This project has received funding from Horizon 2020, European Union's Framework Programme for Research and Innovation, under grant agreement No.831138

Deliverable D3.1

Initial version of Service Platform modules (software)

SCENE Project

Grant Agreement No. 831138

Call H2020-EIC-FTI-2018-2020 "Fast Track to Innovation"

Topic EIC-FTI-2018-2020– Fast Track to Innovation (FTI)

Start date of the project: 1 December 2018

Duration of the project:24 months

Disclaimer

The SCENE consortium consists of the following partners.

No.	Name	Short Name	Country
1	VISIONWARE - SISTEMAS DE INFORMAÇÃO, SA	VISIONWARE	PT
2	JCP-CONNECT SAS	JCP-C	FR
3	ALMAVIVA - THE ITALIAN INNOVATION COMPANY SPA	ALMAVIVA	IT
4	COMMISSARIAT A L'ENERGIE ATOMIQUE ET AUX ENERGIES ALTERNATIVES	CEA	FR
5	AZIENDA METROPOLITANA TRASPORTI CATANIA SPA	CAT	IT

Document Information

Project short name and number	SCENE (AMD-831138-1)
Work package	WP2
Number	D3.1
Title	
Version	V1.0
Responsible partner	ALMAVIVA
Type[1]	R
Dissemination level[2]	PU
Contractual date of delivery	30/11/2019
Last update	18/12/2019

[1]**Types.R:** Document, report (excluding the periodic and final reports); **DEM:** Demonstrator, pilot, prototype, plan designs; **DEC:** Websites, patents filing, press & media actions, videos, etc.; **OTHER:** Software, technical diagram, etc.
[2]**Dissemination levels.PU:** Public, fully open, e.g. web; **CO:** Confidential, restricted under conditions set out in Model Grant Agreement; **CI:** Classified, information as referred to in Commission Decision 2001/844/EC.

Document History

Version	Date	Status	Authors, Reviewers	Description
V 0.1	7/10/2019	Draft	ALM	Initial version, TOC definition and first contents
V 0.2	26/12/2019	Draft	ALM	ALM contribution to Service Platform
V 0.3	03/12/2019	Draft	JCP-C	JCP contribution to Service Platform
V 04	04/12/2019	Draft	ALM	Version for Review
V 0.5	16/12/2019	Draft	VIS	Reviewed
V 1.0	18/12/2019	Final	ALM	Final release

Assinado Por: FILIPE MIGUEL FERNANDES GOMES CL
Localização: Lisbon, Portugal
Motivo: I approve this document
Data da Assinatura: 23/12/2019 11:21:44 GMT +00:00

Acronyms and Abbreviations

Acronym / Abbreviation	Description
API	Application Programming Interface
CC	Cache Controller
CAS	Content Access Statistic
CDN	Content Delivery Network
CM	Content Management
CUM	Content User Management
GUI	Graphical User Interface
HTTPS	Hypertext Transfer Protocol Secure
IDS	Intrusion Detection System
IGW	Intelligent Gateway
IM	Infrastructure Management
IoT	Internet of Things
KPI	Key Performance Indicators
LoRa	Long Range
MQTT	Message Queuing Telemetry Transport
PM	Person Month
PMC	Project Management Committee
QoE	Quality of Experience
RESTful	Representational State Transfer ful
SCENE	Smart City on the Edge Network Enhancements
SP	Service Platform
UI	User Interface
UC	Use Case
VQM	Video Quality Metric

Contents

Figures

1 INTRODUCTION

Security, utility management (water, electricity, etc.), transportation, smart communities, smart cities and many other sectors are influenced by the technology evolution in data analytics and its various interesting applications. However, the availability of data itself in real-time is dependent on the communication technology and its underlying infrastructure.

To overcome the challenge of high initial investment in traditional communication systems that allows to collect data for smart environments (cities, communities, groups, services, etc.) the Internet of Things (IoT) is introduced as a low-cost solution. The low cost of sensors and the wide variety of supported applications encouraged the adoption of the technology in several cases.

Collecting and processing secured data in real-time is a major challenge, but not the only one. Data analysis and consequential response reactions - which varies from simple signalling up to huge content delivery - are other major challenges that need processing power for analytics and transmission bandwidth for potential content. In many cases the response reaction may involve configuration setup, documents/multimedia transmission, etc. that might be beyond the capacity of the used IoT systems, and as a result complementary systems/implementations should be deployed to perform this transmission tasks.

SCENE (Smart City on the Edge Network Enhancement) is proposing an integrated solution based on vehicular networks for a secured environment that securely collects, and processes sensor data based on IoT technologies. This data is transmitted centrally to be analysed and processed by smart-city applications. In the other direction, the system also supports a content delivery platform that deploys a suitable protocol stack for secured content delivery. With this functionality, SCENE is building an integrated security system for both IoT based data collection as well as multimedia secured content delivery. The content delivery platform is intelligent enough to be deployed in both IoT mode and in the stand-alone mode to deliver contents to subscribed smart applications.

This deliverable presents the first version of the Service Platform software components realized in the Work Package WP3. In the following weeks the platform will be tested in the 1st Pilot Test campaign to be held in Italy, Portugal and France.

2 ARCHITECTURE OVERVIEW

This chapter recalls the general architecture of the SCENE Platform with a brief description of the main modules from a functional point of view, as presented in Deliverable D2.2 "Initial version of SCENE Modules Design"

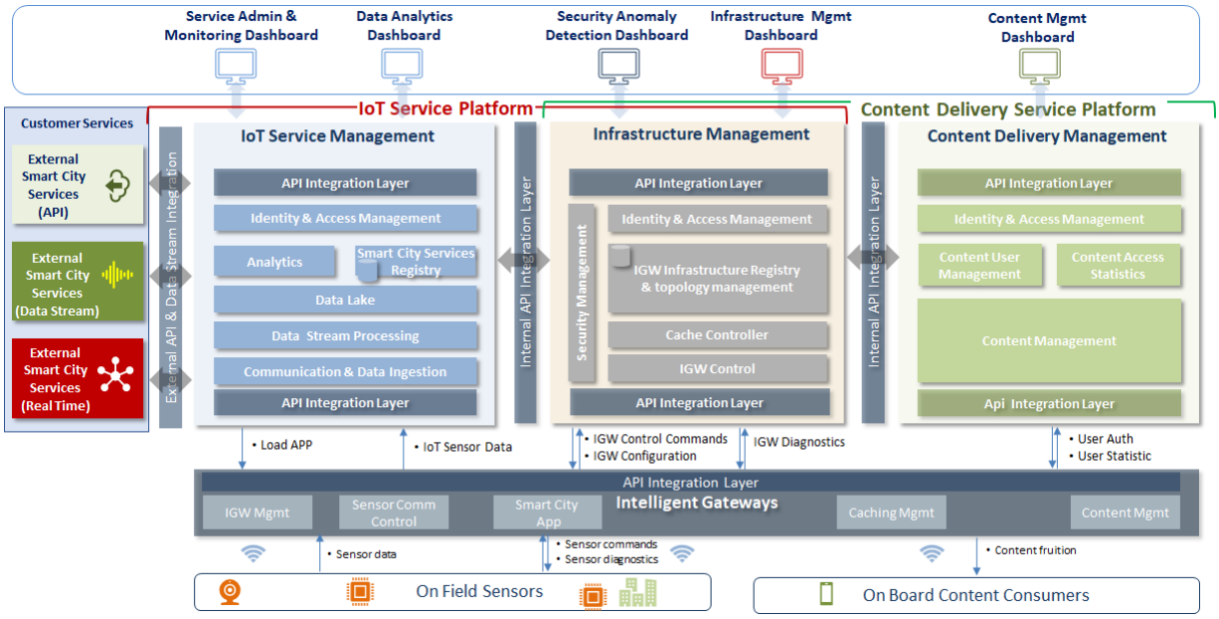

Figure 1 - SCENE Platform Architecture

2.1 Service Platform

The Service Platform is the central core of the SCENE platform. From a functional point of view, it is composed by the following main components:

a. **IoT Service Management**: this component is in charge of all the aspects related to the IoT functionalities such as Smart City Services and Sensor Registry Management, IoT Data Ingestion, Data Stream Processing, IoT Data Storing, Analytics Management for IoT platform monitoring, Identity and Access Management for IoT platform

b. **Infrastructure Management**: this component has the aim of managing the Network Infrastructure, the Intelligent Gateway registry and configuration, the IGW software delivery and the IGW setup for sensor's management (e.g. software libraries, sensor vendor tools). It manages also the installation and configuration of the Apps that could be provided by the External Partners for the Edge Processing purpose. This module provides also the Intrusion Detection functionalities implemented by the sub-module Security Management. It is also the house for hosting common general modules useful for all the platform

c. **Content Delivery Management**: This component takes care of all the aspects of content delivery to users in the transport system, including user registration, content delivery, caching, prefetching, communication with the Cache Controller.

d. **Interface Layer: SCENE Service Platform implement an integration layer which allows interaction** with others component of SCENE Platform and with external systems accessing data and services exposed by the Service Platform through API paradigm.

2.2 Intelligent Gateway (IGW)

This component has the aim to collect sensor data at the edge of the network, process it through local applications and send it to the service platform. Additionally, the Intelligent Gateway ensures content delivery to the edge users with caching and content pre-fetching services. It also provides Intelligent management such as configuration, provisioning and diagnostics.

