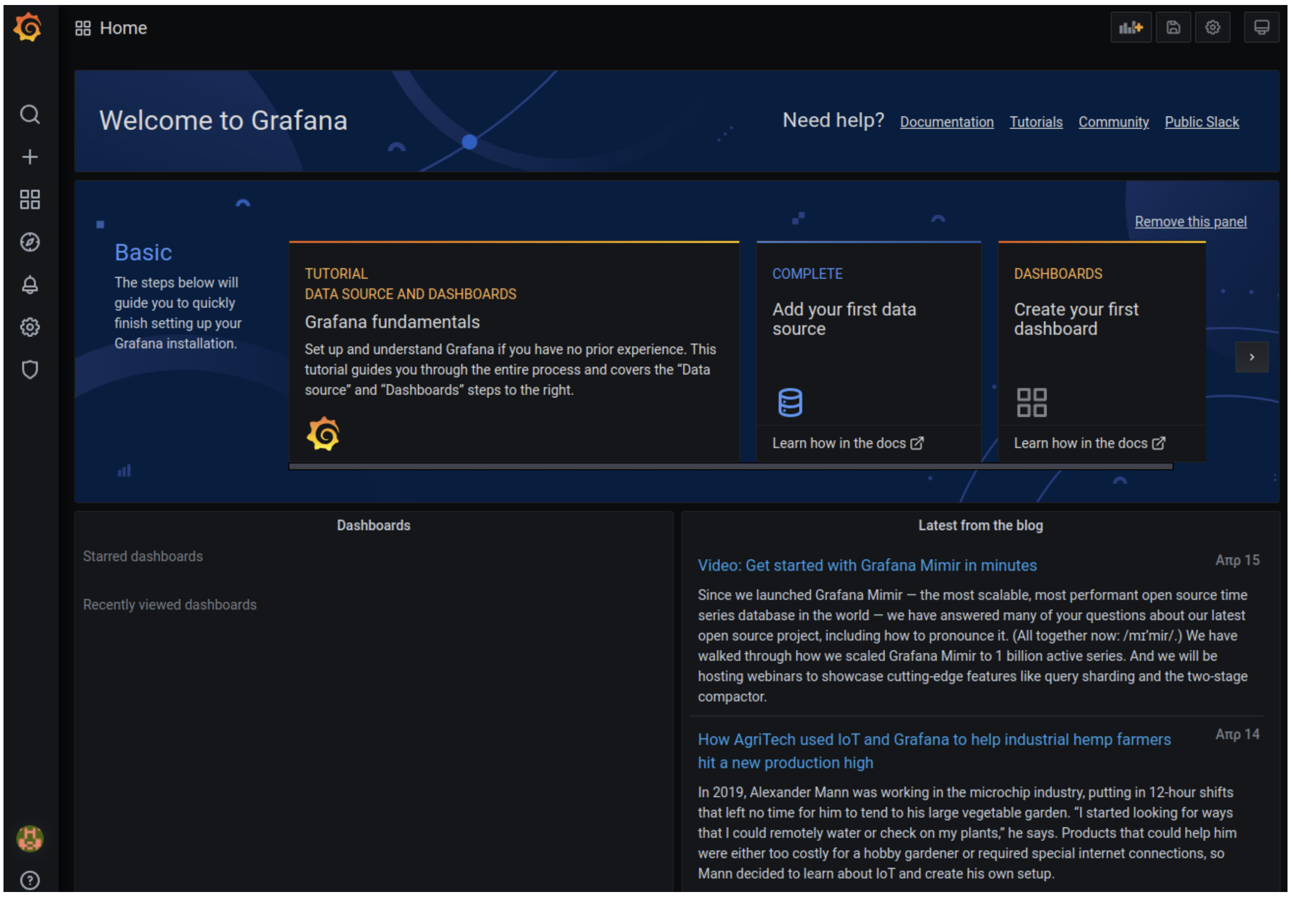
For the purposes of the project regarding software monitoring, we have deployed Prometheus [10] and Grafana [11].

Prometheus is an open-source tool under Apache License, used for event monitoring and alerting. It records real time metrics and stores them in a time series database. It features functionalities like distributed storage, multiple nodes of graphing and dashboards support and can collaborate with a wide range of tools like Docker, Kubernetes and Grafana. In order for Prometheus to retrieve more information form the host machine, it uses node exporter.

Grafana is open-source and extendable analytics and interactive visualization web application, that allows a user to query and visualize data, through a set of charts, graphs and alerts, no matter where this data is stored.

Table 6: Monitoring tools

Docker image	Dependencies	GitLab repository	Endpoints
prom/prometheus:v2.34.0	-	https://gitlab.com/urbanageeu/urbanage-tools.git	-
grafana/grafana:7.4.5-ubuntu	Prometheus	https://gitlab.com/urbanageeu/urbanage-tools.git	https://ecosystem.urbanage.eu/grafana/
prom/node-exporter:v1.3.1	Prometheus	https://gitlab.com/urbanageeu/urbanage-tools.git	-



Home

Welcome to Grafana

Need help? Documentation Tutorials Community Public Slack

Remove this panel

Basic

The steps below will guide you to quickly finish setting up your Grafana installation.

