

**INTERNATIONAL CENTER OF NUMERICAL METHODS IN
ENGINEERING (CIMNE)**

OSI4IOT Platform Tutorial



TABLE OF CONTENTS

1	OSI4IOT Platform	4
1.1	OSI4IOT platform services	4
1.2	Data flow in OSI4IOT platform	6
1.3	OSI4IOT platform installation	14
1.3.1	Requirements	14
1.3.1.1	<i>Docker installation</i>	14
1.3.1.2	<i>Telegram Bot token, chat id and group invitation link</i>	14
1.3.1.3	<i>Domain name</i>	18
1.3.1.4	<i>Email address</i>	19
1.3.2	Command Line Interface (CLI) installer	19
1.3.3	Running OSI4IOT CLI	21
1.3.3.1	<i>Local deployment</i>	22
1.4	Documentation	25
1.4.1	Home Screen	25
1.4.2	Platform Assistant	27
1.4.3	Dashboards	28
1.4.4	Digital Twin Simulator	28
1.4.5	Mobile Sensors (Only Android devices)	29
1.4.6	OSI4IOT Platform Roles	29
1.4.7	Super admin role	30
1.4.7.1	<i>Buildings</i>	31
1.4.7.2	<i>Floors</i>	34
1.4.7.3	<i>Organizations</i>	36
1.4.7.4	<i>Global Users</i>	37
1.4.7.5	<i>Refresh Tokens</i>	40
1.4.7.6	<i>Tools</i>	41
1.4.8	Organization admin role	42
1.4.8.1	<i>Organizations Managed</i>	42
1.4.8.2	<i>Organizations Users</i>	43
1.4.9	Group admin role	48
1.4.9.1	<i>Groups managed</i>	48
1.4.9.2	<i>Group members</i>	50
1.4.9.3	<i>Devices</i>	52
1.4.9.4	<i>Topics</i>	54
1.4.9.5	<i>Dashboards</i>	58
1.4.9.6	<i>Digital Twins</i>	59
1.4.10	User role	62
1.4.10.1	<i>User Profile</i>	62
1.4.10.2	<i>Memberships in Orgs</i>	62
1.4.10.3	<i>Memberships in Groups</i>	64
2	References	65

LIST OF FIGURES

Figure 1: Open source packages used in OSI4IOT platform.....	6
Figure 2: Data flow in OSI4IOT platform.....	7
Figure 3: Conversion from TensorFlow to TensorFlow.js.....	8
Figure 4: Grafana dashboards.....	9
Figure 5: Telegram messages notification channel.....	9
Figure 6: Email message notification channel.....	10
Figure 7: Digital twin models logic in a Node-Red instance.....	11
Figure 8: Start bottom of Telegram BotFather application.....	15
Figure 9: Add members to the telegram group.....	17
Figure 10: Fill the group name.....	17
Figure 11: Telegram Group info.....	18
Figure 12: Location of the CLI installer.The folders for installation found in the subfolder indicate the type of operative system and architecture. Therefore:	19
Figure 13: Summary of the CLI installers based on OS and architecture	20
Figure 14: Create the project	20
Figure 15: Download the installer using curl in Linux terminal.....	20
Figure 16: Execute the installer	21
Figure 17: The CLI starting screen.	21
Figure 18: Local deployment	22
Figure 19: Local deployment configuration.	24
Figure 20: Local deployment with no cert configuration. Output after correctly installed.	25
Figure 21: Home screen.....	26
Figure 22: Login page.....	26
Figure 23: Geolocation screen.....	27
Figure 24: Grafana login page.....	28
Figure 25: Digital twin simulator page.....	28
Figure 26: Role navigation bar.....	29
Figure 27: Super Admin role default page.	30
Figure 28: Creating a building.....	31
Figure 29: GeoJson.io home screen.....	32
Figure 30: Creation of a polygon representing the outline of the building.....	32
Figure 31: Building edition.	33
Figure 32: Parameters of Buildings table.....	34
Figure 33: Floors table.....	34
Figure 34: Creation of a floor in geojon.io application.	35
Figure 35: Create floor with GeoJSON data.	35
Figure 36: Parameters of Floors table.	36
Figure 37: Organizations table.....	36
Figure 38: Parameters of Organizations table.	37
Figure 39: Global user table.	38
Figure 40: Creation of a new global user.	38
Figure 41: Edit global user.....	39
Figure 42: Parameters of Global users table.	39
Figure 43: Refresh token table.	40
Figure 44: Parameters of Refresh tokens table.....	40
Figure 45: Tools page.....	41

Figure 46: Default page of the Org admin role.....	42
Figure 47: Parameters of Orgs managed table.....	43
Figure 48: Organizations user table.....	43
Figure 49: Parameters of Org users table.....	44
Figure 50: Groups table.....	44
Figure 51: Creation of a Group.....	45
Figure 52: Parameters of Groups table.....	46
Figure 53: Table of Nodered instances in orgs.	47
Figure 54: Table of Nodered instances in orgs.	47
Figure 55: Group admin role tabs.	48
Figure 56: Parameters of Groups managed table.	49
Figure 57: Edition of a Group managed.	50
Figure 58: Group members table.	50
Figure 59: Edit role for group member.	51
Figure 60: Parameter of Group members table.	51
Figure 61: Devices table.....	52
Figure 62: Creation of a device.....	52
Figure 63: Defining the device location.	53
Figure 64: Parameters of devices table.	54
Figure 65: Topics table.	54
Figure 66: Creation of a new topic.	55
Figure 67: Edition of topic.	56
Figure 68: Topics table parameters.	57
Figure 69: Measurements table.	57
Figure 70: Dashboards table.	58
Figure 71: Parameters of Dashboards table.	59
Figure 72: Digital twins table.	59
Figure 73: Creation of a new digital twin with Grafana dashboard type.	60
Figure 74: Creation of a new digital twin with Gltf 3D model type.	60
Figure 75: Parameters of Digital twins table.	61
Figure 76: Default page of User role.	62
Figure 77: Memberships in orgs table.	63
Figure 78: Memberships in orgs table.	63
Figure 79: Memberships in groups table.	64

1 OSI4IOT Platform

The OSI4IOT platform has been developed to give an adequate answer to all the requirements of the Shipyard 4.0 strategy conceived in the Fibre4Yards project. The OSI4IOT name stand for "Open Source Integration For Internet of Thing". The platform OSI4IOT is based on open software and itself has been conceived as open software. OSI4IOT continues in development and can be download from the web address: <https://github.com/osi4iot/osi4iot>

In the next sections are explained the different elements currently developed in the OSI4IOT platform.

1.1 OSI4IOT platform services

The OSI4IOT seamlessly integrates various open source software packages with custom code tools.

The open source package used in the platform are the following (Figure 1):

- **Eclipse Mosquitto:** Is an open source (EPL/EDL licensed) message broker that implements the MQTT protocol versions 5.0, 3.1.1 and 3.1. Mosquitto is suitable for use on all devices from low power single board computers to full servers [1].
The MQTT protocol provides a lightweight method of carrying out messaging using a publish/subscribe model. This makes it suitable for Internet of Things messaging such as with low power sensors or mobile devices such as phones, embedded computers or microcontrollers.
Authentication and authorization mechanisms are handled by a plugin developed in Go language called mosquitto-go-auth [2].
- **Node-RED:** Is a visual programming tool for wiring together hardware devices, APIs and online services. It provides a web browser-based flow editor, which can be used to create JavaScript functions. Elements of applications can be saved or shared for re-use. The runtime is built on Node.js. The flows created in Node-RED are stored using JSON [3].
- **PostgreSQL:** Is an advanced, enterprise-class, and open-source relational database system. PostgreSQL supports both SQL (relational) and JSON (non-relational) querying [4].
- **Timescaledb:** Is an extension on PostgreSQL. It gives you all the power of PostgreSQL, plus new facilities for working with time series data and complex SQL queries [5].
- **Pgadmin:** Is a web-based Graphical User Interface (GUI) management application used to communicate with PostgreSQL and derivative relational databases on both local and remote servers [6].

- **Grafana:** Is an open source interactive data-visualization platform, developed by Grafana Labs, which allows users to see their data via charts and graphs that are unified into one dashboard for easier interpretation and understanding. Grafana allows you to consult and configure alerts on data and metrics from anywhere that information is stored [7].
- **Grafana Image Renderer:** Is a Grafana backend plugin that handles rendering panels and dashboards to PNGs using a headless browser (Chromium) [8].
- **Traefik:** Is a modern HTTP reverse proxy and load balancer that makes deploying microservices easy. Traefik integrates with your existing infrastructure components (Docker, Swarm mode, Kubernetes, Marathon, Consul, Etcd, Rancher, Amazon ECS, etc.) and configures itself automatically and dynamically. Pointing Traefik to the orchestrator used should be the only configuration step needed [9].
- **Minio:** Is a High Performance Object Storage released under GNU Affero General Public License v3.0. It is API compatible with Amazon S3 cloud storage service. MinIO can be used to build high performance infrastructure for machine learning, analytics and application data workloads [10].
- **Portainer:** Is a lightweight service delivery platform for containerized applications that can be used to manage Docker, Swarm, Kubernetes and ACI environments. It is designed to be as simple to deploy as it is to use. The application allows you to manage all your orchestrator resources (containers, images, volumes, networks and more) through a ‘smart’ GUI and/or an extensive API [11].
- **Keepalived:** Is a routing software written in C. The main goal of this project is to provide simple and robust facilities for loadbalancing and high-availability to Linux system and Linux based infrastructures. Keepalived implements a set of checkers to dynamically and adaptively maintain and manage loadbalanced server pool according to their health [12].