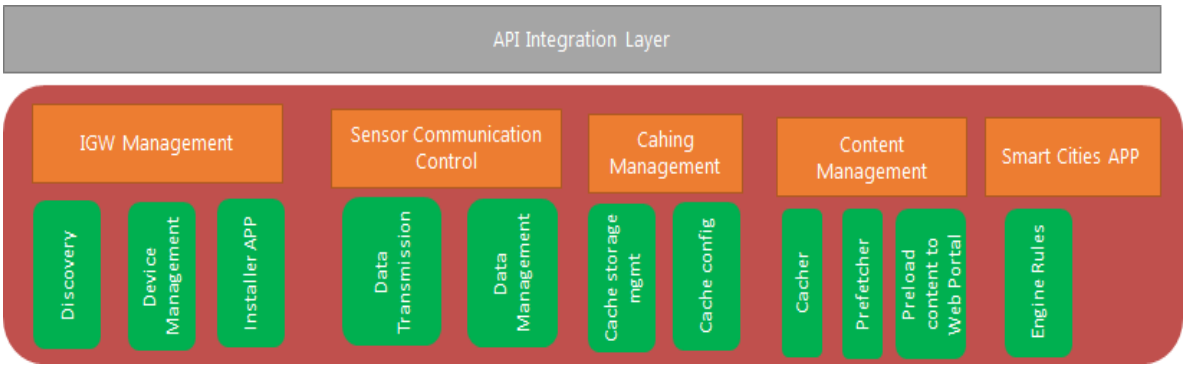

Figure 2 - Intelligent Gateway modules

The Intelligent Gateway has the following core components:

IGW Management: It manages different functions related to IoT sensors and content delivery. Additionally, it is responsible to perform different functions, such as IoT sensor discovery, device management (status info, configuration, etc.), and install/update new APP.

Sensor Communication: This component performs actions related to sensor communication. It collects and transmits the IoT sensor data to the IoT service platform and provides the IoT sensor status via a MQTT API. It manages the data related to IoT sensors.

Caching Management: It is mainly related with the caching content, as it manages the local storage for caching and configure caching stuff. The main objective is to decide about which content should be cached, and which cached content should be removed in order to make the storage space available for new cached content.

Content Management: It is a component that manages the concern actions related to content. It caches the content, executes the prefetch instruction from the Cache controller (CC), and preloads the content on the web portal.

Smart Cities App: It loads and manages the APP that belongs to external smart cities service entities. It registers, configures and defines the engine rules for the external APP (e.g. run application on scheduled timer, alerts & notify update to the system, etc.)

2.3 Dashboard

This is the main GUI to access and manage the SCENE Platform. It logically collects specific Dashboards for each pillar described above. These dashboards will provide dedicated modules for the human interaction with the related platform functionalities.

The structure of the User Roles will reflect the structure of the Dashboards and, as general rule, there will be a role for each Dashboard section.

Here below the list of the main dashboard sections:

- Service Admin & Monitoring Dashboard: it allows the system administration and the monitoring of the whole platform;
- Data Analytics Dashboard: it allows the visualization of the data analytics calculated on the IoT sensors data received by the platform (I.e. for the specific use cases);
- Infrastructure Management: it allows the management and visualization of the Intelligent Gateways configuration and network parameters;
- Content Management: it allows the monitoring and management of the Content Management platform.
- IDS Dashboard: It provides the user interface to display security analytics related to smart city protection by the IGW and gateway self-protection.

3 SERVICE PLATFORM SOFTWARE V.1.0

This chapter presents the software implemented in the 1st version of the SCENE Platform, regarding the Central Service Platform, included API management and security features for enabling the execution of the 1st phase of pilot tests. The implementation phase has been conducted and is based on the previous WP2 "Requirements Elicitation, System Specification".

3.1 IoT Service Management

The main components of the IoT Service Management realized in the first version of the platform are the following:

- **IoT Data Ingestion**: the software component that manages the ingestion of sensors measures which arrive at the platform through the Intelligent Gateways;
- **IoT Data Processing**: the component that manages the processing and the storing of the IoT data coming from sensors via Intelligent Gateways;
- **IoT Registry**: a relational database that keeps the registry of the IoT Platform elements like Sensors, Intelligent Gateways, Observed Properties, etc.
- **Data Lake**: two non-relational databases where large amounts of data, composed essentially by *enriched* sensor measures, are stored and accessed with small latency times;
- **Custom API Modules**:
    - **Scene-sm-backend**: Custom API Module for the access to the IoT Registry database from the Service Management Dashboard;
    - **Scene-iop-backend**: Custom API Module for the access to the IoT Registry from the Infrastructure Management subsystem;
    - **Scene-data-backend:** Custom API Module that manages the access to the IoT data for the Smart City Service.
- **Identity Access Manager**: module that manages the users, the roles and the access rights to the IoT resources;
- **API Manager**: module that manages the usage of the API interfaces implemented in the IoT Service Platform;

3.1.1 IoT Data Ingestion

The IoT Data Ingestion is the functionality that allows the acquisition in the platform of the IoT data coming from the sensors through the Intelligent Gateways.

In the first release of the Service Platform, the 1st step of this functionality has been implemented using an MQTT Broker component, based on the standard services "Publish&Subscribe". They have been configured on the dedicated SSL/TSL port (8883) and with the subject mqtt.SCENE.

All the IGWs must send information to the IoT Service Platform following a standard message template that has been defined not only for the IoT data but for a generalized interaction. In this part of the project, the general message format is designed for data coming from sensors to Service Platform.

On the external side, the different Intelligent Gateways send the IoT messages with the sensor measurements to the MQTT broker through the "Publish" operation on the "mqtt.SCENE" subject.

On the internal side, a dedicated software component extracts the data from the MQTT broker through the "Subscribe" operation on the same "mqtt.SCENE" subject and dispatches such messages to the implemented processing components dedicated to acquire the IoT data.

The implemented authentication mechanism for the connection of the Intelligent Gateways towards the IoT Service Platform MQTT Broker is based on username and password credentials. Moreover, the IGWs have been provided with the Certification Authority certificate that allow the SSL/TLS connection.

3.1.2 IoT Data Processing

The processing of the messages coming from the Intelligent Gateways consists essentially in the IoT data enrichment and the storing of the processed data in the data layer.

The component that has been implemented in this phase for the enrichment functionality gets the IoT messages from the MQTT broker with the Subscribe operation. Then it joins these messages with the IoT registry data loaded from the dedicated relational database. The result is the production of one expanded record for each measure coming from the sensors.

Such expanded records contain all the information needed for the analytics functions and makes each record self-consistent with no need of external references.

Finally, this process stores the enriched data in two different non-relational databases: one (HDFS) dedicated to the analytics functionalities and another (Elasticsearch) used for the access to the IoT data from the customers (each customer can apply the specific processing to his data by himself).

3.1.3 IoT Registry

The registry of the IoT elements is stored in a relational database (MySql) which contains the information about the main entities involved in the IoT management (Sensors, Observed Properties, Locations, …).

The tables Sensors, ObservedProperty and Gateway of this database are populated by the IoT Infrastructure Management subsystem via specific API.

All the data contained in this registry are utilized for the enrichment of the IoT sensor measures coming from sensors via the Intelligent Gateways.

Here below the data model diagram of the registry database that has been implemented so far, as initially described in deliverable D2.2 “Initial version of SCENE Modules Design”:

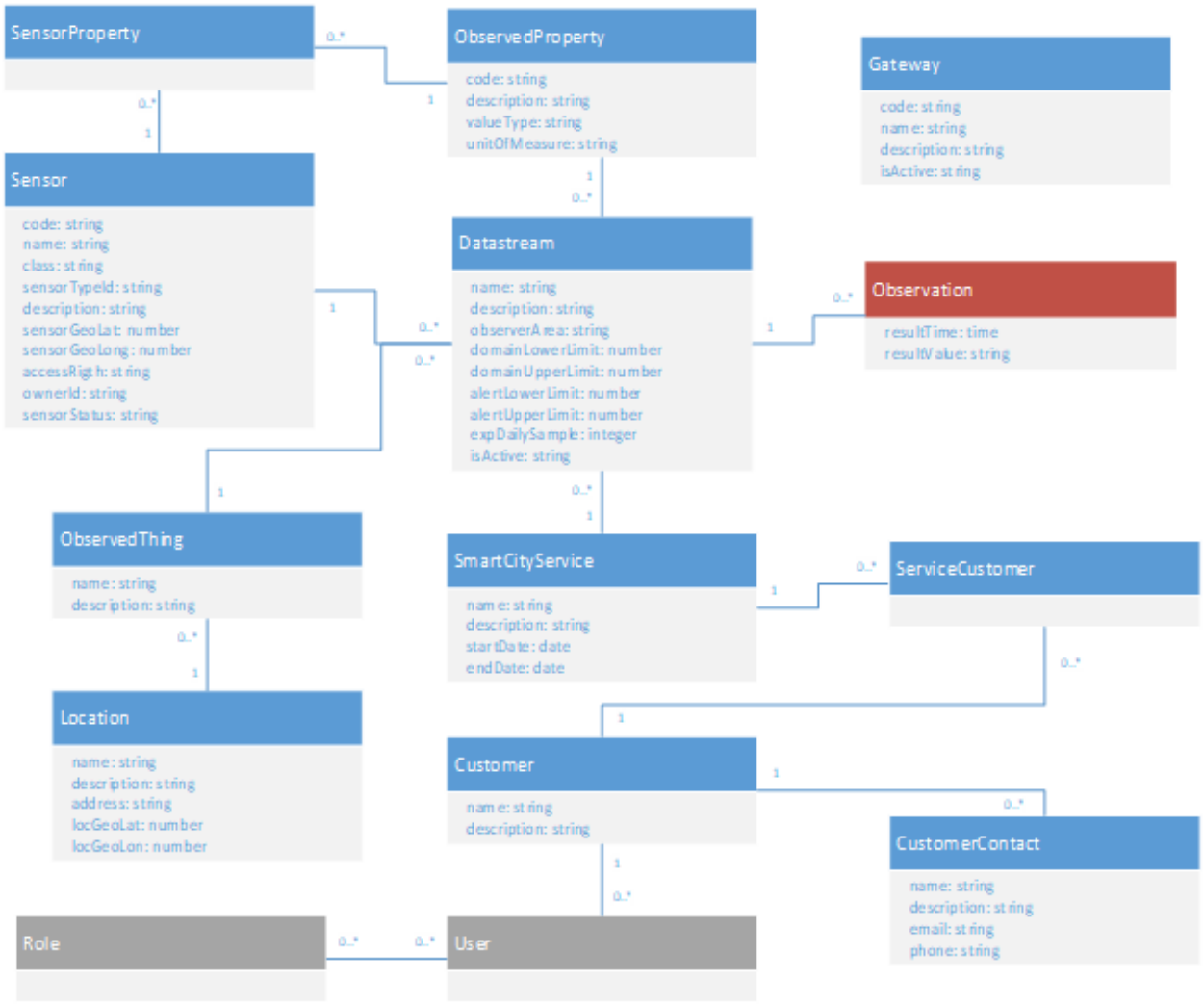

Figure 3 – Data Model

Each User is characterized by the Role and the Customer (organization) to which it belongs to. The Role identifies which functionalities are enabled for him in the Service Management dashboard. The Customer (organization) attribute allows the segregation of the owned data through the Smart City Services entities.

In the following picture is shown the related subpart of the data model:

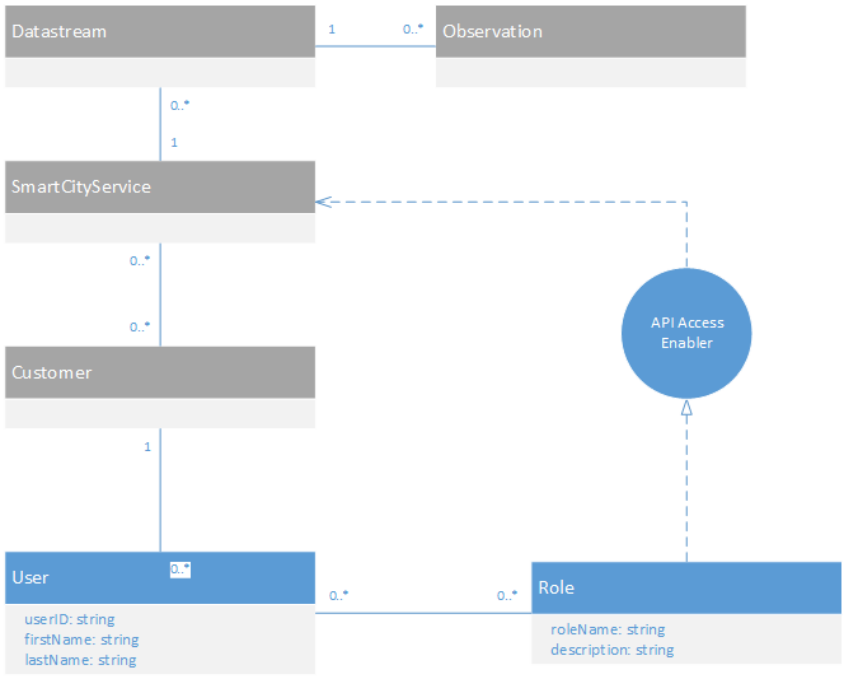

Figure 4 – Part of the Data Model regarding Users

3.1.4 Data Lake

The enriched IoT data (sensor measures) are stored in two non-relational databases in the flat format where each record contains all the needed information in terms of consistency and completeness, necessary for fast data analysis (analytics). Each record is self-consistent and can be elaborated with no need of joining other records or tables, resulting in faster output.

The non-relational databases utilized in this phase are:

- HIVE system on HDFS: it is used for the data accessed from the from the IoT Analytics Module; the inserted data are partitioned basing on the year-month-day of each record;
- Elasticsearch: it is used for the data access to the IoT data specifically related to Customer Smart City Services via API

Here below the list of the fields of the enriched record as for the current implementation of the data lake:

messagetime	datastream_expdailysamples
sourceid	datastream_isactive

sourcetype	smart_city_service_id
sourcelat	smart_city_service_name
sourcelng	smart_city_service_description
destinationid	smart_city_service_startdate
payloadtype	smart_city_service_enddate
sensorid	sensor_id
sensorlat	sensor_code
sensorlng	sensor_name
measurepropertyid	sensor_class
measuretime	sensor_sensortypeid
measurevalue	sensor_description
gateway_id	sensor_accessright
gateway_code	sensor_ownerid
gateway_name	sensor_sensorstatus
gateway_isactive	observed_thing_sid
observed_property_unitofmeasure	observed_thing_name
datastream_id	observed_thing_description
datastream_name	location_id
datastream_description	location_name
datastream_observedarea	location_description
datastream_domainlowerlimit	location_address
datastream_domainupperlimit	location_locgeolat
datastream_alertlowerlimit	location_locgeolong
datastream_alertupperlimit	

Such enriched record is also referred as *Datastream*.

3.1.5 Custom API Modules

A set of API modules has been developed for the access to both IoT Registry and IoT data. They expose externally the related resources and methods for the specific data access operations. Internally, they connect to the relational database (MySql) and non-relational database (Elasticsearch).

The authentication mechanism implemented for these API is based on the OAuth2 – Open Id Connect protocol. Before the call to the APIs, the clients must provide the credentials to the Identity Server in order to obtain an Authentication Token that must be provided in the following APIs call.

The token managed with this protocol contains additional information about the Customer (organization) to which the user belongs to. In this way the API backend verifies that the Customer effectively owns the requested Service and provides as output only the data that belong to the customer specified in the token.

In the following paragraphs, these API modules are briefly described.

Scene-sm-backend

This custom API Module has been developed for the management of the IoT Registry from the Service Management Dashboard. The main purpose of these APIs is the implementation of the CRUD operations on the IoT Registry objects (Locations, Datastreams, Sensors, …).

Here below the list of the APIs of this module:

METHOD	RESOURCE	Description
POST	/scene/1.0/customers	Creates a new Customer
PUT	/scene/1.0/customers	Updates a Customer
DELETE	/scene/1.0/customers	Deletes a Customer
GET	/scene/1.0/customers/all	Returns all Customers
GET	/scene/1.0/customers/ids/*	Returns the Customers by ID list
POST	/scene/1.0/datastreams	Creates a new Datastream
PUT	/scene/1.0/datastreams	Updates a Datastream
DELETE	/scene/1.0/datastreams	Deletes a Datastream
GET	/scene/1.0/datastreams/all	Returns all Datastreams

GET	/scene/1.0/datastreams/ids/*	Returns the Datastreams by ID list
GET	/scene/1.0/home/stats	Returns home statistics
POST	/scene/1.0/locations	Creates a new Location
PUT	/scene/1.0/locations	Updates a Location
DELETE	/scene/1.0/locations	Deletes a Location
GET	/scene/1.0/locations/all	Returns all Locations
GET	/scene/1.0/locations/ids/*	Returns the Locations by ID list
GET	/scene/1.0/sensors/all	Returns all Sensors
GET	/scene/1.0/sensors/ids/*	Returns the Sensors by ID list
GET	/scene/1.0/observed-properties/all	Returns all Observed Properties
GET	/scene/1.0/observed-properties/ids/*	Returns the Observed Properties by ID list
GET	/scene/1.0/observed-properties/sensor/{ids}	Returns the Observed Properties related to Sensor
POST	/scene/1.0/observed-things	Creates a new Observed Thing
PUT	/scene/1.0/observed-things	Updates an Observed Thing
DELETE	/scene/1.0/observed-things	Deletes an Observed Thing
GET	/scene/1.0/observed-things/all	Returns all Observed Things
GET	/scene/1.0/observed-things/ids/*	Returns the Observed Things by ID list
POST	/scene/1.0/smart-city-services	Creates a new Smart City Service
PUT	/scene/1.0/smart-city-services	Updates a Smart City Service
DELETE	/scene/1.0/smart-city-services	Deletes a Smart City Service
GET	/scene/1.0/smart-city-services/all	Returns all Smart City Services
GET	/scene/1.0/smart-city-services/ids/*	Returns the Smart City Services by ID list

Scene-iop-backend

This custom API Module manages the access to the IoT Registry from the Infrastructure Management subsystem in order to let such subsystem to synchronize the data layer with the information regarding new Sensors, IGWs, Sensor Types, Observed properties, etc.