TUTORIAL
DATA SOURCE AND DASHBOARDS
Grafana fundamentals

Set up and understand Grafana if you have no prior experience. This tutorial guides you through the entire process and covers the "Data source" and "Dashboards" steps to the right.

COMPLETE
Add your first data source
Learn how in the docs

DASHBOARDS
Create your first dashboard
Learn how in the docs

Dashboards

Starred dashboards

Recently viewed dashboards

Latest from the blog

Video: Get started with Grafana Mimir in minutes — Апр 15

Since we launched Grafana Mimir — the most scalable, most performant open source time series database in the world — we have answered many of your questions about our latest open source project, including how to pronounce it. (All together now: /mɪ'mir/.) We have walked through how we scaled Grafana Mimir to 1 billion active series. And we will be hosting webinars to showcase cutting-edge features like query sharding and the two-stage compactor.

How AgriTech used IoT and Grafana to help industrial hemp farmers hit a new production high — Апр 14

In 2019, Alexander Mann was working in the microchip industry, putting in 12-hour shifts that left no time for him to tend to his large vegetable garden. "I started looking for ways that I could remotely water or check on my plants," he says. Products that could help him were either too costly for a hobby gardener or required special internet connections, so Mann decided to learn about IoT and create his own setup.

Figure 6: Grafana dashboard

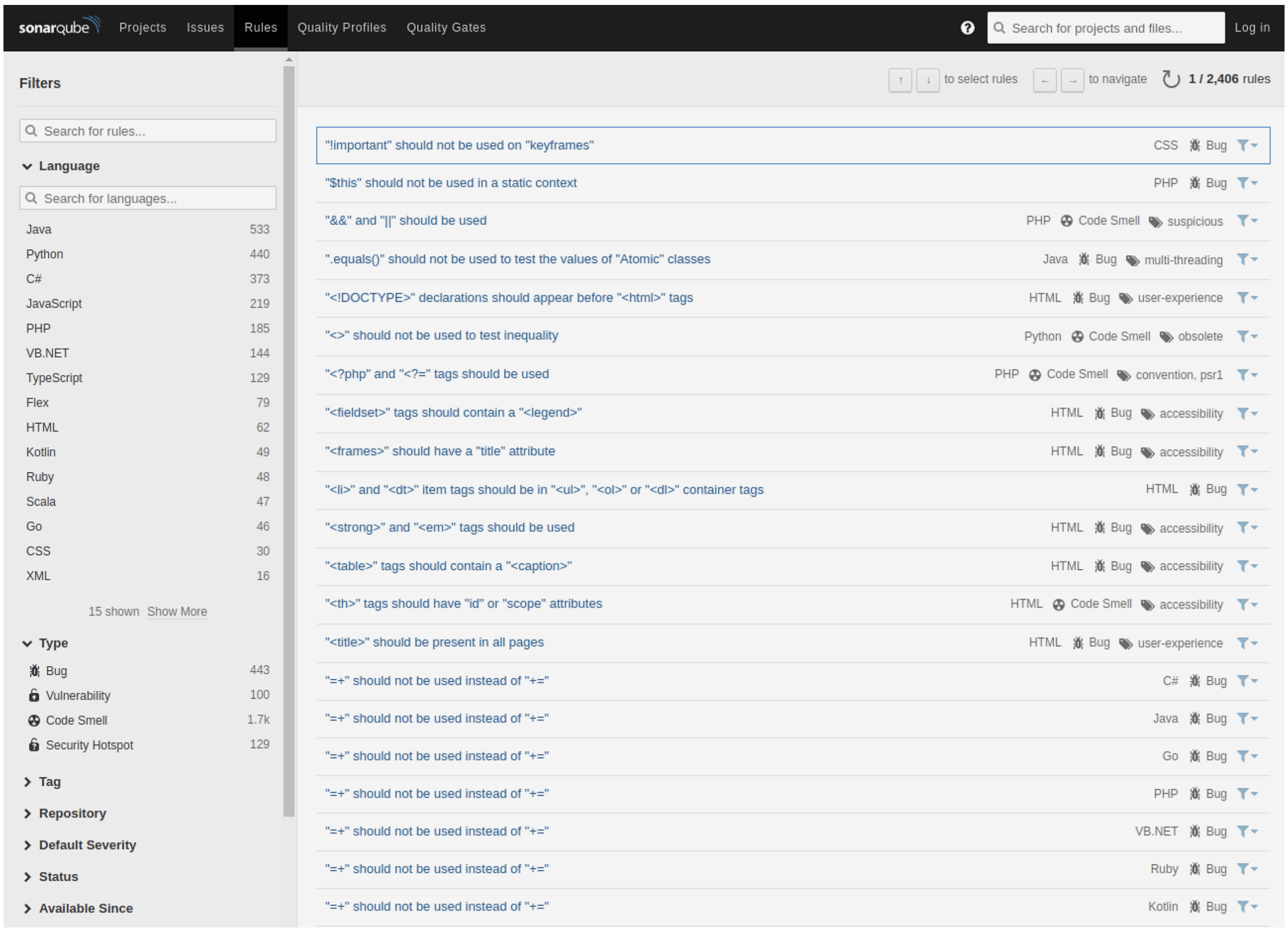
3.3.5 Code Quality

In order to have a more reliable and globally accepted measure of code quality, for the various quality metrics defined in the validation methodology, the popular SonarQube, a quality gateway, has been installed.