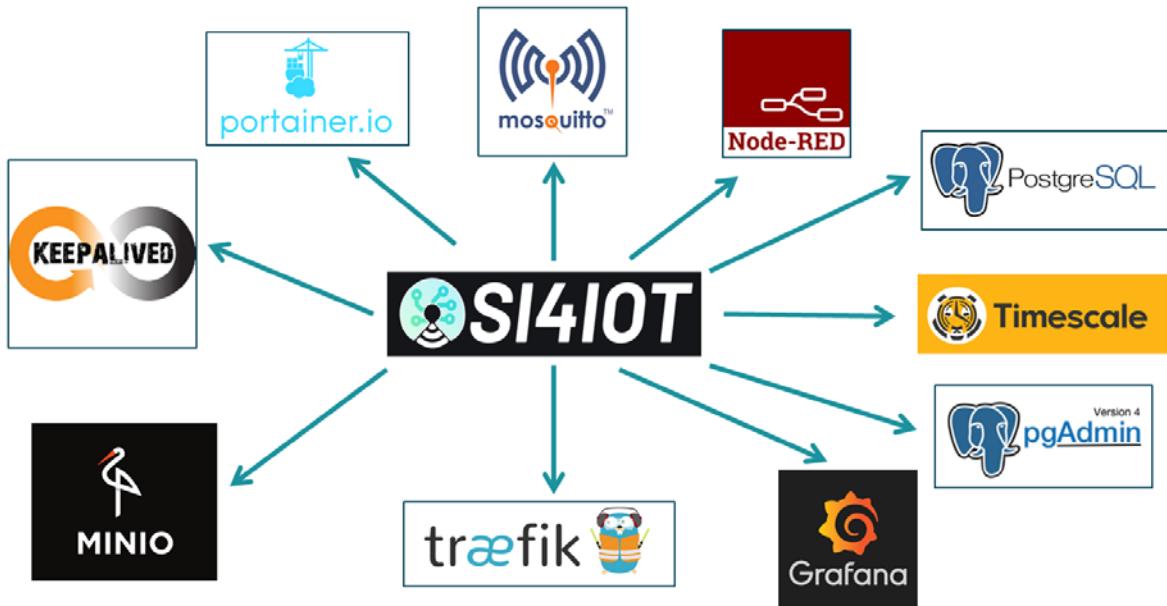


Figure 1: Open source packages used in OSI4IOT platform.

The custom code tools developed in the OSI4IOT platform are the following:

- **Admin_api**: Is a microservice written in Node.js that serves as the back-end of the platform. It is used as an intermediary between the frontend and the database.
- **Frontend**: Is the web user interface of the platform. It is written in React.js
- **Dev2pdb**: Is a microservice written in Go language. It receive the data sent by the sensors using the MQTT protocol and transform it to be stored in the database.

In the OSI4IOT platform, all the services previously commented are running inside Docker containers [13]. Docker is an open software platform that allows you to build, test, and deploy applications quickly. Docker packages software into standardized units called containers that have everything the software needs to run including libraries, system tools, code, and runtime. Using Docker, you can quickly deploy and scale applications into any environment and know your code will run.

In a production environment Docker containers are used together a container orchestrator. Container orchestration automates the provisioning, deployment, networking, scaling, availability, and lifecycle management of containers. Today, Kubernetes is the most popular container orchestration platform but it has a complex installation process and a steep learning curve. Other container orchestration tools include Docker Swarm and Apache Mesos.

Docker Swarm will be used on the OSI4IOT platform because it is straightforward to install and easier to use.

1.2 Data flow in OSI4IOT platform

The data flow in the OSI4IOT platform can be divided into five layers as shown the Figure 2:

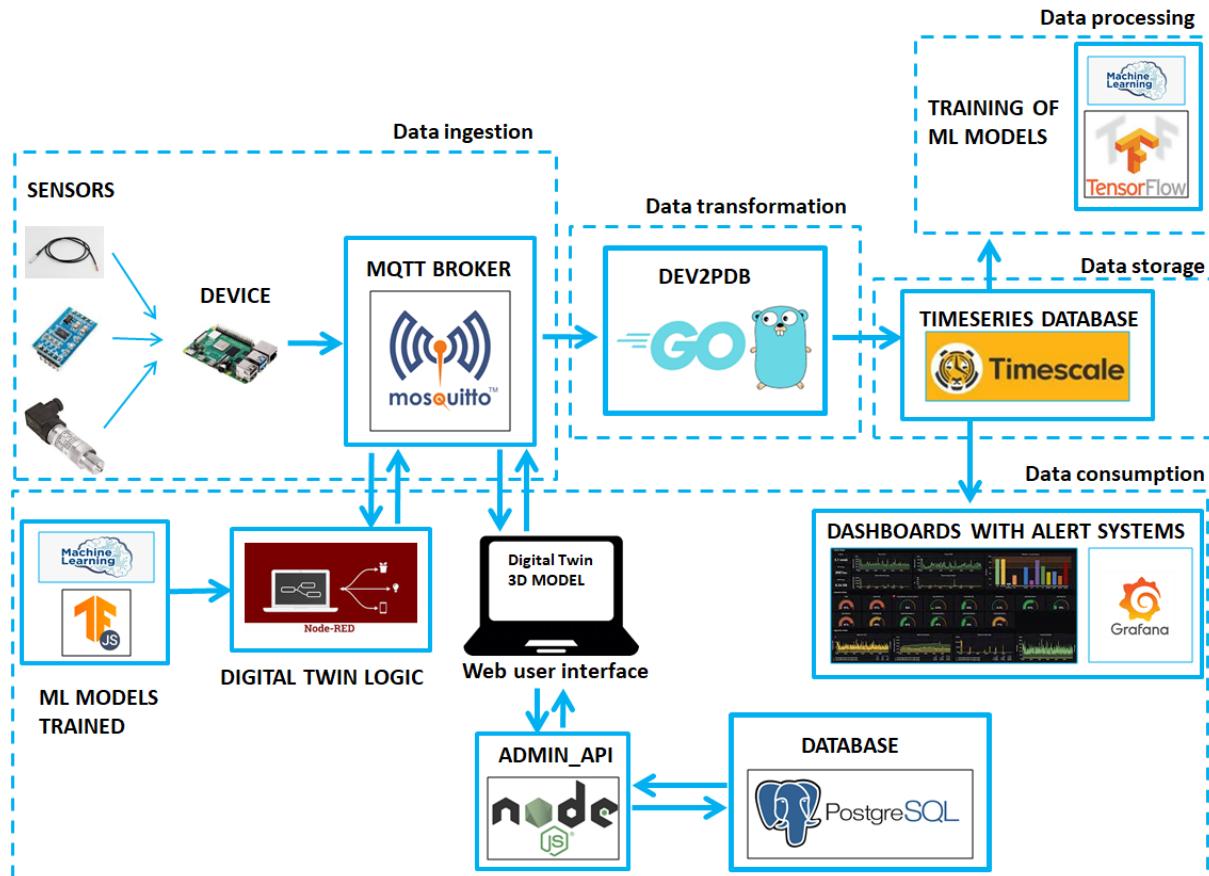


Figure 2: Data flow in OSI4IOT platform.

- **Data ingestion:** In this layer, the sensor data from assets is collected in a device connected to internet. Then data are sent to the platform using the MQTT protocol. The way in which the sensor data is collected by the devices depends on the control system used in each factory.
- **Data transformation:** Using the dev2pdb microservice the data is conveniently transformed to be stored in the database.
- **Data storage:** The data is stored in the Timescale database. This time-series database allow setting automated data retention policies that drop old data according to a schedule and defined rules.
- **Data processing:** The data stored in the Timescale database can obtained by mean of http requests to the admin_api service. Once the data has been obtained, it can be used to train machine-learning models of the assets. TensorFlow is used to train the machine learning models. TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. TensorFlow provides a stable Python API. TensorFlow also has a library for machine learning in JavaScript called TensorFlow.js. Using the provided JavaScript APIs, TensorFlow.js

allows users to use either Tensorflow.js models or converted models from TensorFlow (see Figure 3). In this way, the models converted to TensorFlow.js can be used in Node-RED.

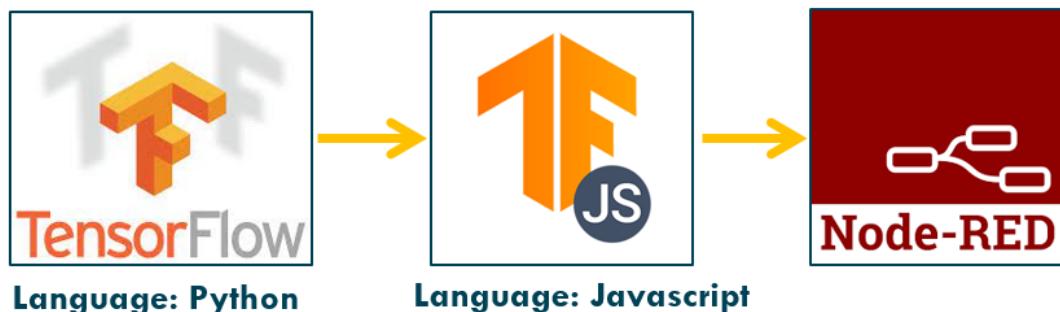


Figure 3: Conversion from TensorFlow to TensorFlow.js

- **Data consumption:** This layer has several elements that deserve to be explained in detail:
 - 1-) Grafana dashboards with alert system
 - 2-) Digital twin models logic
 - 3-) Web user interface (WUI)

Grafana dashboards with alert system

Once the data is stored in the Timescale database it can be visualized in almost real-time using the dashboards provided by Grafana service.

Grafana also has an alert system that launches a notification message to some notification channel when the value of any parameter is outside an admissible threshold. The notification channels used in the OSI4IOT platform are email and/or Telegram messages.