Here below the list of the APIs of this module:

METHOD	RESOURCE	Description
POST	/scene-iop/1.0/sensors	Creates a new Sensor
PUT	/scene-iop/1.0/sensors	Updates a Sensor
DELETE	/scene-iop/1.0/sensors	Deletes a Sensor
GET	/scene-iop/1.0/sensors/all	Returns all Sensors
GET	/scene-iop/1.0/sensors/ids/*	Returns the Sensors by ID list
POST	/scene-iop/1.0/observed-properties	Creates a new Observed Property
PUT	/scene-iop/1.0/observed-properties	Updates an Observed Property
DELETE	/scene-iop/1.0/observed-properties	Deletes an Observed Property
GET	/scene-iop/1.0/observed-properties/all	Returns all Observed Properties
GET	/scene-iop/1.0/observed-properties/ids/*	Returns the Observed Properties by ID list
POST	/scene-iop/1.0/gateways	Creates a new Gateway
PUT	/scene-iop/1.0/gateways	Updates a Gateway
DELETE	/scene-iop/1.0/gateways	Deletes a Gateway
GET	/scene-iop/1.0/gateways/all	Returns all Gateways
GET	/scene-iop/1.0/gateways/ids/*	Returns the Gateways by ID list

Scene-data-backend

A specific software component has been developed in this phase in order to provide access to IoT data through API Restful mechanism. This API takes in input the Smart City Service name and returns only the data related to that Service; this will guarantee the segregation of the data for the Customer.

This module takes the data from the Elasticsearch database and is called by the Customer client applications.

Here below the list of the APIs of this module:

METHOD	RESOURCE	Description
GET	/scene-data/1.0/data_by_service/{ServiceId}	Returns the IoT data related to the input Service Id

3.1.6 API Manager System

All the implemented Restful APIs in the Scene project (Scene-sm-backend, Scene-iop-backend and Scene-data-backend) have been managed using the WSO2 API Manager product.

This product implements the presentation layer of the APIs, allows the management of the access rights based on the user roles and keeps the connections and interactions with the backend processes that physically implement the APIs.

3.2 Infrastructure Management

Infrastructure Management (IM) is responsible for managing the Intelligent Gateway (IGW) network by considering IGW registry and configuration, IGW software delivery, and IGW setup for sensor's management (e.g. software libraries, sensor vendor tools). In addition, it manages the installation and configuration of the Applications in the IGW provided by the External Partners. The Security Management is a sub-module of IM, which implements the Intrusion Detection functionalities. The IM also consist of multiple components that manages the whole platform.

3.2.1 Cache Controller (CC)

The CC is a key element of Infrastructure Management which is responsible for analysing user's content requests and making optimal caching and prefetching decisions. The CC contains different logical blocks which perform required functions such as Bloom Filter (BF), Prefetching, Network Topology, etc. Each IGW has two cached repositories: Cache and Prefetch. Based on the cached content reported by IGWs, the popularity of could be calculated, and later, CC makes the prefetching decision that indicates the corresponding IGW, about which content should be prefetched. Any operation of each cache (including cache and prefetch repository), like adding a chunk and removing a chunk of video content, should be recorded in CC's in the form of database (DB). Each user request also needs to be sent to CC and be stored in a DB, which will be used for chunk or file popularity calculation. CC also communicates with content delivery management via internal API to perform the necessary operations. The IGWs communicate with Cache Controller (CC) via RESTful APIs, and CC gets all required information from IGWs, as shown below.

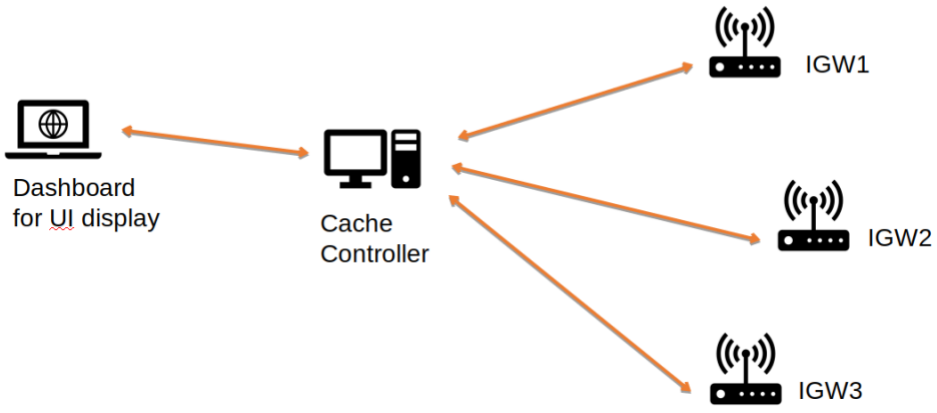

Figure 5 - Communication between IGWs and CC

Based on collected information at CC from various components of IGWs, the following features are available to display in the dashboard:

- Display IGW network topology
- Display network traffic on IGW
- IGW configuration
- Caching management in IGW

3.2.2 IGW registry and topology management

The IGW registry module adds the IGW to the networks and updates it in the network topology. The IGW registration process requires important information which are Identification (ID), type, IP address, Caching capacity, Caching algorithm, Connect/active network, status (active/inactive), and other important fields related to IGW.

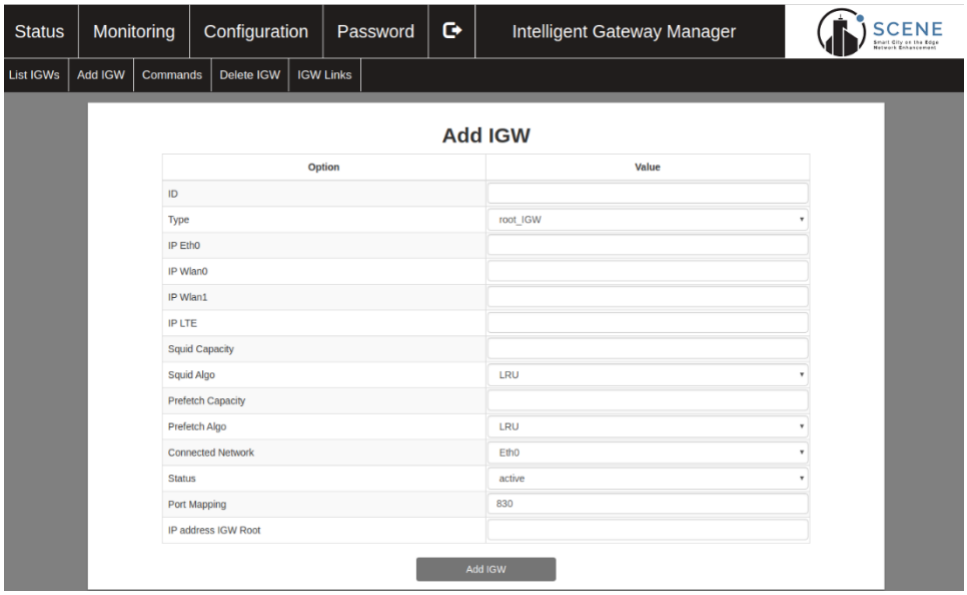

Figure 6 – Register/Add IGW

After registering the IGW, the network topology module displays the IGW in the topology layout.

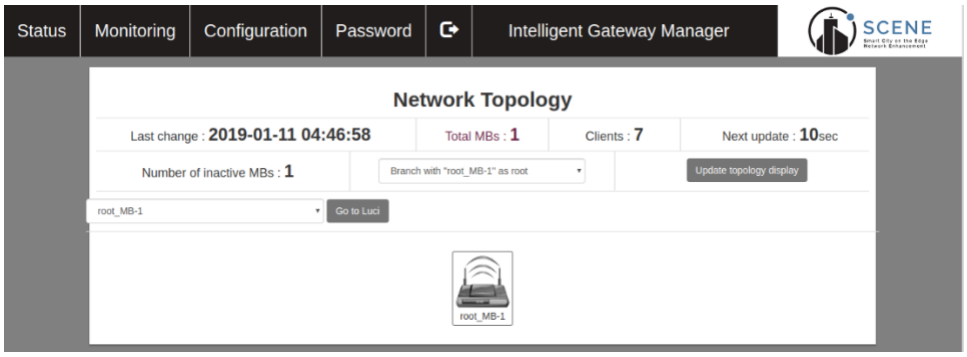

Figure 7 – Network Topology

3.2.3 IGW control

The IGW control module is responsible for configuring and monitoring the IGW, as it configures the software as well as hardware components of the IGW. Currently, some software configuration related to content caching module is available that can set/get the configuration of caching module of the selected IGW. The configuration parameters are Cache Memory Size, Cache Replacement policy (e.g. LRU), etc.