SonarQube [12] is an open-source platform developed for continuous inspection of code quality to perform automatic reviews with static analysis of code to detect bugs, code smells, and security vulnerabilities on more than 20 programming languages. SonarQube offers reports on duplicated code, coding standards, unit tests, code coverage, code complexity, comments, bugs, and security vulnerabilities.

Table 7: SonarQube

Docker image	Dependencies	GitLab repository	Endpoints
sonarqube:7.7-community	PostreSQL	https://gitlab.com/urbanageeu/urbanage-tools.git	https://ecosystem.urbanage.eu/sonarqube/



Filters

Search for rules...

Language

Search for languages...

Java	533
Python	440
C#	373
JavaScript	219
PHP	185
VB.NET	144
TypeScript	129
Flex	79
HTML	62
Kotlin	49
Ruby	48
Scala	47
Go	46
CSS	30
XML	16

15 shown Show More

Type

Bug	443
Vulnerability	100
Code Smell	1.7k
Security Hotspot	129

Tag

Repository

Default Severity

Status

Available Since

"!important" should not be used on "keyframes"	CSS	Bug	
"$this" should not be used in a static context	PHP	Bug	
"&&" and "||" should be used	PHP	Code Smell	suspicious
".equals()" should not be used to test the values of "Atomic" classes	Java	Bug	multi-threading
"<!DOCTYPE>" declarations should appear before "<html>" tags	HTML	Bug	user-experience
"<>" should not be used to test inequality	Python	Code Smell	obsolete
"<?php" and "<?=" tags should be used	PHP	Code Smell	convention, psr1
"<fieldset>" tags should contain a "<legend>"	HTML	Bug	accessibility
"<frames>" should have a "title" attribute	HTML	Bug	accessibility
"<li>" and "<dt>" item tags should be in "<ul>", "<ol>" or "<dl>" container tags	HTML	Bug	
"<strong>" and "<em>" tags should be used	HTML	Bug	accessibility
"<table>" tags should contain a "<caption>"	HTML	Bug	accessibility
"<th>" tags should have "id" or "scope" attributes	HTML	Code Smell	accessibility
"<title>" should be present in all pages	HTML	Bug	user-experience
"=+" should not be used instead of "+="	C#	Bug	
"=+" should not be used instead of "+="	Java	Bug	
"=+" should not be used instead of "+="	Go	Bug	
"=+" should not be used instead of "+="	PHP	Bug	
"=+" should not be used instead of "+="	VB.NET	Bug	
"=+" should not be used instead of "+="	Ruby	Bug	
"=+" should not be used instead of "+="	Kotlin	Bug	

Figure 7: SonarQube dashboard

3.4 Overview of the components installed

In this section, we provide a brief overview of the first versions of the URBANAGE components installed on our development server based on M15 integration plan. For every component / system we provide a quick overview of its role in the URBANAGE ecosystem, as well as deployment information, like the docker image used, the component’s dependencies, the related GitLab repository, its public endpoints (if any) and screenshots from its UI (if any). In summary these components are part of:

- The Big Data Storage
- The Digital Twin System
- The Big Data Analytics System
- The Data Management System
- The Platform UIs

3.4.1 Big Data storage

It provides a storage facility for the data collected and enables further analysis by the Big Data Analytics and AI components. It retains data related to both real time and offline processes. The Big Data storage facility is comprised mainly of the Min.io tool [13], and also the databases that URBANAGE components depend on, more specifically MySQL [14], PostgreSQL [15], MongoDB [16] and RDF4J [17]. More details can be found at D5.1.

To support the need for high available Big Data storage the Min.io service is set up in distributed mode. Specifically, it comprises of 4 server instances that are reverse proxied through Nginx load balancing.

Table 8: Big Data Storage

Docker image	Dependencies	GitLab repository	Endpoints
quay.io/minio/minio:RELEASE.2022-04-01T03-41-39Z	Nginx	https://gitlab.com/urbanageeu/data-storage	http://ecosystem.urbanage.eu:9001/login
mysql:5.7	-	https://gitlab.com/urbanageeu/urbanage-tools.git	-
postgres:13.0-alpine	-	https://gitlab.com/urbanageeu/urbanage-tools.git	-
mongo:latest	-	https://gitlab.com/urbanageeu/urbanage-tools.git	-
yyz1989/rdf4j		https://gitlab.com/urbanageeu/urbanage-tools.git	-
nginx:1.19.2-alpine	-	https://gitlab.com/urbanageeu/urbanage-tools.git	-