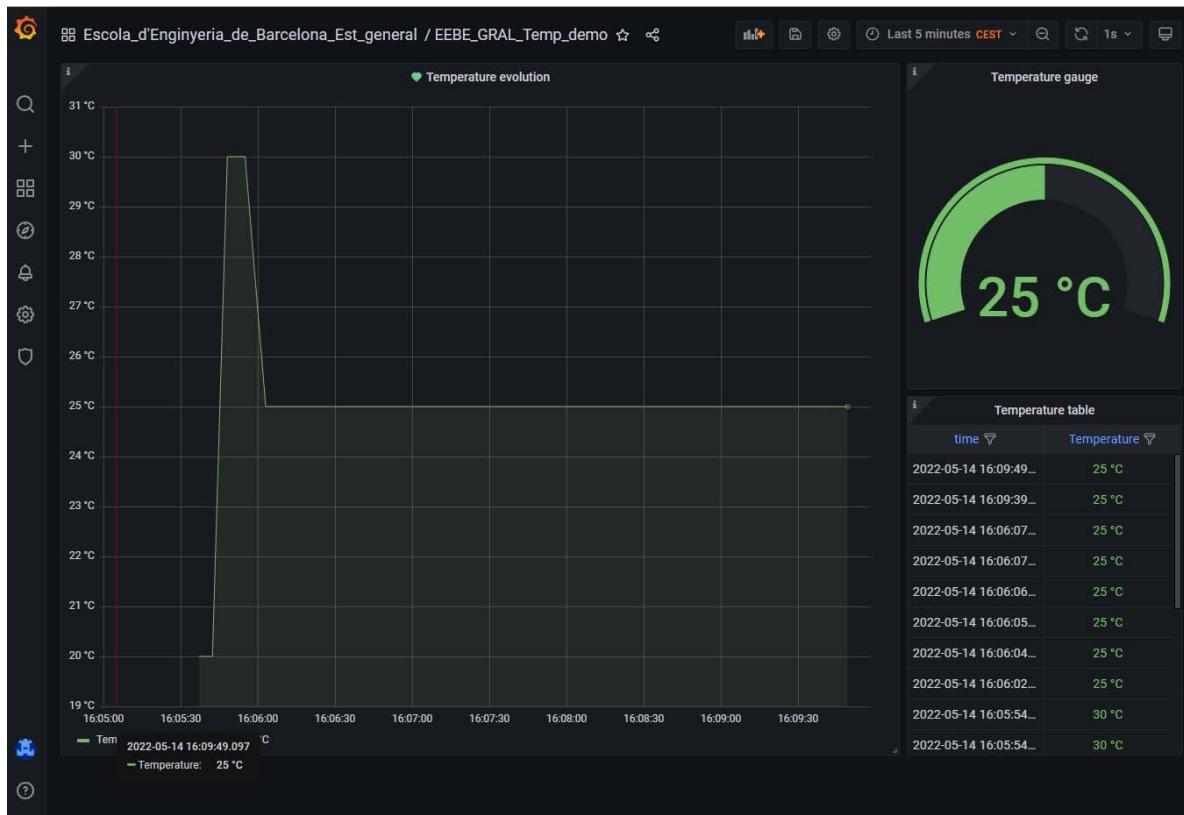


Figure 4: Grafana dashboards.

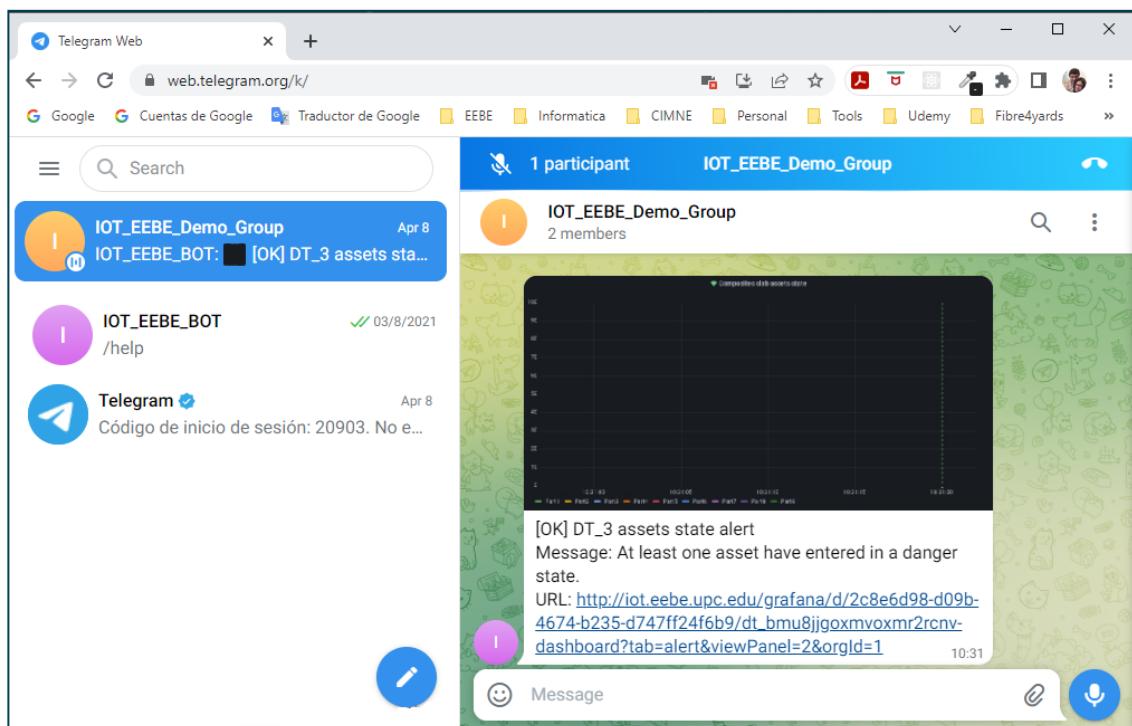


Figure 5: Telegram messages notification channel.

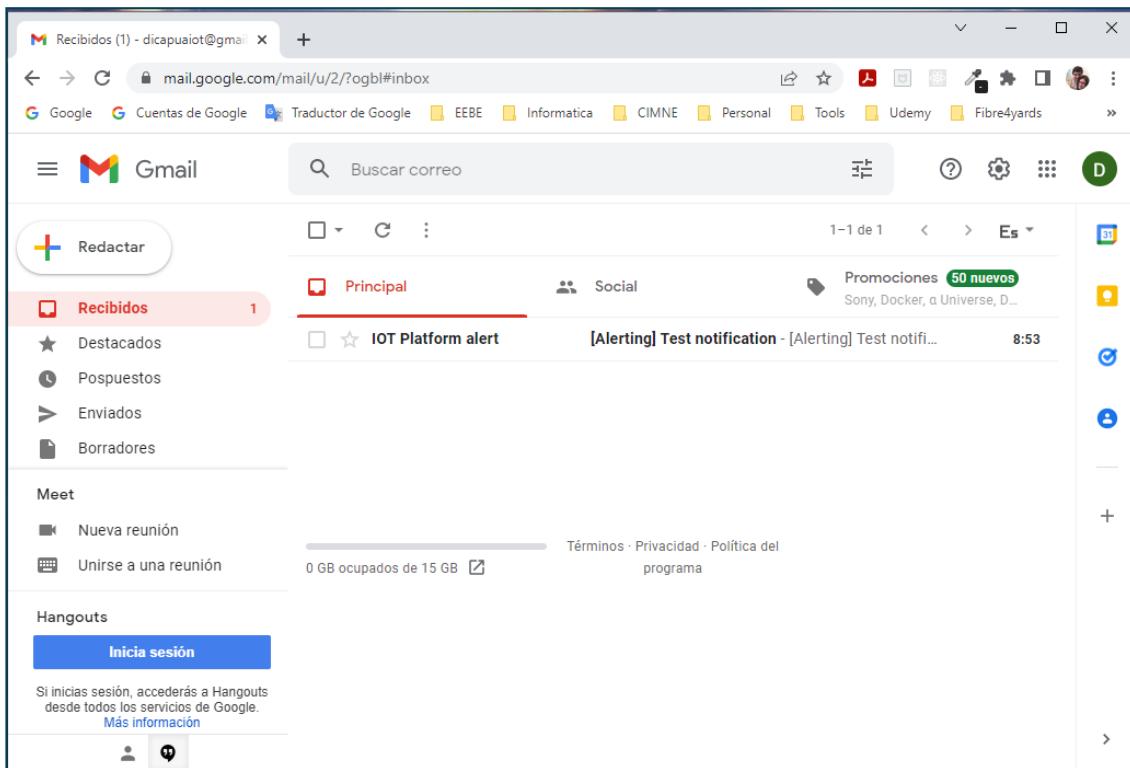


Figure 6: Email message notification channel.

Digital twin models logic

The logic necessary for the operation of the digital twin models of the assets is implemented in Node-RED instances. Node-RED lets you easily build applications by joining black box functions called “nodes” using a web interface.

Node-RED provides access support to the MQTT protocol. It provides an MQTT subscription node and a publication node. The subscription node is used for data input, while the publication node can be used for data output.

A subscription node can be configured to listen to a topic related to sensor data of a specific asset. Once the data of this topic arrive to the MQTT broker, this subscription node listens for it and then sends the data to a function node where the digital twin model (DTM) logic is defined.

Digital twin models logic can be based in basic rule engine or in already trained machine learning models that are previously converted to TensorFlow.js. The output of this function node is information related with the asset state.

A message with this information of the asset state is send by a MQTT publication node. This message can be listen by the WUI and react accordingly.