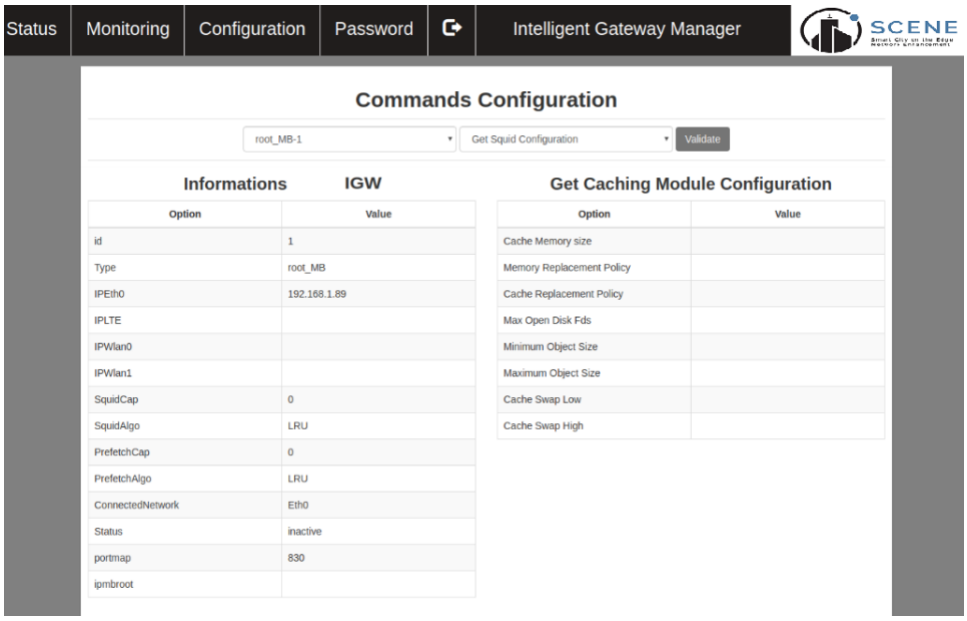

Figure 8 – Configuration of the IGW Caching Module

3.3 Security

3.3.1 Certification Authority

A unique self-signed Certification Authority has been created to be used by all pillars of the SCENE Platform. This Certification Authority has been used for the signing of the platform server certificates and then the Certification Authority certificate has been exported to the clients.

For example, it has been utilized for the signature of the MQTT Broker server and it has been exported to the MQTT clients (Intelligent Gateways) that must access the Broker over the secured SSL/TLS channel.

Here below the CA certificate snapshot:

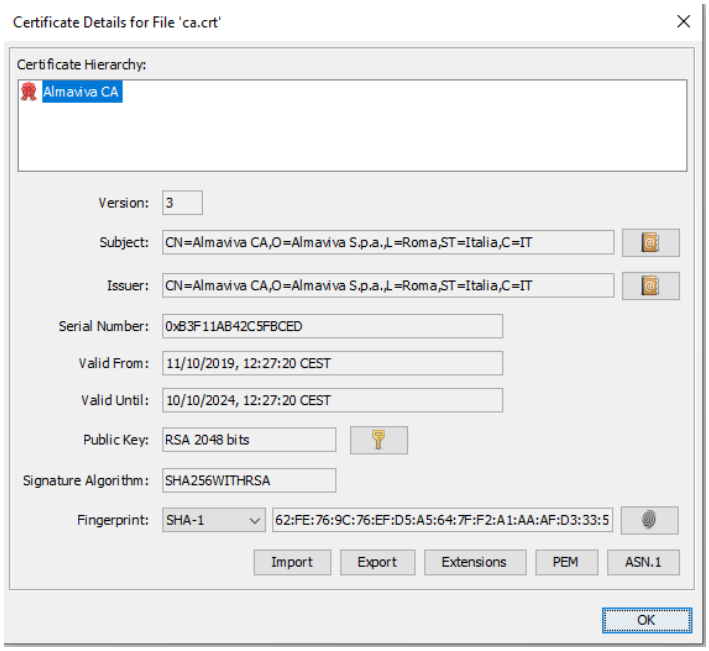

Figure 9 – CA Certificate snapshot

and the MQTT Broker certificate, server signed by the above CA:

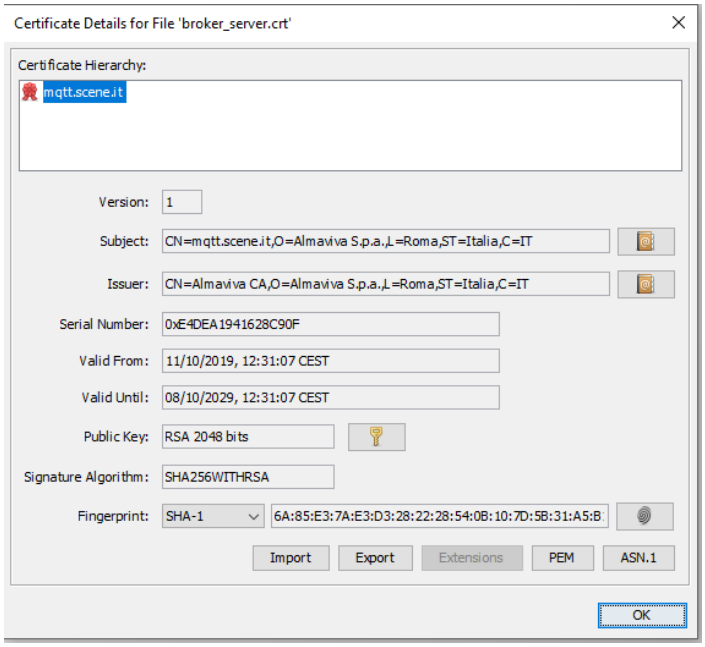

Figure 10 – MQTT Broker Certificate snapshot

3.3.2 Identity and Access Management

IoT Service Management

For the IoT Service Management, the Identity and Access Management functionalities have been implemented using specific Open Source product "WSO2 Identity Server". In particular, the access from clients to different sections of the system and the communication between the subsystems have been implemented through the API Restful protocol. The user authentication and the access to these Restful APIs is regulated by a token mechanism obtained from the Identity Server component. The token management is based on the "OAuth2 – Open Id Connect" protocol.

The implemented roles and users are:

SCENE_ADMIN_ROLE: SCENE Platform Administrator enabled for the full operation management and access to all data. ADMIN_ROLE will be generally associated to users belonging to the owner (tenant) of the platform such as the municipality of Catania, or Rennes, or Cascais. Full access (CRUD operation) is granted to all data (also those belonging to other customers).

CUSTOMER_ADMIN_ROLE: on the Scene platform, the Customers are organizations which can use SCENE to implement their own Smart City Services, for example by deploying their own sensors, or using already implemented smart services in order to build on top their own applications using SCENE APIs.

So, this role is assigned to users, belonging to a customer organization, having full access rights to create and configure smart city services.

Full access (CRUD operation) is granted only to owned data (their own smart city services - and related datastreams - and owned sensors) identified by the ownerId attribute. No visibility is granted to data belonging to other customers.

CUSTOMER_USER_ROLE: this role is assigned to users, belonging to a customer organization, having only read only access to customers' owned services.

On the MQTT Broker side, the component that accepts the messages with sensor measures from the Intelligent Gateways, two users have been defined:

- **scene_user_igw**: this user has the "publish" and the "subscribe" rights (read/write user) on the subject MQTT.scene. It is utilized for the authentication of the Intelligent Gateways for the connection to the MQTT Broker
- **scene_user_subscriber**: this user has only the "subscribe" rights (read user) on the subject MQTT.scene.

In the final version of the Scene project, the Identity Server utilized in the Service Management subsystem will be federated with the Identity Server provided for the Content Management subsystem.

Infrastructure Management

This component is not present in the first version of the platform.

3.3.3 Intrusion Detection System Manager

This component is not present in the first version of the platform.

3.4 Dashboard

The Dashboard module is the main Web GUI (Graphical User Interface) that permits a human user to interact with the SCENE platform. It offers functionalities typical of the SCENE Platform to manage IoT Services and Content Delivery system. It also offers administrative functions for the setup, configuration, and monitoring of the Platform itself. It has been developed using Angular technologies and includes the HTML and CSS tools.

The access to this GUI, and the related enabled functionalities, is regulated by the roles SCENE_ADMIN_ROLE, CUSTOMER_ADMIN_ROLE and CUSTOMER_USER_ROLE explained above.

The following paragraphs describe the various sections of the implemented dashboard.

3.4.1 IoT Service Management dashboard

The IoT Service Management dashboard is the GUI section utilized for the management of the IoT elements of the platform. It allows the data-entry and the management of the IoT Registry for the main elements of the platform (Locations, Sensors).

For example, for each couple <Sensor – Observed Property>, this dashboard section allows the definition of the related data stream where some specific parameters can be specified (lower and upper measure limits, …).

The “IoT Service Management” dashboard implemented in this phase provides the functionalities that allow to register and configure the Smart City Services, the Realm objects (Locations) and the Observed Things. The Sensors and the Observed Properties are defined by the “Infrastructure Management” but they can be visualized in this dashboard and they can be used for the Smart City Service definition.

Moreover, it provides the links for the main dashboard sections: IoT Service Management, Infrastructure Management, IoT Data Analytics, Content Management, Operation Administration and Security Analytics. In the current version only the links to the IoT Service Management and IoT Data Analytics have been implemented.