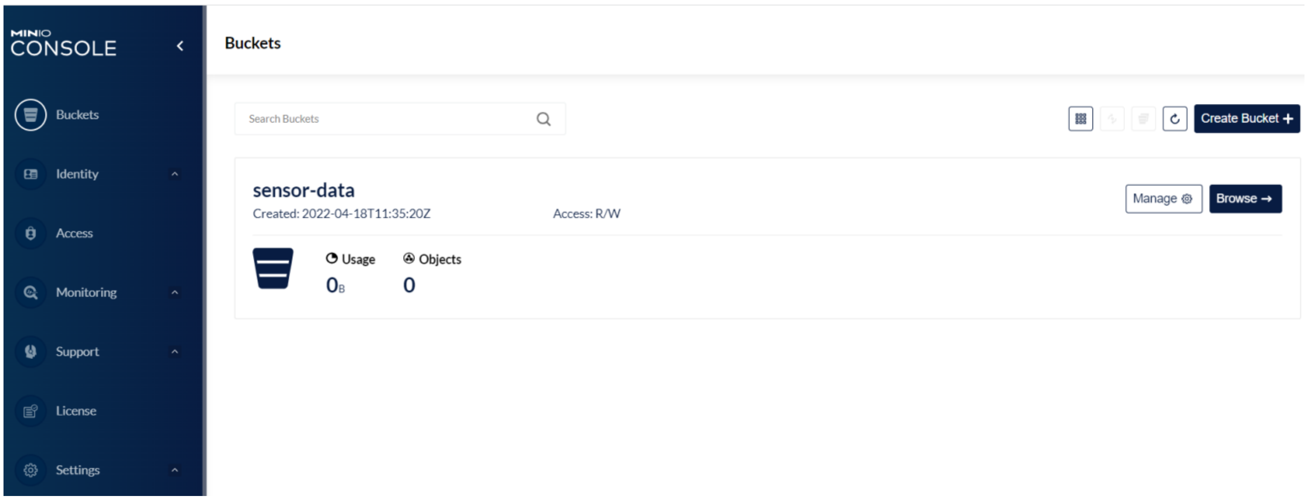


Figure 8: Min.io console

3.4.2 Digital Twin System

The Digital Twin System is a web application that gathers and processes city geographical data which presents to the user in an intuitive manner. The application comprises of various components, like a CIM database that holds the processed data, connectors to perform the necessary processing etc. More details about the Digital Twin system can be found at D5.1.

Table 9: Digital Twin System

Docker image	Dependencies	GitLab repository	Endpoints
tumgis/3dcitydb-web-map:alpine-v1.9.0	cim-db, cim-wfsm, cim-pgadmin	https://gitlab.com/urbanageeu/city-information-model.git	http://ecosystem.urbanage.eu:18086/
3dcitydb/3dcitydb-pg:9.6-3.1-4.2.0-alpine	-	https://gitlab.com/urbanageeu/city-information-model.git	-
3dcitydb/wfs:5.0.0-alpine	cim-db	https://gitlab.com/urbanageeu/city-information-model.git	https://ecosystem.urbanage.eu/wfsclient/
dpage/pgadmin4:6.8	cim-db	https://gitlab.com/urbanageeu/city-information-model.git	-

3.4.3 Big Data Analytics System

The Big Data Analytics System is responsible for analysing the data gathered by the Data Management System and provide useful information and functionalities to the user. Currently, initial versions of the Optimisation and DPPA components have been integrated on the project's development server. More details about the Big Data Analytics system and its components can be found at D5.1

3.4.3.1 Descriptive Predictive and Prescriptive Analysis

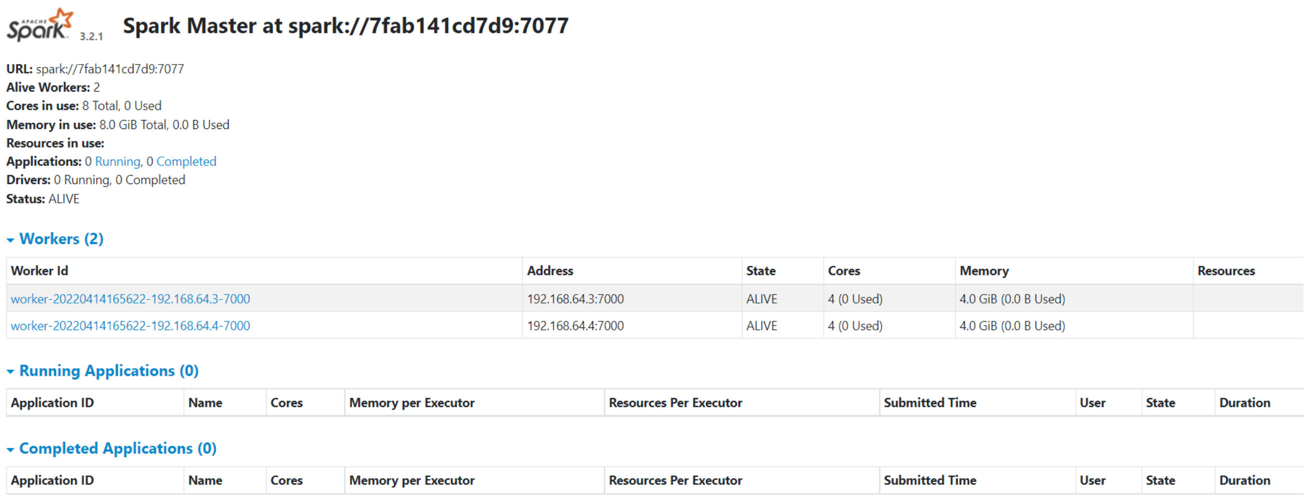
The main objective of this module is to draw valuable knowledge from all the data gathered in URBANAGE. This valuable knowledge can be divided into three different categories:

- *descriptive*, which can take raw data and obtain significant conclusions to be analysed.
- *prescriptive*, which can also be considered as optimization, which obtains the needed information and offers to the user different solutions for helping in the decision-making process.
- and *predictive*, which uses algorithms that fall within the ML and DL category, and which obtain data as input for predicting future stages of certain domains.