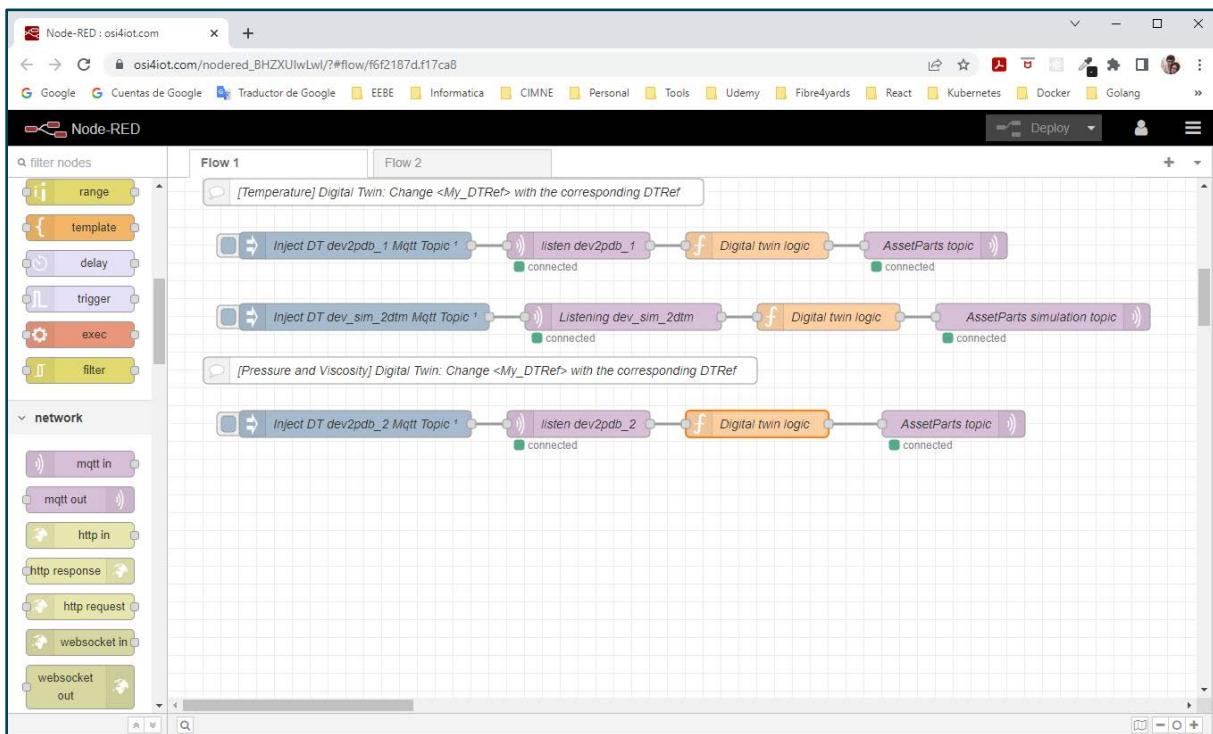


Figure 7: Digital twin models logic in a Node-Red instance.

Web user interface

The WUI of the OSI4IOT platform can be used for monitoring the assets state in the shipyard. This tool allows the geolocation of the organizations, groups and devices that make up the shipyard. Also allows you to geo-locate assets with a bad operating status.



Figure 8: Geolocation of assets with a bad operating status.

It also has a module for the 3D visualization of the digital twin models. This tool allow monitoring the asset state in almost real time. Dynamics parts of the 3D model are updated

in function of the asset state. When a part of an asset has problems, the tool makes it easy to locate this area by means of a blinking red light.

The 3D models are first created in Blender [14] and then exported in gltf format that is supported by the platform. Gltf files are stored in an S3 bucket.

In the Figure 9 can be seen a digital twin model of the pultrusion manufacture process developed by the IRURENA partner [19].

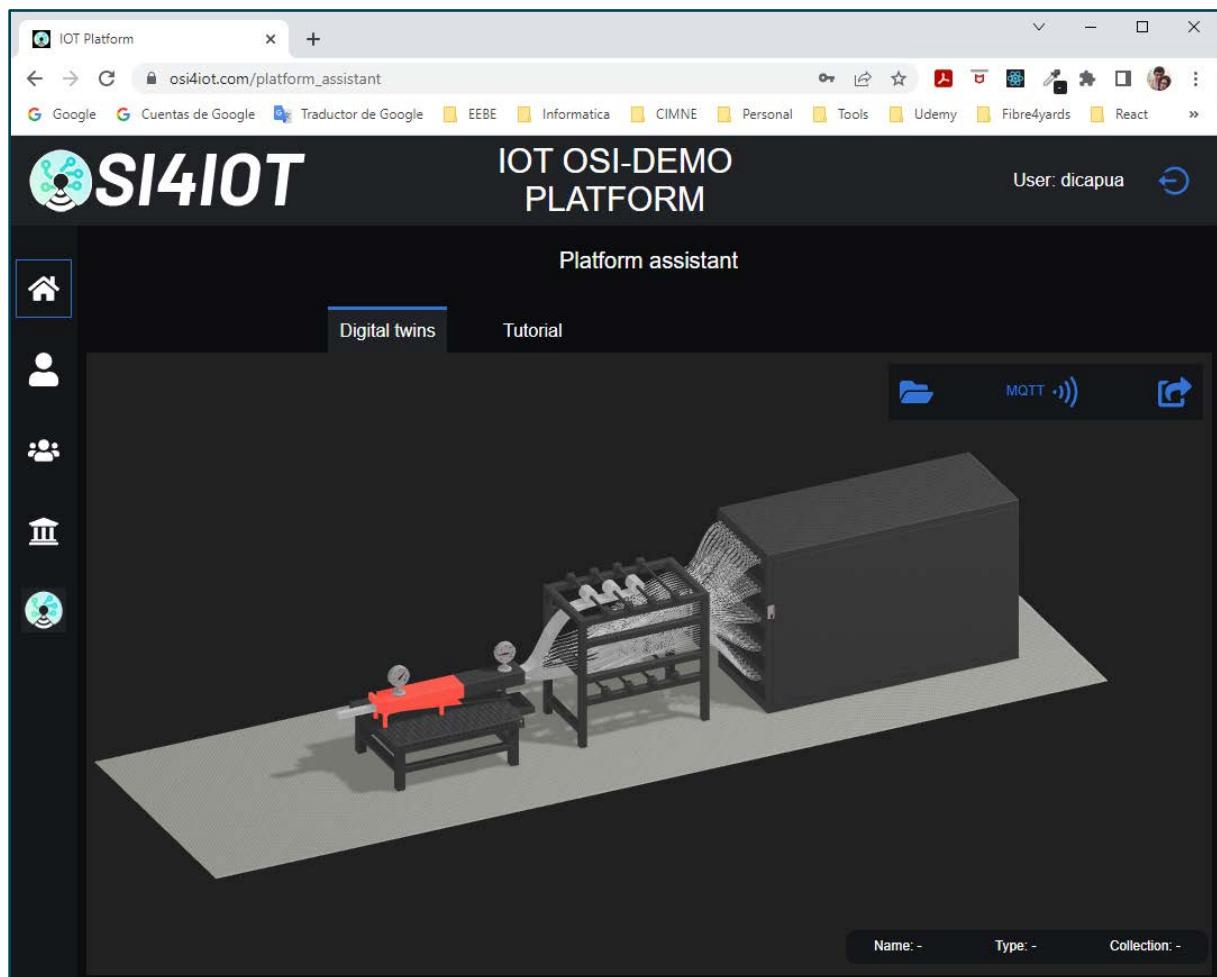


Figure 9: 3D visualization of the DTM of pultrusion process of IRURENA partner.

This 3D visualization module can be integrated with the results obtained by physical-driven numerical models as the Finite Element Method (FEM). These results are updated in almost real-time based on the asset state.

The FEM result files are stored in a S3 bucket. In this way, each digital twin model can have several FEM results files.

In Figure 10 is shown a digital twin model of a gas tank integrated with the result of Von Mises stresses obtained by a FEM simulation.

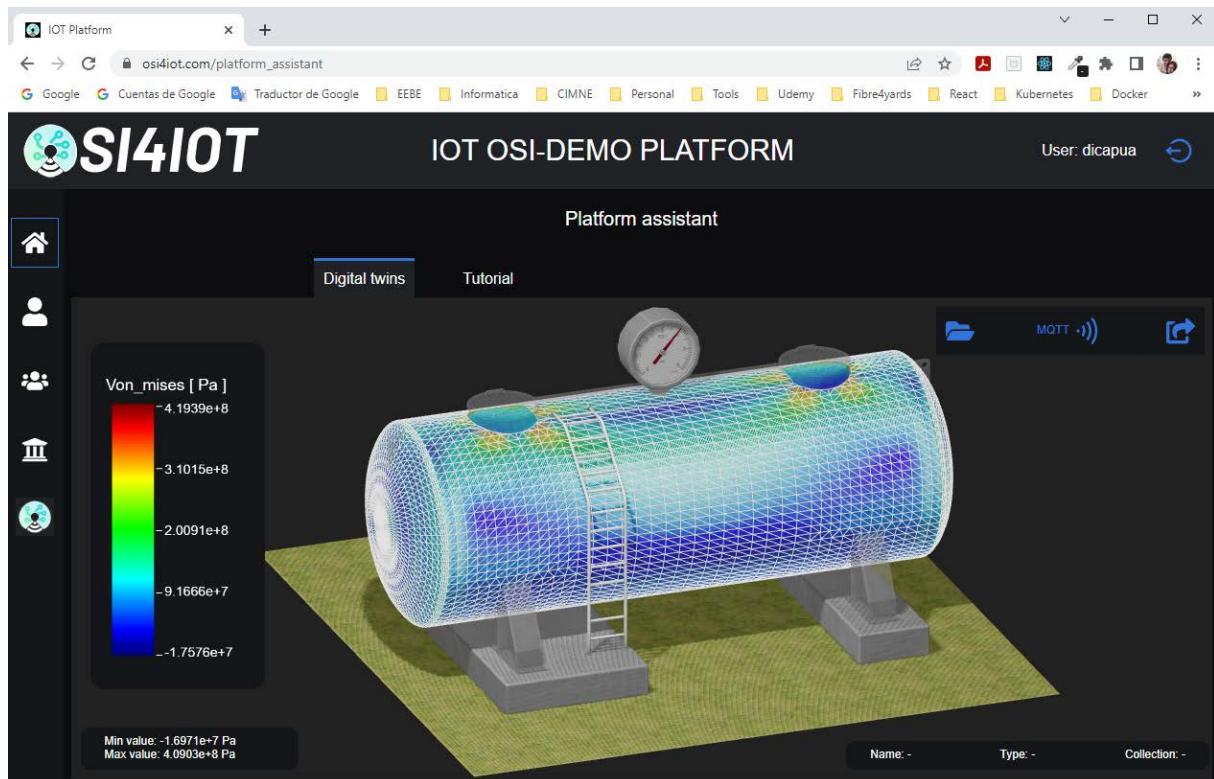


Figure 10: 3D visualization of the DTM of a gas tank with FEM results integrated.