Smart City Service Management

This functionality allows the registration and configuration of the Smart City Services entities (e.g. Infrastructure Monitoring, Parking Control, Water Monitoring)

The definition of a Smart City Service involves the following steps:

- Define the Smart City Service information like name, description, start date
- Create the associated Datastreams; the main key of each Datastream is the related couple <Sensor – Observed Property> that defines exactly the stream of measures; these objects can be selected from the list of configured elements; they contain also other attributes like description, domain limits, etc.
- From a pop-up list, associate the Customers that own the Smart City Service that we are creating. These customers will have access to the IoT data produced for this Smart City Service.

On all the objects above:

- the users with CUSTOMER_USER_ROLE have only the visualization rights
- The users with CUSTOMER_ADMIN_ROLE have read/write rights

Here below the snapshot of the Service Management Dashboard with the details about one Smart City Service:

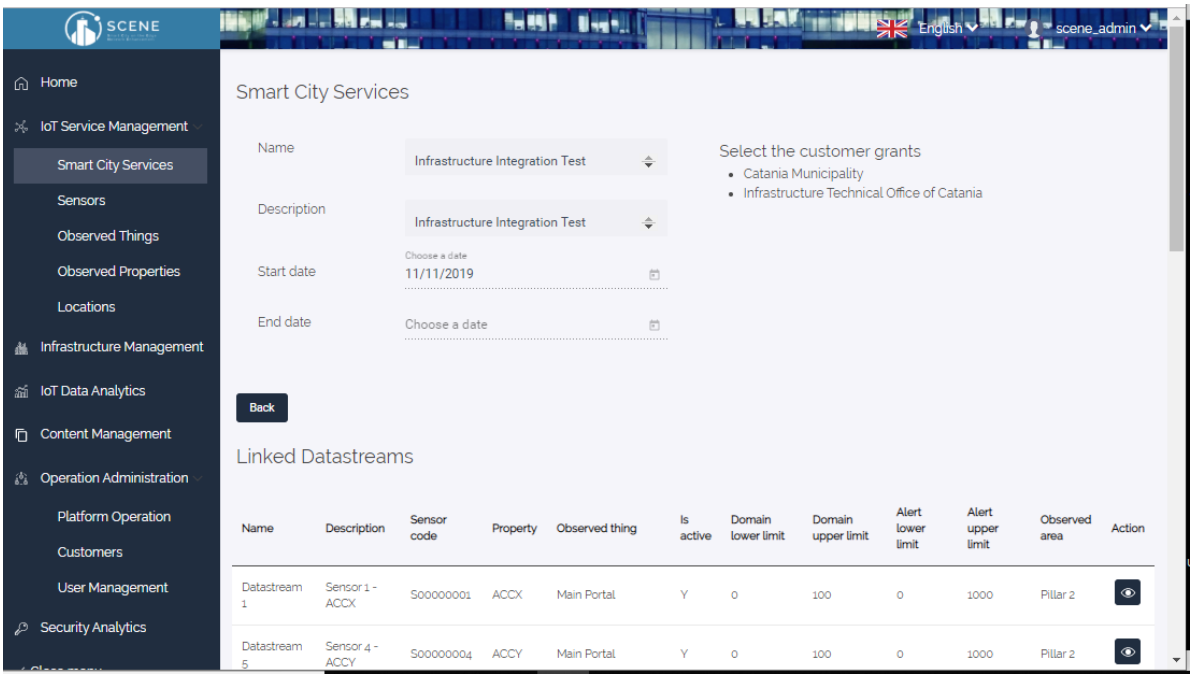

Figure 11 – IoT Service Management Dashboard snapshot

Visualization of Sensors and Observed Properties

This dashboard sections allow the visualization of the details associated to the Sensors and Observed-Things entities. These objects are managed (create, update and delete) from the Infrastructure Management system that takes care about the physical characteristics of the devices.

Management of Locations and Observed Things

These dashboard sections allow the complete management of the Locations (places where the Sensors are positioned) and the Observed Things (particulars of the observed area) entities. For the Location, the application keeps also the Latitude and Longitude coordinates and provides the map with the identified point.

Here below the screenshot for one Location object in Rennes:

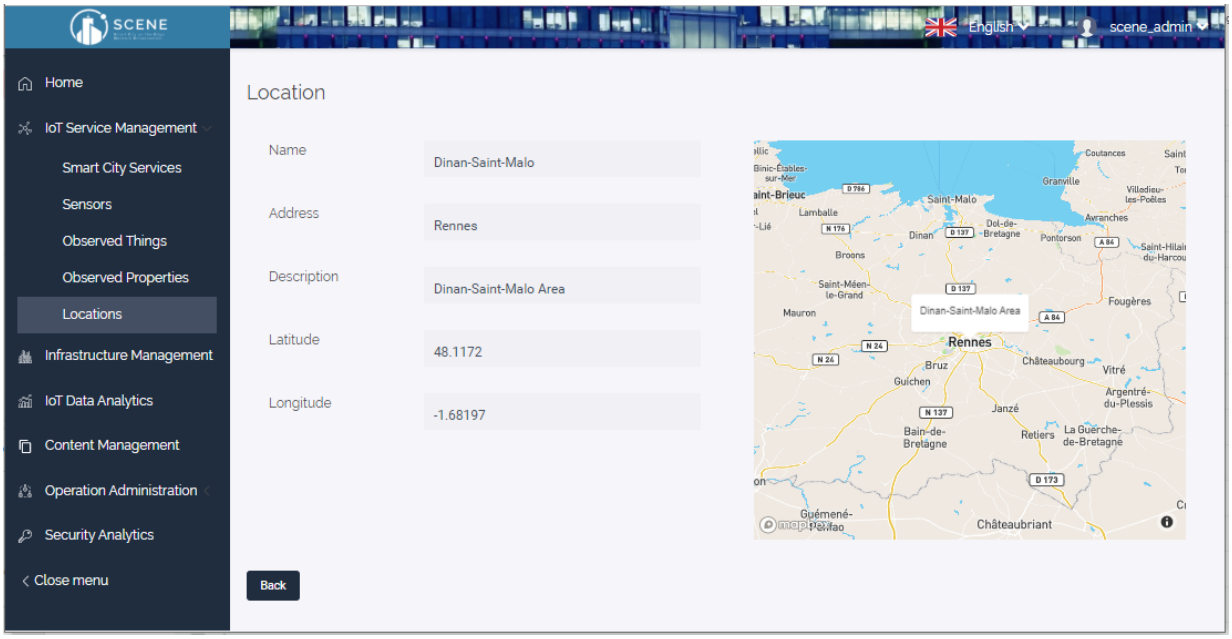

Figure 12 – A Location object snapshot

3.4.2 IoT Data Analytics dashboard

This module has been implemented using an Open Source product (Redash) specifically designed for the diagrams production of the IoT data with different representation methods (for example lines, pies, bars, …). In the Scene context, the Analytics Module reads the enriched data from the specific non-relational HDFS database and provides the users with analytical diagrams.

The Analytics functions are intended to provide some views about the platform behavior and consider the platform operational aspects like the number of the received messages, the latency time of measures, etc.

Also, Analytics functions implemented so far offer the basic standard view over the registered IoT data.

The queries, written using an SQL-like language, access the enriched data stored in the flat format and, in general, provide views about the healthy status of the platform.

In the first version of the platform, the following analytics have been implemented:

- Received Message Number: count of received messages grouped by Day and Smart City Service
- Measure Latency: latency between MeasureTime and MessageTime (acquisition on the platform)

- Measure Latency Average: table with average latency between MeasureTime and MessageTime grouped by Smart City Service
- IoTDataDrillDown: drill down on enriched data related to the measures arrived from the sensors

3.4.3 Infrastructure Management Dashboard

The Infrastructure management dashboard will provide the complete network topology layout of IGWs, and each IGW will provide the detail information for each connected IoT device. The IGWs communicate with Cache Controller (CC), and CC gets all required information from IGWs, as shown below. The network topology layout is an important part of the dashboard, and there is a possibility to build the infrastructure diagram builder to further enhance the visualization of infrastructure management by mapping it on the Maps. Some screen shots based on the initial work of the Infrastructure Management Dashboards are given below, which contains the configuration of IGWs, caching modules, and network topology of IGWs.