The DPPA component relies on the Apache Spark Big Data framework [18] as the processing backbone. A Spark cluster consisting of 2 worker nodes is set up to address the need for a reliable and highly available processing pipeline.

Table 10: DPPA

Docker image	Dependencies	GitLab repository	Endpoints
dpp-analysis:latest	Apache Spark cluster	https://gitlab.com/urbanageeu/dpp-analysis.git	-
cluster-apache-spark:3.2.1	Spark master	https://gitlab.com/urbanageeu/dpp-analysis.git	http://ecosystem.urbanage.eu:9070/
cluster-apache-spark:3.2.1	Spark worker 1	https://gitlab.com/urbanageeu/dpp-analysis.git	http://ecosystem.urbanage.eu:9071/
cluster-apache-spark:3.2.1	Spark worker 2	https://gitlab.com/urbanageeu/dpp-analysis.git	http://ecosystem.urbanage.eu:9072/


Spark Master at spark://7fab141cd7d9:7077

URL: spark://7fab141cd7d9:7077
Alive Workers: 2
Cores in use: 8 Total, 0 Used
Memory in use: 8.0 GiB Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (2)

Worker Id	Address	State	Cores	Memory	Resources
worker-20220414165622-192.168.64.3-7000	192.168.64.3:7000	ALIVE	4 (0 Used)	4.0 GiB (0.0 B Used)	
worker-20220414165622-192.168.64.4-7000	192.168.64.4:7000	ALIVE	4 (0 Used)	4.0 GiB (0.0 B Used)	

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

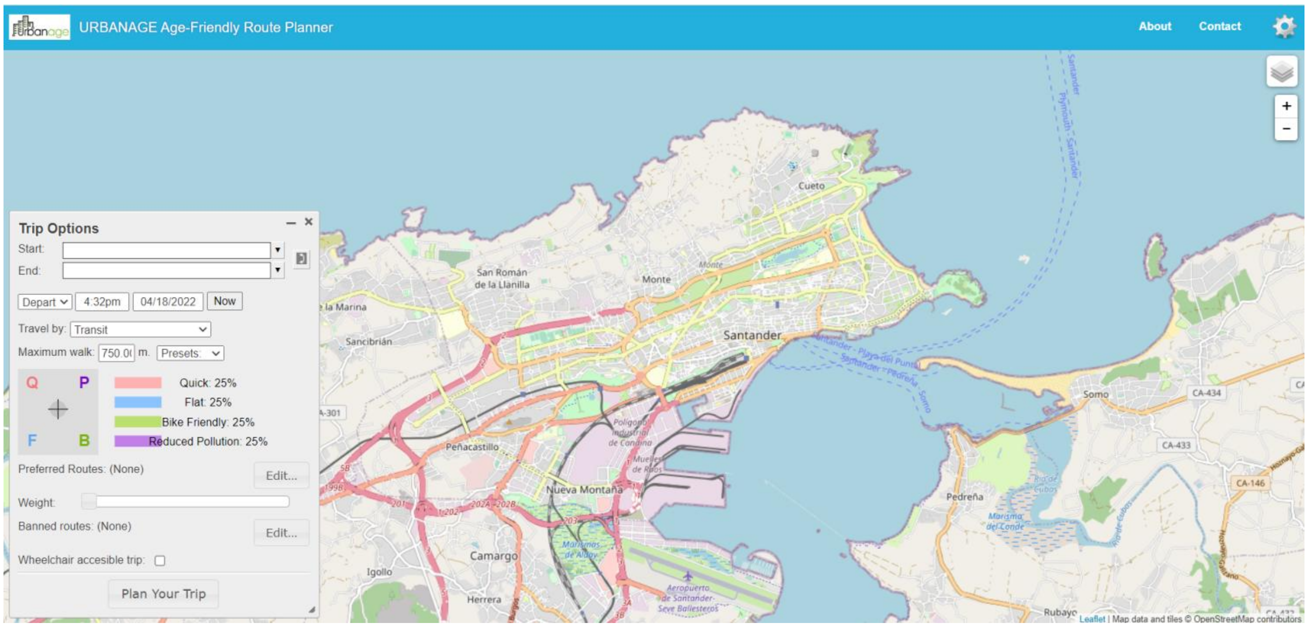
Figure 9: Spark console

3.4.3.2 Optimisation - Age friendly route planner

This module is composed by the algorithms in charge of solving optimization problems of URBANAGE. More concretely, all the algorithms and functionalities regarding the route planning system are comprised within this category.

Table 11: Age friendly route planner

Docker image	Dependencies	GitLab repository	Endpoints
registry.gitlab.com/urbanageeu/age-friendly-route-planner:latest	-	https://gitlab.com/urbanageeu/age-friendly-route-planner.git	http://ecosystem.urbanage.eu:8099/



Trip Options
Start:
End:
Depart 4:32pm 04/18/2022 Now
Travel by: Transit
Maximum walk: 750.0 m. Presets:
Q P F B
Quick: 25%
Flat: 25%
Bike Friendly: 25%
Reduced Pollution: 25%
Preferred Routes: (None) Edit...
Weight:
Banned routes: (None) Edit...
Wheelchair accesible trip: ☐
Plan Your Trip

Figure 10: Age friendly route planner UI

3.4.4 Data Management

The Data Management system allows the initial transformation of the data, its modelling and integration into a database more typical of a Big Data system. This system offers the functionalities to access, collect, aggregate, and harmonise data coming from heterogeneous sources, such as databases, Open Data Management Systems (e.g., CKAN, Socrata, etc.), exiting IT platforms (e.g., legacy systems) and sensors.