1.3 OSI4IOT platform installation

In order to install and have the **OSI4IOT** platform running correctly, a few requirements need to be met:

- Docker installed in every node for the platform
- A Telegram bot token.
- A Telegram chat id for main organization default group.
- A Telegram invitation link for main organization default group.
- A domain name to access the platform through internet.
- An e-mail address used to send the notifications from the platform. The password of this e-mail is also required.

1.3.1 Requirements

1.3.1.1 Docker installation

The Docker installation depends of the operating system of the node. For Windows or Mac the best option is to install Docker Desktop application (<https://docs.docker.com/desktop/>). For Linux servers the preferred option is to install Docker Engine (<https://docs.docker.com/engine/>).

1.3.1.2 Telegram Bot token, chat id and group invitation link

To create a Telegram bot must be used the BotFather application. Enter in your Telegram account and find the user “BotFather”:

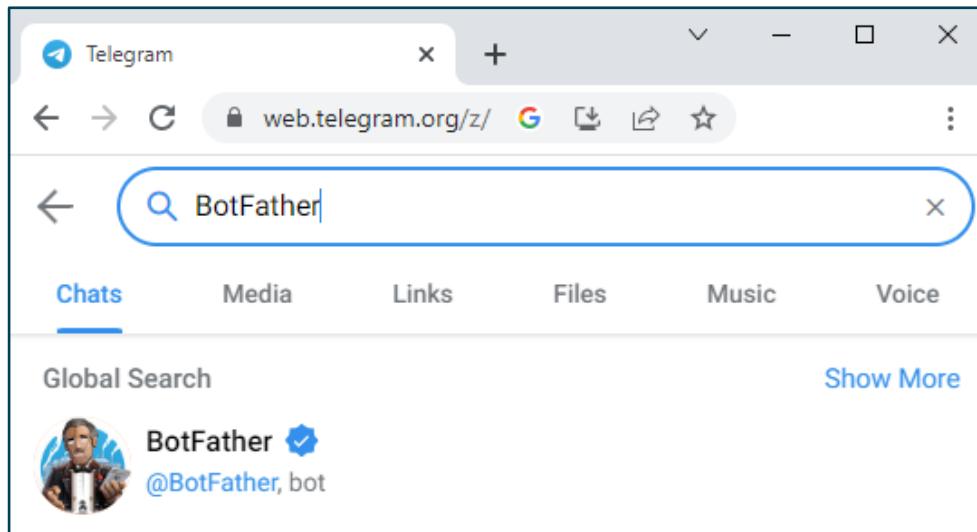


Figure 11 Select BotFather user in Telegram.

The press START bottom to start BotFather application.

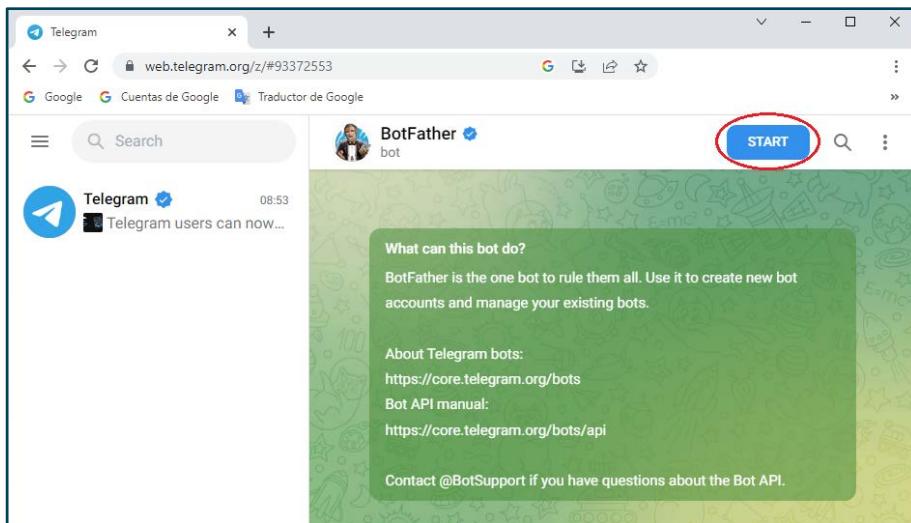


Figure 8: Start bottom of Telegram BotFather application.

Once the START bottom is pressed, it can be seen the window shown in Figure 12. To initiate a new bot type the command "/newbot".

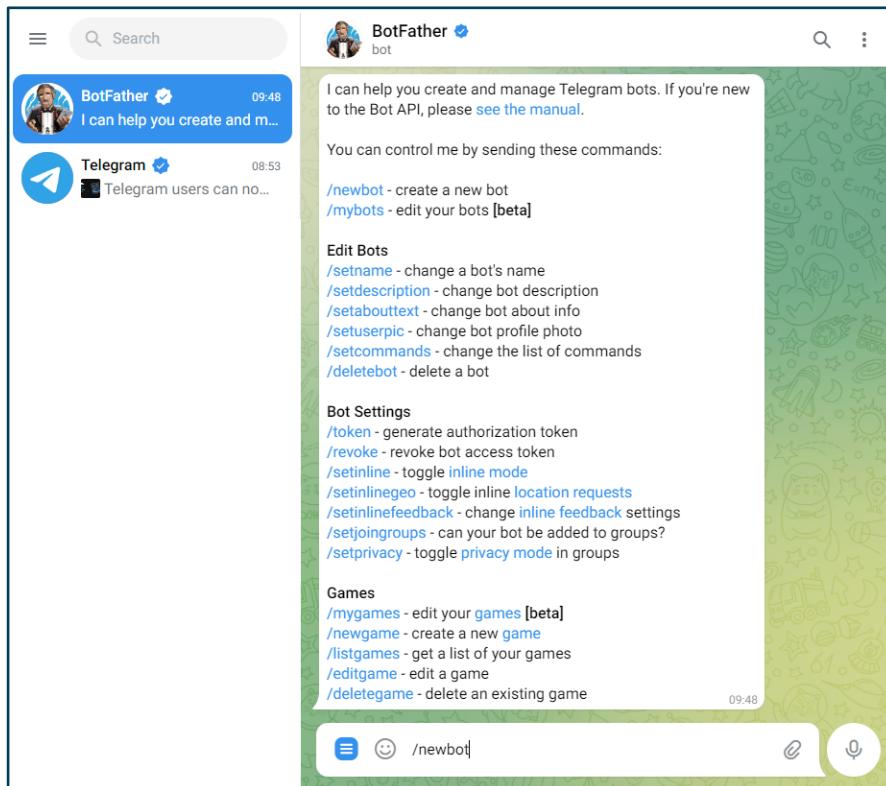


Figure 12: Telegram BotFather application option.

Then complete the prompts required by the BotFather form.

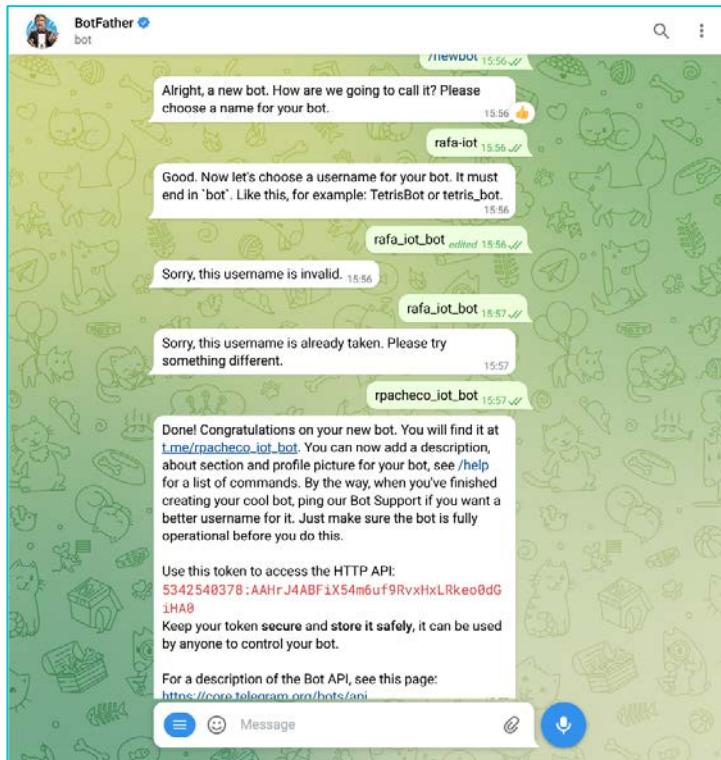


Figure 13: Filling the requirements of BotFather.

If everything has gone correctly, the bot token will be generated.

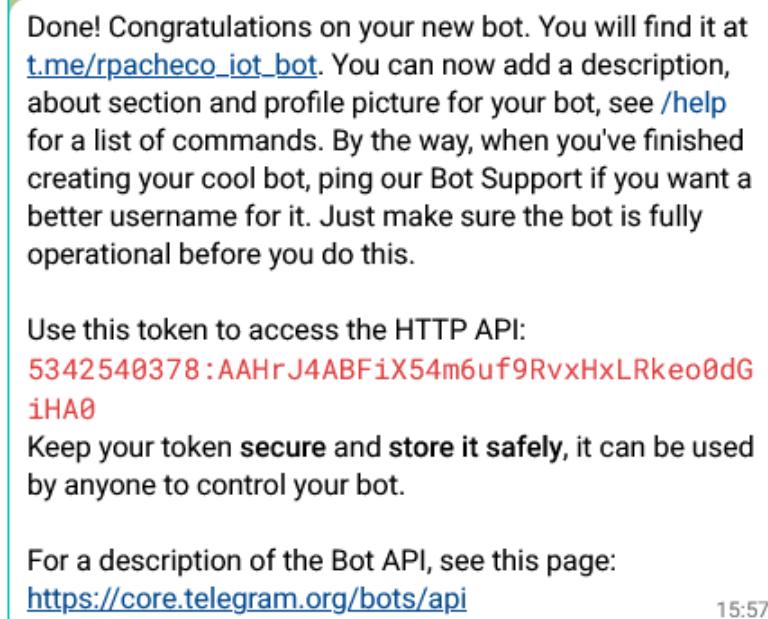


Figure 14: Bottoken created successfully using BotFather.