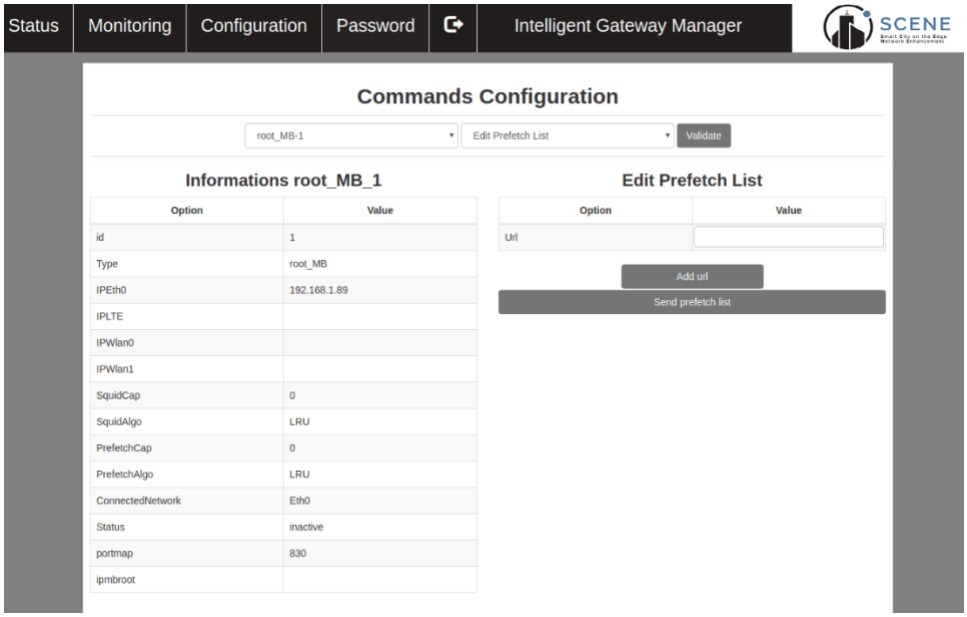

Figure 13 – Initial Dashboard for Command configuration in IGW

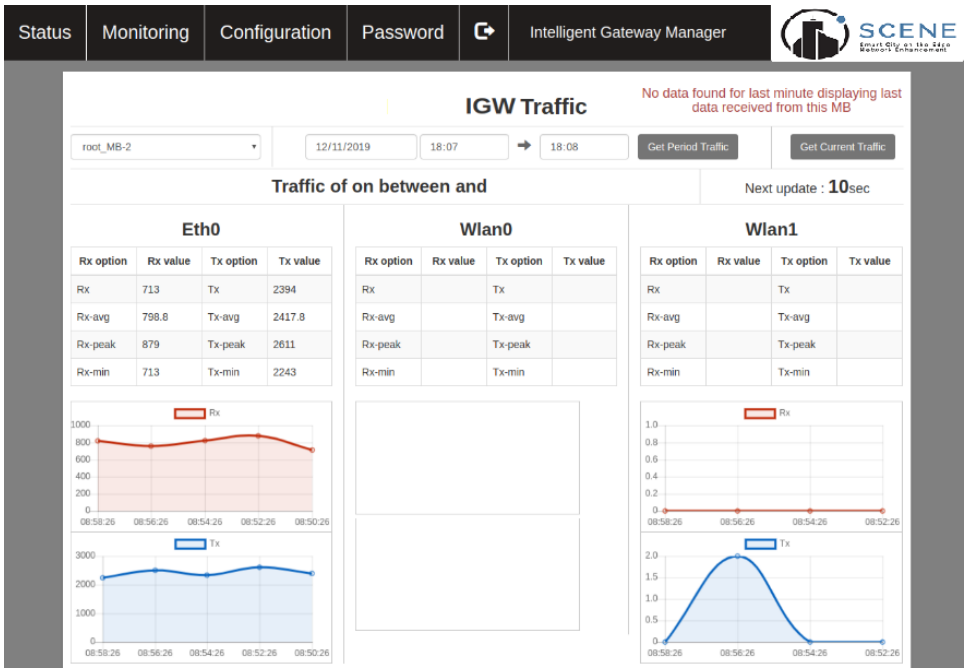

Figure 14 – Initial Dashboard IGW traffic monitoring

Device Registration

The device registration dashboard consists of two types of devices, i.e. sensors and Intelligent Gateway (IGW). This dashboard contains the detail information of each device. Initially, it contains the following information about the sensor devices.

- Define the sensor information, such Sensor ID, TypeID, Name, Description, Latitude, Longitude, etc.
- Store the sensor information in the database, and differentiate the sensors based on its type/properties.
- All sensors information, in JSON format, send to IGWs for collecting the information from the remote field sensor.

The snapshot of the initial dashboard for device registration is given below.

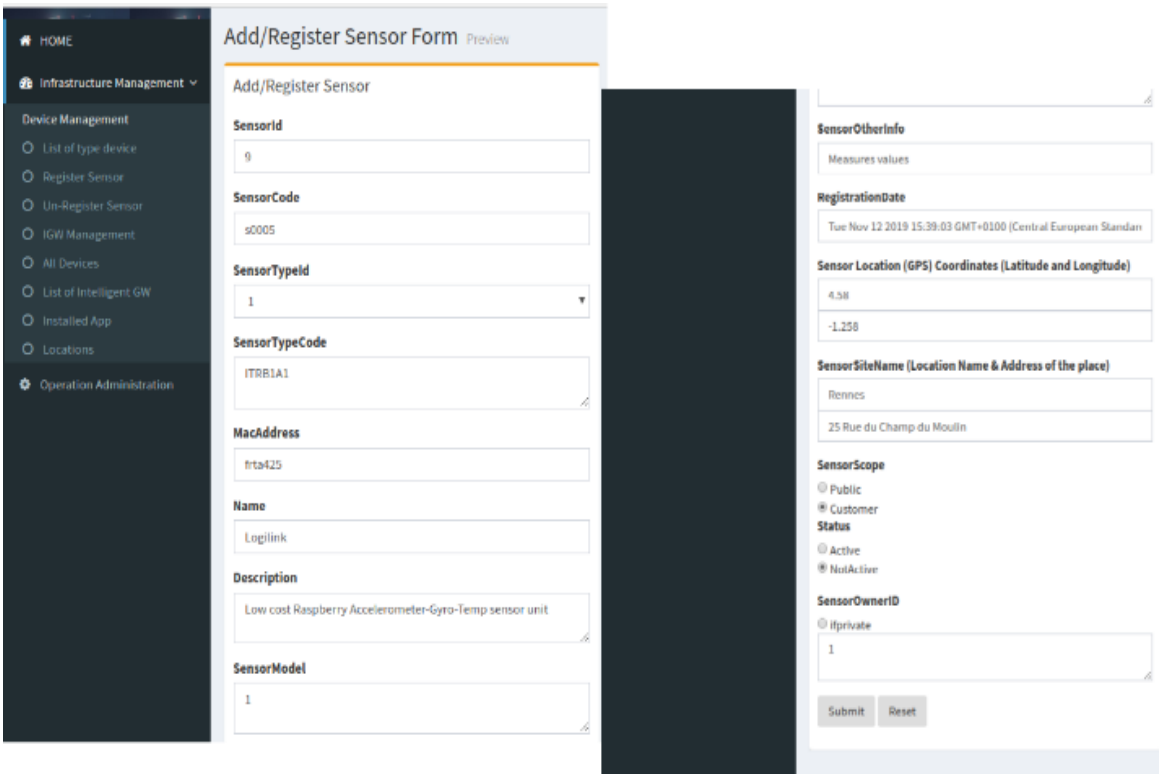

Figure 15 – Initial Dashboard for sensor device registration

3.4.4 Content Delivery dashboard

This component is not present in the first version of the platform.

3.4.5 Intrusion Detection System dashboard

This component is not present in the first version of the platform.

3.5 Content management

The content management entity consists of multiple components that are responsible for implementing the various functions related to the content delivery and management stuff. The given figure shows the communication among various components using the RESTful API. The control link is a communication between the components in the content management, while the traffic link is a communication for video traffic. The three components of Content Delivery Management module (CUM, CAS, CM) get the required

information from the IGW using different APIs, which are given below in the table. The Cache Controller (CC) communicates with content delivery component via internal API integration layer. The CC manages the infrastructure detail regarding deployed IGW, but it also gets up-to-date information about content caching in various IGW and executes the Prefetch command based on the content popularity or user movement in the destined IGW.

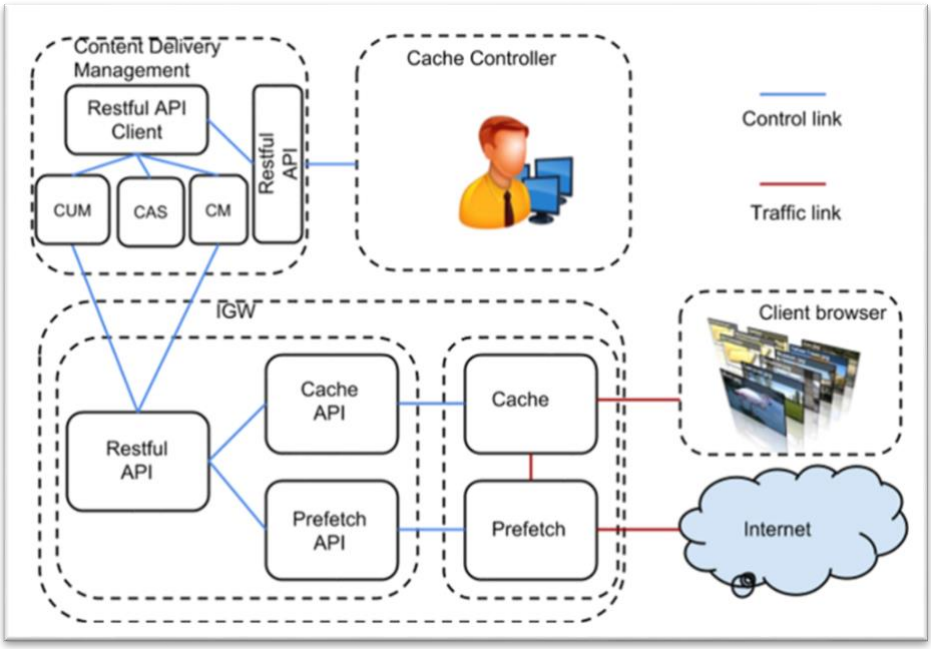

Figure 16 – Content Management Specification

This custom API Module manages the access to Content Management from the Infrastructure Management subsystem (i.e. CC) which extracts the required information from the IGW.