3.4.4.1 Context Broker

This module provides functionalities for the management of context information lifecycle, providing APIs to allow the interaction with IoT devices and IT platforms. It uses for its realisation the FIWARE Orion Context Broker [19]

Table 12: Context Broker

Docker image	Dependencies	GitLab repository	Endpoints
fiware/orion-ld	mongoDB	https://gitlab.com/urbanageeu/context-broker.git	NGSI/HTTP interface: http://ecosystem.urbanage.eu:1026/

3.4.4.2 Data Gateway

This module allows the federation of existing Open Data Portals (or Open Data Management Systems – ODMS) based on different technologies providing a unique access point to search and discover open datasets coming from them. Its realisation is based on Idra [20].

Table 13: Data Gateway

Docker image	Dependencies	GitLab repository	Endpoints
data-gateway_idra:latest	MySQLDB (v5.7) Sesame (RDF4J Server and RDF4J Workbench)	https://gitlab.com/urbanageeu/data-gateway.git	https://ecosystem.urbanage.eu/IdraPortal/

3.4.4.3 Data Repositories Federator

This module, working as a distributes SQL query engine, provides functionalities to perform SQL queries against different data sources. Its realisation is based on Presto [21].

Table 14: Data Repositories Federator

Docker image	Dependencies	GitLab repository	Endpoints
prestodb:latest	Postgres DB	https://gitlab.com/urbanageeu/data-repository-federator.git	https://ecosystem.urbanage.eu/ui/