Once the bot token and the bot has been created. Create a Group and add the bot you have created. Select **New Group**.

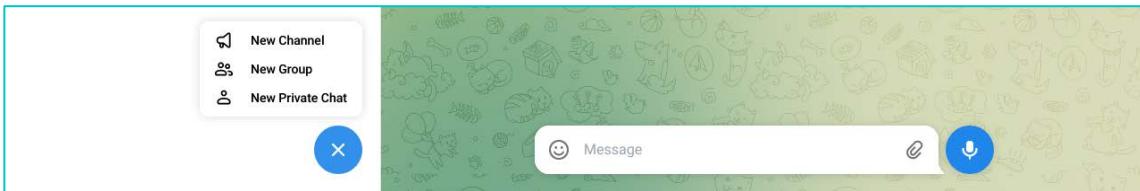


Figure 15: Create a Telegram group.

Then it will prompt to add members. Remember that when the bot was created it provides you a link to send a message to it. Here the username will be shown with an @ at the beginning of the username.

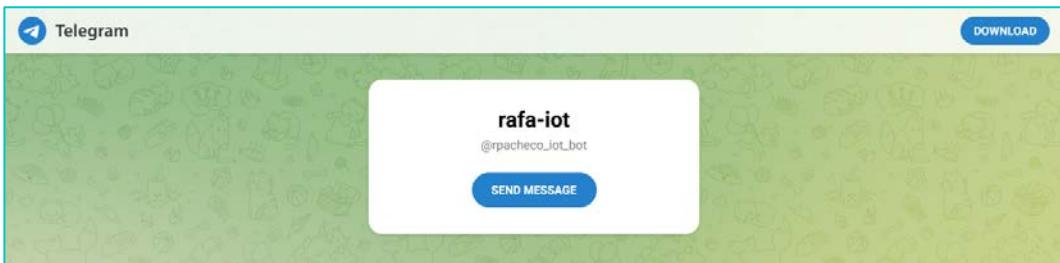


Figure 16: Profile of the bot created.

Add the bot as a Member of the group you are creating.

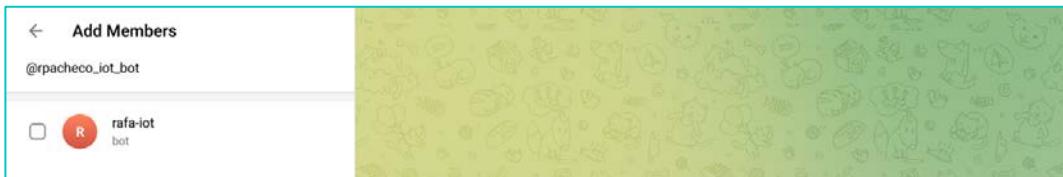


Figure 9: Add members to the telegram group.

When all the group members have been added beside the bot, you will be prompt to fill the Group Name.

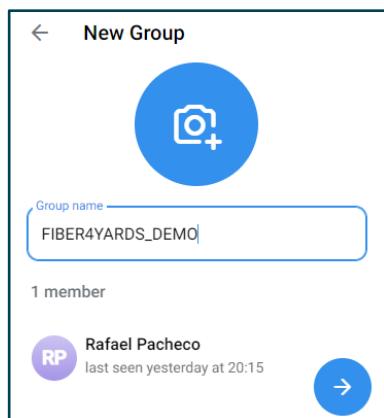


Figure 10: Fill the group name.

Then the group is created and this should be visible on the screen.

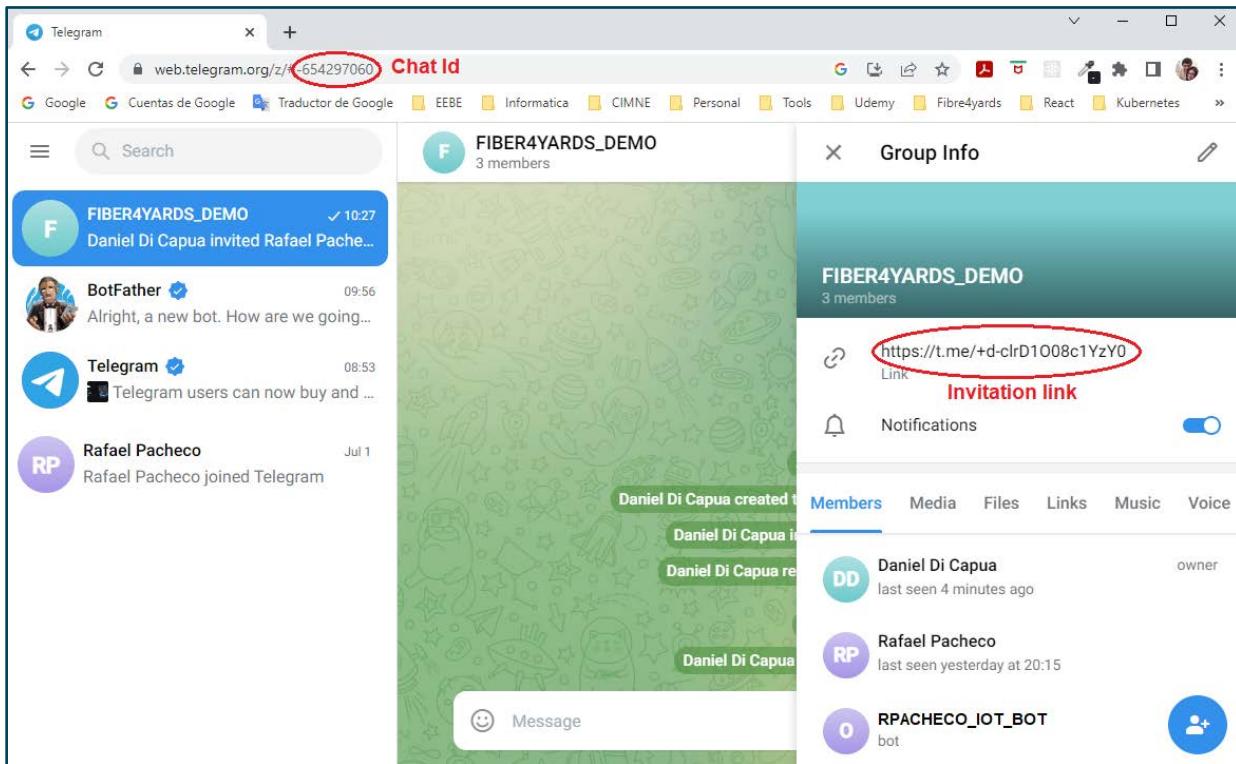


Figure 11: Telegram Group info.

When the group is created, now you have the three following things:

- **Bot token: 5342540378:AAHrJ4ABFiX54m6uf9RvxHxLRKeo0dGiHA0**
- **Chat id: -654297060**
- **Invitation Url: https://t.me/+d-clrD1O08c1YzY0**

Note that the **Bot token** is obtained from the BotFather chat, after filling the form. These three keys are needed later when creating an organization in the Command Line Interface (CLI) of the platform. In particular when initialising the platform.

1.3.1.3 Domain name

A domain name is a string of text that maps to an alphanumeric IP address, used to access a website from a browser. The domain name is used to identify a particular installation of the OSI4IOT platform.

The domain names are typically sold by domain name registrars. A domain name registrar is a company that allows you to purchase and register domain names. All domain name registrars are accredited by ICANN (Internet Corporation for Assigned Names and Numbers), a non-profit organization responsible for managing domain names.

The top domain registrars are:

- Domain.com (<https://www.domain.com/>)

- Bluehost: <https://www.bluehost.com/>
- Network Solutions: <https://www.networksolutions.com/>
- HostGator (<https://www.hostgator.com/>)
- GoDaddy: (<https://www.godaddy.com/>)
- Namecheap: (<https://www.namecheap.com/>)
- DreamHost: (<https://www.dreamhost.com/>)
- BuyDomains: (<https://www.buydomains.com/>)

1.3.1.4 *Email address*

The email address is used to send notifications from the platform. One of the easiest options to get an email address is with Gmail. To obtain a Gmail account follow the steps indicated in the following link:

<https://support.google.com/mail/answer/56256?hl=en>

For the platform to be able to send emails using the previously created Gmail account, it is necessary to obtain an applications password. To obtain an application password use the following link:

<https://support.google.com/mail/answer/185833?hl=en-GB>

1.3.2 *Command Line Interface (CLI) installer*

The OSI4IOT is installed using a command line interface (CLI) application called osi4iot_cli.

To start installing the CLI, download the correct installer based on your OS. In the GitHub repository, there are three installers in the subfolder

https://raw.githubusercontent.com/osi4iot/osi4iot/master/utils/osi4iot_cli/dist/

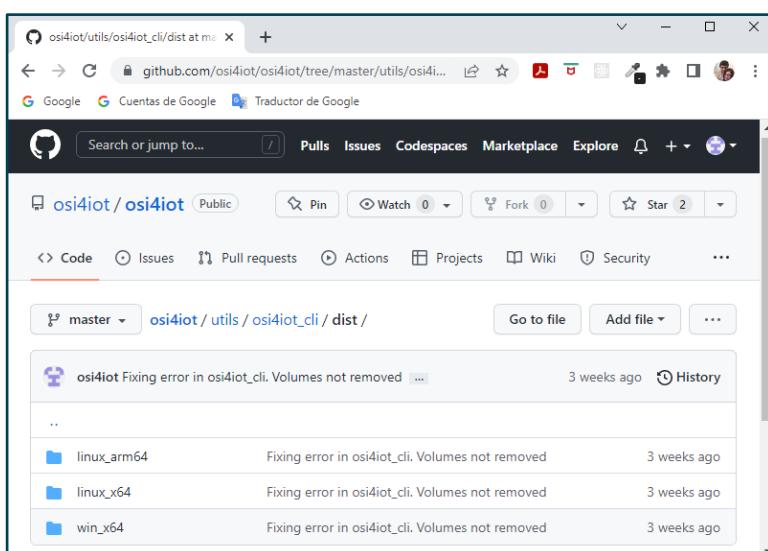


Figure 12: Location of the CLI installer. The folders for installation found in the subfolder indicate the type of operative system and architecture. Therefore:

Operative System	Architecture	Installer Folder
Windows	AMD (x64)	win_x64
Linux AMD	AMD (x64)	linux_x64
Linux ARM	ARM (arm64)	linux_arm64
Linux Mac	AMD (x64)	linux_x64

Figure 13: Summary of the CLI installers based on OS and architecture.