The following is an initial API list based on the developed content delivery management modules:

Method	Group	Resource	Description
GET	Request-list	/scene/gateway/api/v1.0/Request-list/connected_user/all	Get details of the connected user.
GET	Request-list	/scene/gateway/api/v1.0/Request-list/content_view/all	Get content viewed by the user

Method	Group	Resource	Description
GET	Prefetch-list	/scene/gateway/api/v1.0/Prefetch-list/explore	Get a prefetch list
POST	Prefetch-list	/scene/gateway/api/v1.0/Prefetch-list/add	Perform prefetching
GET	Caching	/scene/gateway/api/v1.0/Caching/get_content	Get cache content
GET	Caching	/scene/gateway/api/v1.0/Caching/configuration	Get cache module configuration
PUT	Caching	/scene/gateway/api/v1.0/Caching/configuration	Set caching module configuration
GET	Traffic	/scene/gateway/api/v1.0/Traffic/interfaces	Retrieve traffic on network interfaces

3.5.1 Content User Management (CUM)

It contains information about contents which are viewed by all users. The users view different video content, but some content is delivered from the local IGW cache to the users while remaining content is downloaded from the origin content server.

3.5.2 Content Access Statistics (CAS)

It provides statistics information about content in terms of access by all users, QoS parameters (bandwidth, throughput, etc.) and other important information related to content.

3.5.3 Content Management (CM)

This entity is responsible to take care about the content, whether they are available in the IGW cache, and requested from the origin content server. It manages the content which is needed to download on the web portal, content cached in the local IGW, and instructs prefetching command in order to download the content in the destined IGW based on the user movement.

The web portal provides opportunity for the customer to integrate the diverse sources of information in the unique desired way for the end users. A snapshot of the web portal is given below. The Content Delivery Management uses the RESTful based APIs developed for the content management in the web portal.

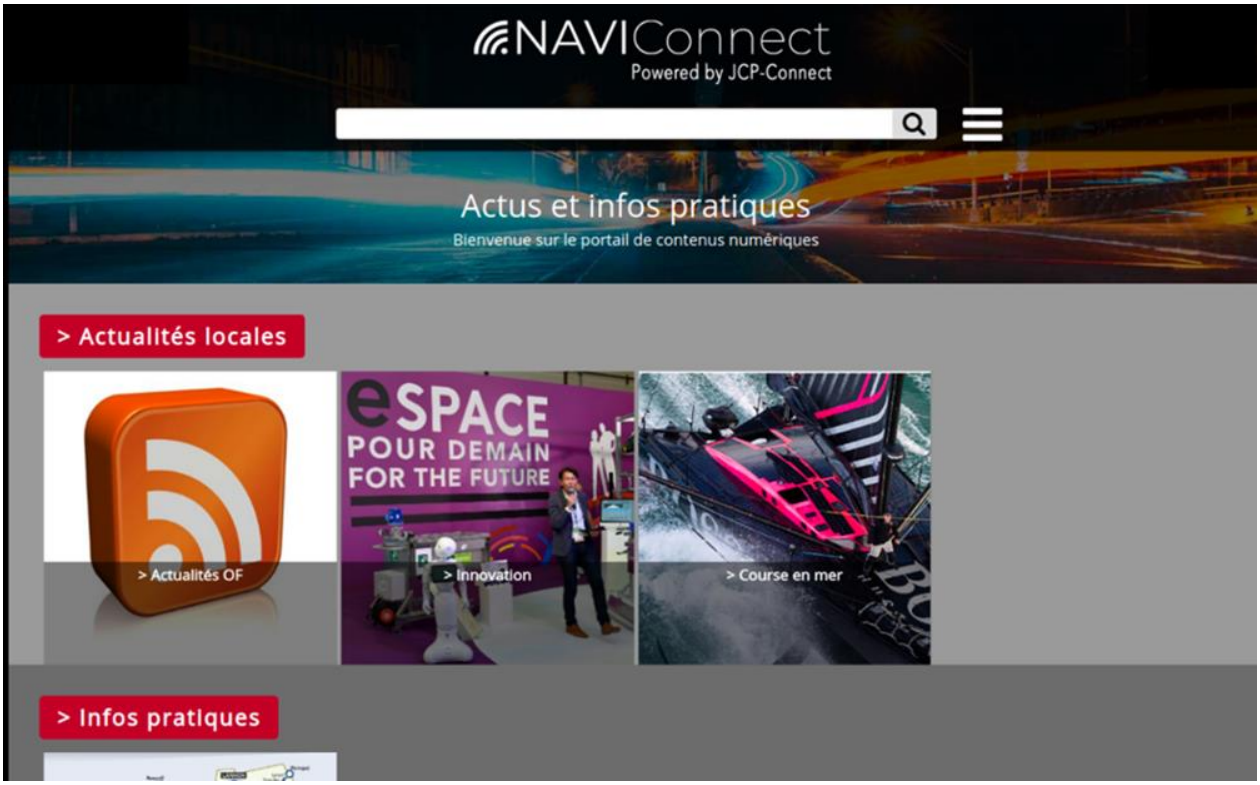

Figure 17 – Web Portal snapshot

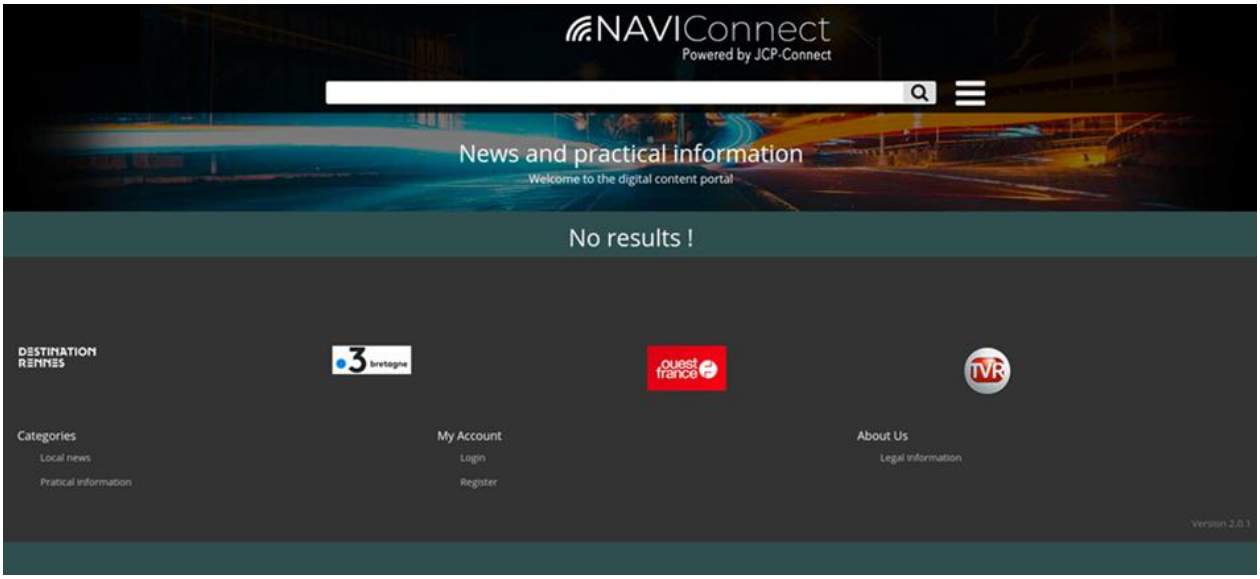

Figure 18 – Web Portal snapshot

The following is the list of Web Portal APIs

Method	Resource	Description
GET	/web-portal/api/1.0/content/1	Read data of the content page
DEL	/web-portal/api/1.0/content/1	Delete the content page
POST	/web-portal/api/1.0/content	Create a content page
PUT	/web-portal/api/1.0/content/1	Update the content page
GET	/web-portal/api/1.0/content	Get list of content pages
GET	/web-portal/api/1.0/rss_feed/1	Read data of the RSS Feed
DEL	/web-portal/api/1.0/rss_feed/1	Delete data of the RSS Feed
Post	/web-portal/api/1.0/rss_feed/1	Create a RSS Feed Url
PUT	/web-portal/api/1.0/rss_feed/1	Update the RSS Feed Url
GET	/web-portal/api/1.0/category/1	Get data of the item
DEL	/web-portal/api/1.0/category/1	Delete the category
POST	/web-portal/api/1.0/category	Create a category
PUT	/web-portal/api/1.0/category/1	Update the category
GET	/web-portal/api/1.0/category	Get list of the categories
POST	/web-portal/api/1.0/upload	Upload File
GET	/web-portal/api/1.0/language	List of Language
GET	/web-portal/api/1.0/sub_category/1	Sub-catogory, Get data of the item
DEL	/web-portal/api/1.0/sub_category/1	Delete the category
POST	/web-portal/api/1.0/sub_category	Create a category
PUT	/web-portal/api/1.0/sub_category/1	Update the category