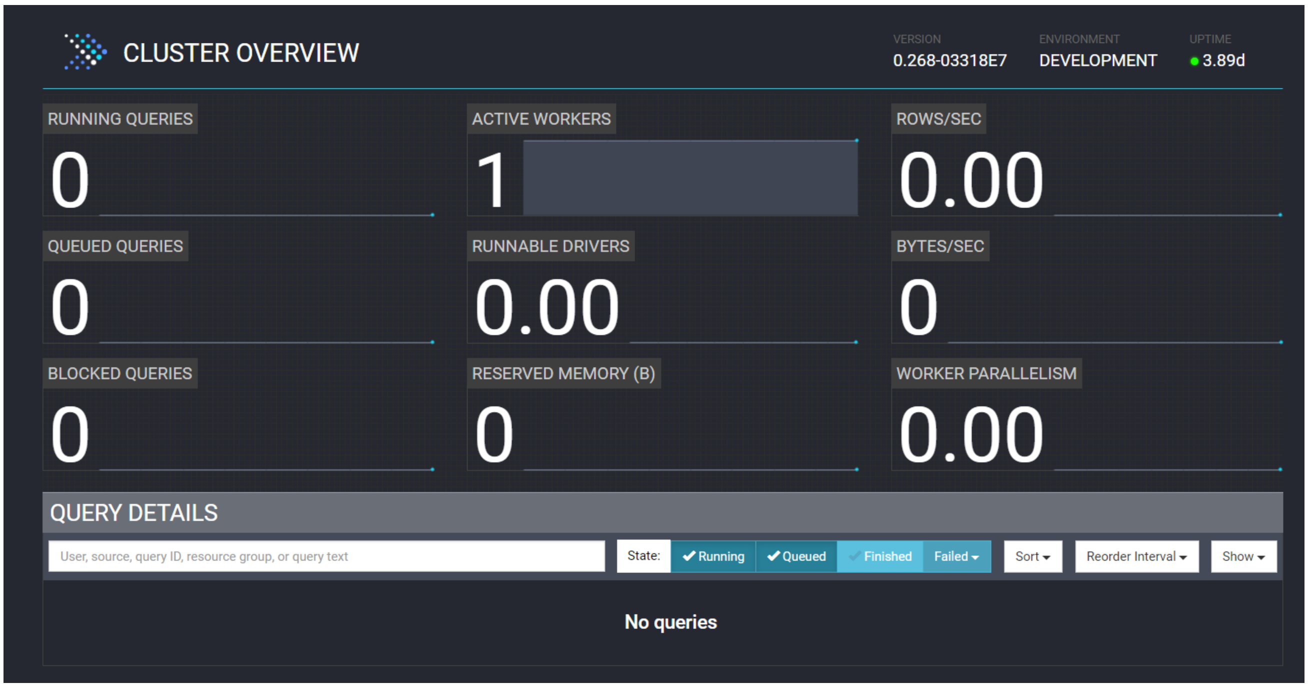


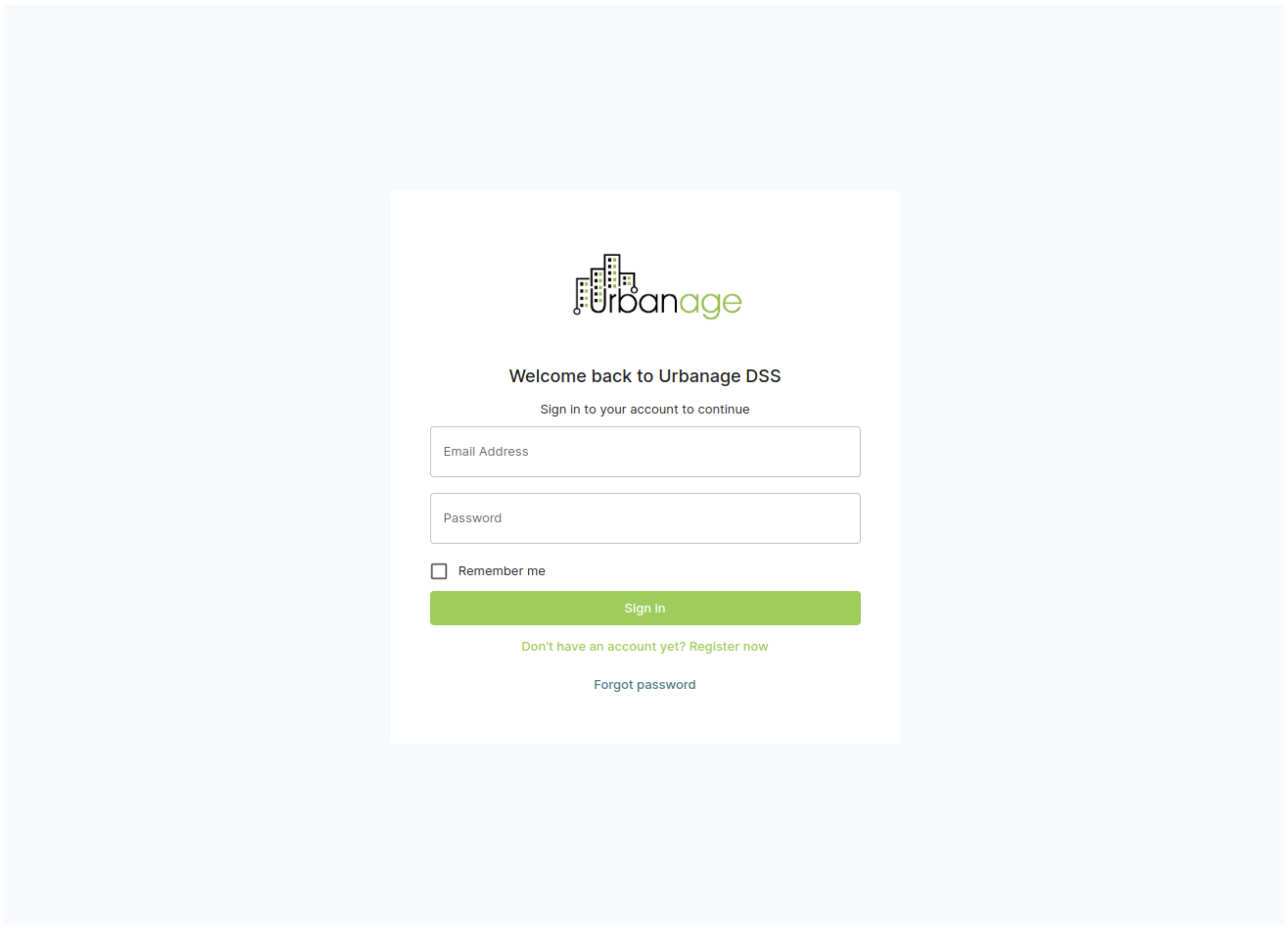
Figure 11: Data Repositories Federator UI

3.4.5 Platform UIs

An initial version of the User and Admin UIs have been implemented and deployed on the development server. These contain the first version of the container UI and the login functionality as part of the User UI and the user management page and services as part of the Admin UI. The UIs are developed in React JS [22], while the users microservice that offers user management functionalities in front of Keycloak, is developed in Java Spring Boot [23].