The ARM architecture in Linux is specially build for low consuming microprocessors such as Raspberry Pi or similar. These types of microprocessors are very common in the building of any IOT infrastructure.

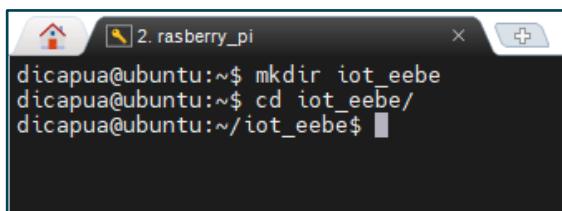
To proceed with the installation, download the correct installer and execute it in terminal (cmd or bash). A typical example for an UNIX-based system (e.g. Linux ARM) would be:

1. Create a folder to download the installer.

```
mkdir <my project> // Example: mkdir iot_fiber4yards
```

2. Change to the current directory.

```
cd <myproject>
```

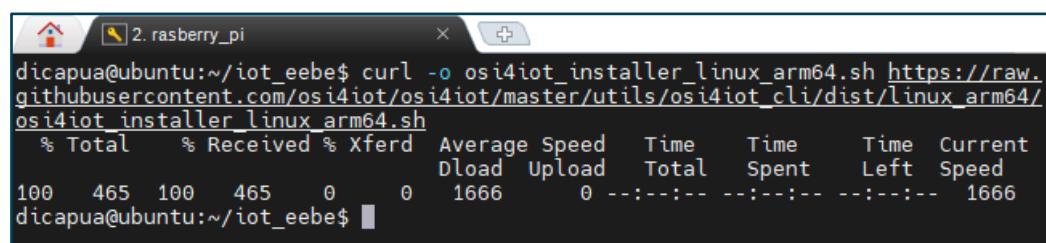
**Figure 14: Create the project.**

3. Download the correct installer.

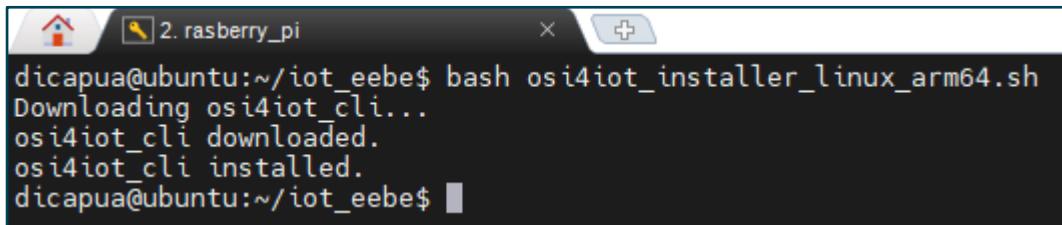
In this case the linux_arm64.sh installer, which is found in

https://raw.githubusercontent.com/osi4iot/osi4iot/master/utils/osi4iot_cli/dist/linux_arm64/osi4iot_installer_linux_arm64.sh or by using the following command

```
curl -o osi4iot_installer_linux_arm64.sh
https://raw.githubusercontent.com/osi4iot/osi4iot/master/utils/
osi4iot_cli/dist/linux_arm64/osi4iot_installer_linux_arm64.sh
```

**Figure 15: Download the installer using curl in Linux terminal.**

4. Execute the `installer.bash osi4iot_installer_linux_arm64.sh`



```
dicapua@ubuntu:~/iot_eebe$ bash osi4iot_installer_linux_arm64.sh
Downloading osi4iot_cli...
osi4iot_cli downloaded.
osi4iot_cli installed.
dicapua@ubuntu:~/iot_eebe$
```

Figure 16: Execute the installer.

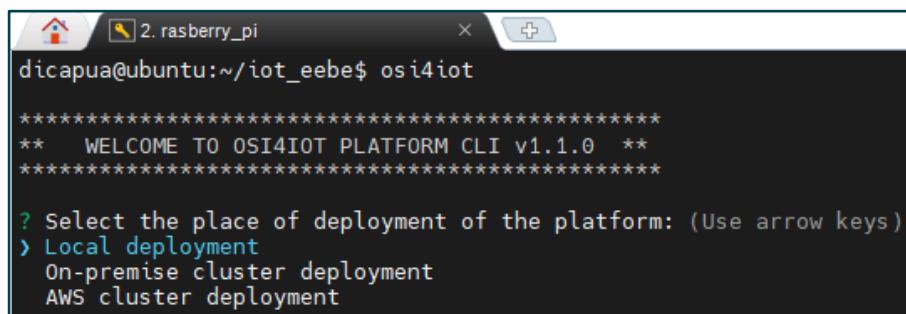
In the case of using the windows installer, the process is similar; however, it prompts the possibility to install it in a different drive.

With this, the **OSI4IOT** CLI has been installed in your system globally and the executable has been correctly updated in the PATH environment variable.

1.3.3 Running OSI4IOT CLI

Once the installer has been executed and the **OSI4IOT** CLI has been installed with no error, execute CLI by typing the following command `osi4iot`.

The following CLI screen appears:



```
dicapua@ubuntu:~/iot_eebe$ osi4iot
*****
** WELCOME TO OSI4IOT PLATFORM CLI v1.1.0 **
*****
```

? Select the place of deployment of the platform: (Use arrow keys)

- > Local deployment
- On-premise cluster deployment
- AWS cluster deployment

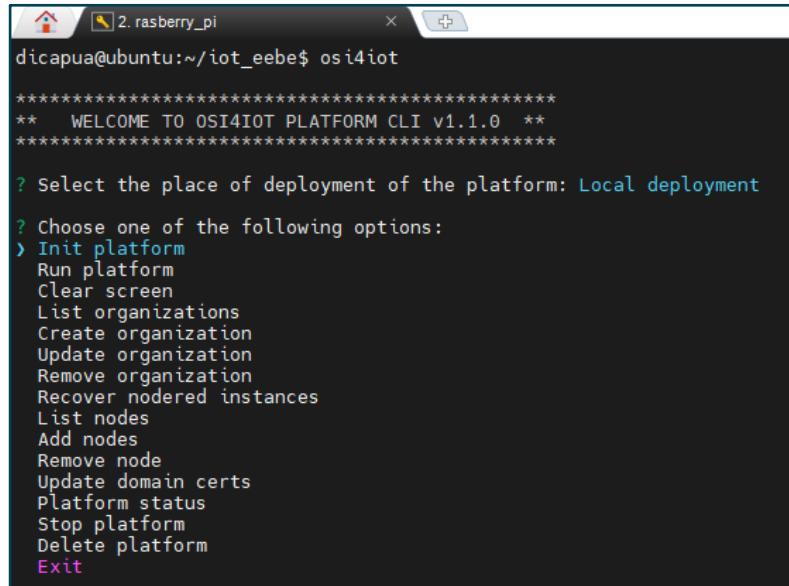
Figure 17: The CLI starting screen.

There are multiple deployment options:

- *Local deployment*: The platform is installed on a single machine.
- *On-premise cluster deployment*: The platform is installed on a cluster of machines that runs in your own premises.
- *AWS cluster deployment*: The platform is installed on a cluster in amazon AWS services.

1.3.3.1 Local deployment

When selecting the *Local deployment* option, the following options will be available:



```

dicapua@ubuntu:~/iot_eebe$ osi4iot
*****
** WELCOME TO OS4IOT PLATFORM CLI v1.1.0 **
*****  

? Select the place of deployment of the platform: Local deployment  

? Choose one of the following options:  

> Init platform  

  Run platform  

  Clear screen  

  List organizations  

  Create organization  

  Update organization  

  Remove organization  

  Recover nodered instances  

  List nodes  

  Add nodes  

  Remove node  

  Update domain certs  

  Platform status  

  Stop platform  

  Delete platform  

  Exit

```

Figure 18: Local deployment.

1. Init platform: Initiate the platform.
2. Run platform: to run it if stopped.
3. List organizations: retrieve the list of organisations.
4. Create organization: creates an organization.
5. Update organization: updates the information of an existing organization.
6. Remove organization: removes an organization.
7. List nodes: retrieve the list of nodes.
8. Add nodes: adds nodes to the platform.
9. Remove node: removes a node from the platform.
10. Update domain certs: update the certificates of the platform is certificates where selected.
11. Platform status: retrieves the status of the platform.
12. Stop platform: stops the platform.
13. Delete platform: deletes the platform.
14. Exits: exits the cli.

The first step in the deployment of a local service starts by selecting **Init platform**. When this option is selected, the following parameters need to be defined:

- Platform name: name of the platform.
- Domain name: name of the registered domain.
- Platform motivational phrase: a novel phrase to describe the platform.
- Platform admin first name.
- Platform admin last name.
- Platform admin username.
- Platform admin email.