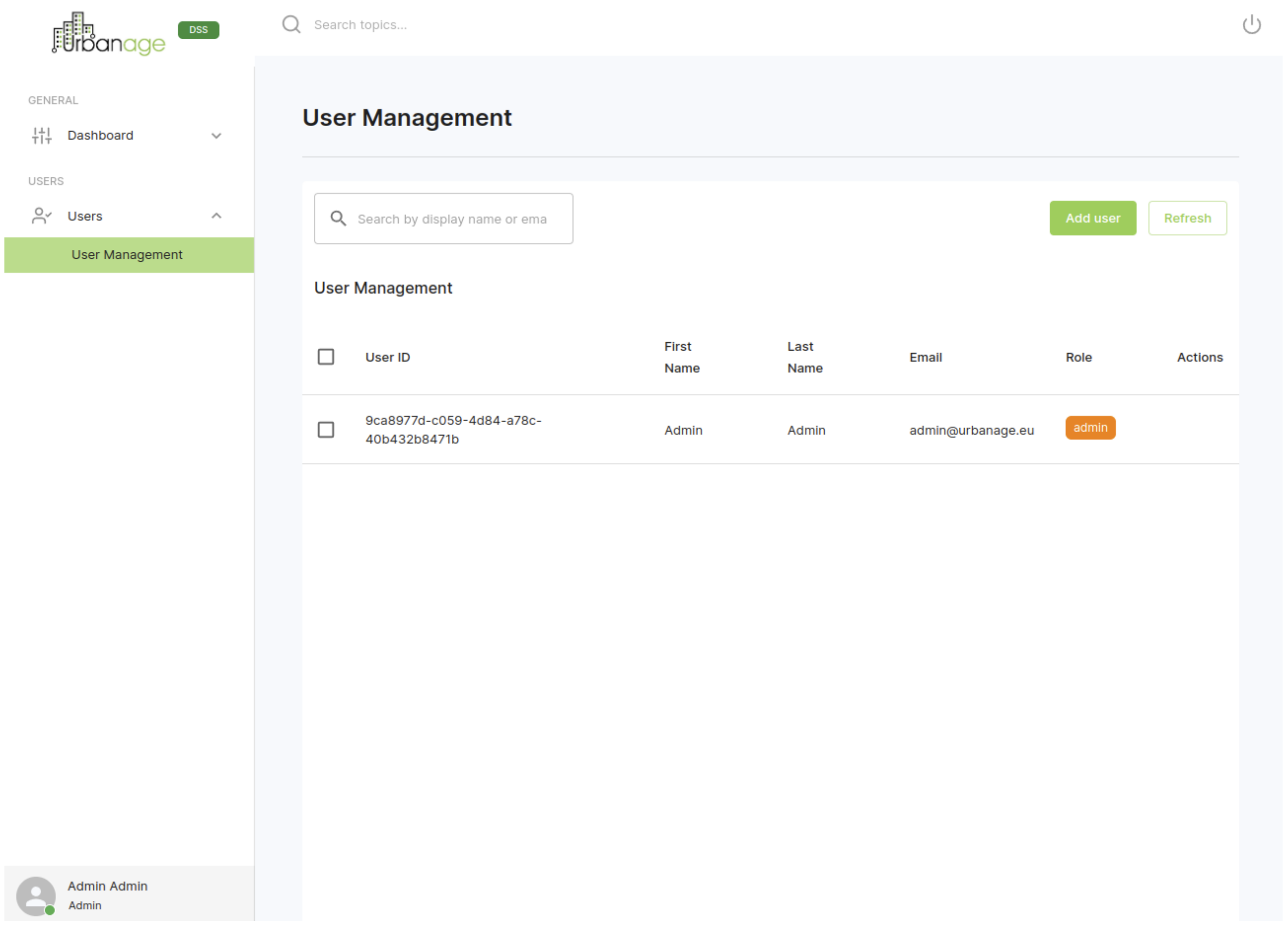
Table 15: Platform UIs

Docker image	Dependencies	GitLab repository	Endpoints
registry.gitlab.com/atceng/ilab/urbanage/urbanage-user-manager	Keycloak	https://gitlab.com/urbanageeu/user-admin-ui.git	-
registry.gitlab.com/atceng/ilab/urbanage/urbanage-dashboard	User Manager microservice	https://gitlab.com/urbanageeu/user-admin-ui.git	http://ecosystem.urbanage.eu:5000/


Welcome back to Urbanage DSS

Sign in to your account to continue

Figure 12: Platform UIs - Login page



User Management

User Management

	User ID	First Name	Last Name	Email	Role	Actions
☐	9ca8977d-c059-4d84-a78c-40b432b8471b	Admin	Admin	admin@urbanage.eu	admin	

Figure 13: Platform UIs - User registration page

4 Conclusion

The goal of this document is to describe the implementation status of the URBANAGE ecosystem as well as the main platforms and tools used for the deployment of the URBANAGE components, thus it acts as an accompanying document of the URBANAGE ecosystem release. The integration phase of the technical components that are being developed in WP3 and WP4 - in releases following the agile principles - allow for the accomplishment of a functional integrated system. The available components as well as the core integration activities of the project, as they have been designed up to now, have been presented in this document. To ensure that the implemented prototype closely follows the real needs of the end user the look and feel of the interface will be influenced by the work of WP4 in order to create a harmonised look and feel among all modules.

The tools that are deployed in the URBANAGE server and will aid the URBANAGE ecosystem in areas like communication, user authentication, monitoring etc. are: Traefik, Keycloak, Kafka, Prometheus, Grafana, SonarQube as described in section 3.3.

The components/systems installed on our development server based on M15 integration plan are: The Big Data storage, the Digital Twin System, the Big Data Analytics System, and the Data Management system, as described in detail under section 3.4.

5 References

[1] D5.1 System Architecture & Implementation Plan
[2] D5.2 Initial Platform Prototype
[3] https://www.hetzner.com/dedicated-rootserver/ax101
[4] https://www.hetzner.com/
[5] https://traefik.io/
[6] https://www.keycloak.org/
[7] https://kafka.apache.org/
[8] https://zookeeper.apache.org/
[9] https://github.com/obsidiandynamics/kafdrop
[10]https://prometheus.io/
[11]https://grafana.com/
[12]https://www.sonarqube.org/
[13]https://min.io/
[14]https://www.mysql.com/
[15]https://www.postgresql.org/
[16]https://www.mongodb.com/
[17]https://rdf4j.org/
[18]https://spark.apache.org/
[19]https://fiware-orion.readthedocs.io/en/master/
[20]https://github.com/FIWARE-GEs/idra
[21]https://prestodb.io/
[22]https://reactjs.org/
[23]https://spring.io/projects/spring-boot