- Platform admin password.
- Min longitude of the geographical zone of the platform: by default it will take from Spain. In this case the maximum longitude coordinate.
- Max longitude of the geographical zone of the platform: by default it will take from Spain. In this case the minimum longitude coordinate.
- Min latitude of the geographical zone of the platform: by default it will take from Spain. In this case the minimum latitude coordinate.
- Max latitude of the geographical zone of the platform: by default it will take from Spain. In this case, the maximum latitude coordinate.
- Default time zone.
- Main organization name: name of the organization.
- Main organization acronym: acronym of the organization.
- Main organization address: the address where the organization is located.
- Main organization city.
- Main organization zip code: ZIP or postal code of the organization.
- Main organization state/province: state or region of the country in which the organization is located.
- Main organization country: the country where the organization is located.
- Telegram bot token for main organization default group: it is the token obtained from the bot of telegram.
- Telegram chat id for main organization default group: group id of telegram of the organization where the bot of telegram is invited.
- Telegram invitation link for main organization default group: link needed to invite and grant access to members of an organization to the organization telegram group.
- Number of master devices in main organization: number of the master devices.
- Email account for platform notifications: e-mail used for the notifications of different alerts in an organization.
- Email account password: the password to access the e-mail for notifications.
- S3 storage bucket name: bucket name used in Minio or later to port to Amazon S3 service.
- Choose the type of domain ssl certs to be used: there are four possible configurations available

1. No Certs:

The simplest, run the platform without domain SSL certification.

2. Certs provided by a CA::

In this case, you will need to request the certificates to a Certificate Authority (CA).

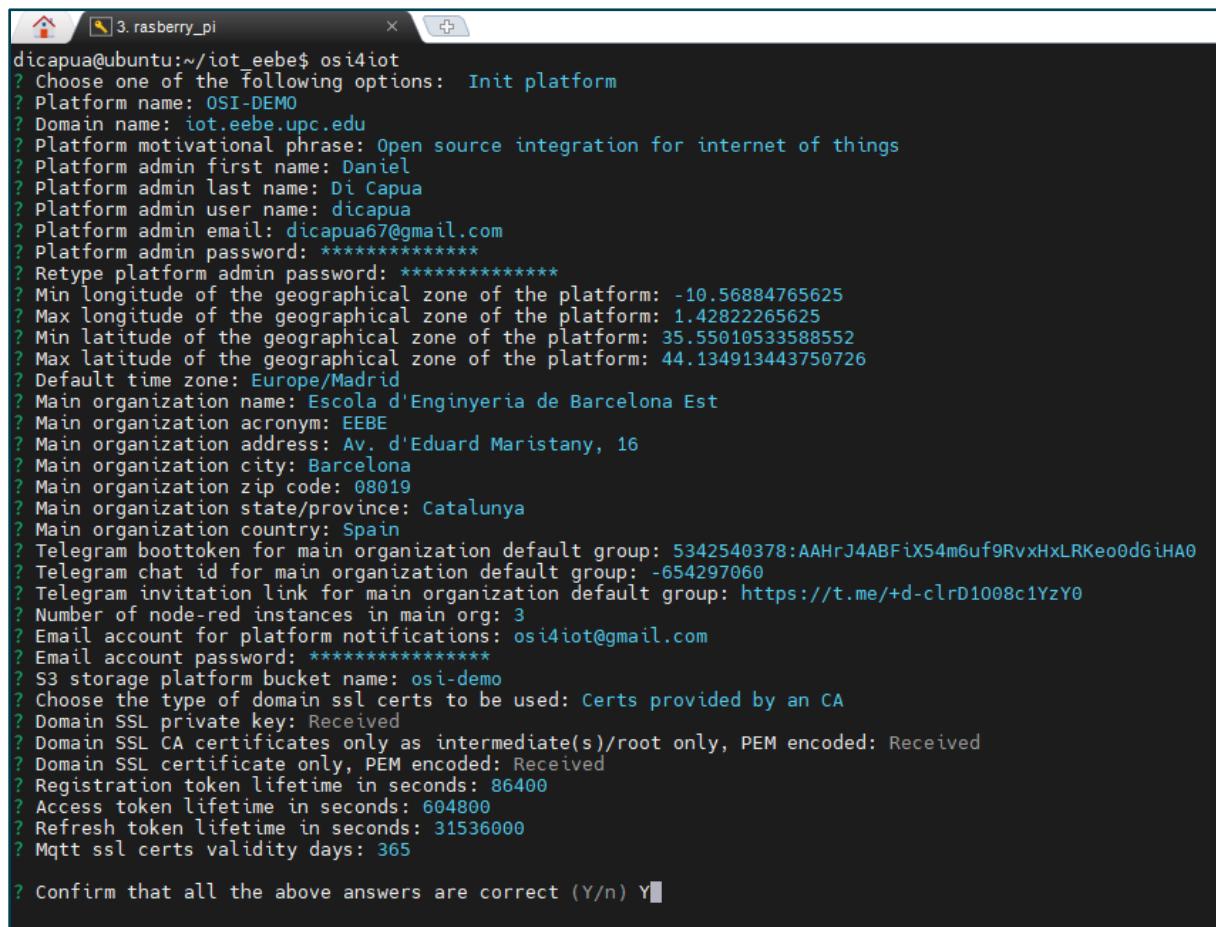
3. Let's encrypt certs and AWS Route 53:

The generation in this case is automatic but the Route 53 is a paid service.

4. AWS Certificate Manager:

This option is used when deploying the platform on an AWS cluster.

- Registration token lifetime in seconds
- Access token lifetime in seconds
- Refresh token lifetime in seconds
- Mqtt ssl certs validity days



```

dicapua@ubuntu:~/iot_eebe$ osi4iot
? Choose one of the following options: Init platform
? Platform name: OSI-DEMO
? Domain name: iot.eebe.upc.edu
? Platform motivational phrase: Open source integration for internet of things
? Platform admin first name: Daniel
? Platform admin last name: Di Capua
? Platform admin user name: dicapua
? Platform admin email: dicapua67@gmail.com
? Platform admin password: ****
? Retype platform admin password: ****
? Min longitude of the geographical zone of the platform: -10.56884765625
? Max longitude of the geographical zone of the platform: 1.42822265625
? Min latitude of the geographical zone of the platform: 35.55010533588552
? Max latitude of the geographical zone of the platform: 44.134913443750726
? Default time zone: Europe/Madrid
? Main organization name: Escola d'Enginyeria de Barcelona Est
? Main organization acronym: EEBE
? Main organization address: Av. d'Eduard Maristany, 16
? Main organization city: Barcelona
? Main organization zip code: 08019
? Main organization state/province: Catalunya
? Main organization country: Spain
? Telegram boottoken for main organization default group: 5342540378:AAHrJ4ABFiX54m6uf9RvxHxLRKeo0dGiHA0
? Telegram chat id for main organization default group: -654297060
? Telegram invitation link for main organization default group: https://t.me/+d-clrD1008c1YzY0
? Number of node-red instances in main org: 3
? Email account for platform notifications: osi4iot@gmail.com
? Email account password: ****
? S3 storage platform bucket name: osi-demo
? Choose the type of domain ssl certs to be used: Certs provided by an CA
? Domain SSL private key: Received
? Domain SSL CA certificates only as intermediate(s)/root only, PEM encoded: Received
? Domain SSL certificate only, PEM encoded: Received
? Registration token lifetime in seconds: 86400
? Access token lifetime in seconds: 604800
? Refresh token lifetime in seconds: 31536000
? Mqtt ssl certs validity days: 365

? Confirm that all the above answers are correct (Y/n) Y

```

Figure 19: Local deployment configuration.

Then confirm if all the above data is correct and the platform is installed as shown in Figure 20

```

? Confirm that all the above answers are correct Yes
Configuring nodes in the cluster...
Joining nodes to swarm;
Joining localhost to swarm...

Removing previous docker images and volumes...

Creating certificates...
Creating secrets...
Creating configs...
Creating stack file...

Generating node labels...

Deploying docker swarm stack:
Creating network osi4iot_default
Creating secret mqtt_certs_ca_cert_1c9e5e0f7d0874c3f0d4d6208f8e6f17
Creating secret eebe_my4YDpp0z1_key_0edddd3b1ab277712c0c1d6922f3b3
Creating secret mqtt_certs_ca_key_6994844f12f1fbef22f97724f3e5c6d
Creating secret eebe_ydq83wxGC7_key_c237702a8259aed4db4877e4bd610d72
Creating secret mqtt_broker_key_667e530f58bb755b520c175671b783b8
Creating secret eebe_UmgFdXRzT9_key_6fc9c84c936a8690b34f2cac385b7ed7
Creating secret iot_platform_ca_fb279493eb96f65f0c3190f09a0729a6
Creating secret iot_platform_key_620e2842c3992a668541663050dcda2e
Creating secret osi4iot_postgres_grafana
Creating secret osi4iot_postgres_password
Creating secret osi4iot_grafana
Creating secret osi4iot_postgres_user
Creating secret osi4iot_minio
Creating secret eebe_UmgFdXRzT9_cert_064682e9316c3bd56cae4d4c31cacf47
Creating secret osi4iot_dev2pdb_password
Creating secret mqtt_broker_cert_aee5f1662f65c278393c03d55a6f9a89
Creating secret iot_platform_cert_fd17d248b63f55def87bc08b45989ac3
Creating secret eebe_ydq83wxGC7_cert_fb1375f9e0958179fc73f3c7317c7d7
Creating secret eebe_my4YDpp0z1_cert_49aa4f5e7649dc783f5c5cb1c9774bf2
Creating secret admin_api_f2b4f6d260e50e1a0c28f50e42b2fb47
Creating config osi4iot_frontend_conf
Creating config osi4iot_mosquitto_conf
Creating config osi4iot_mosquitto_go_auth_conf
Creating config osi4iot_grafana_conf
Creating config osi4iot_admin_api_conf
Creating service osi4iot_org_eebe_nri_UmgFdXRzT9
Creating service osi4iot_portainer
Creating service osi4iot_system_prune
Creating service osi4iot_org_eebe_nri_ydq83wxGC7
Creating service osi4iot_org_eebe_nri_my4YDpp0z1
Creating service osi4iot_frontend
Creating service osi4iot_postgres
Creating service osi4iot_admin_api
Creating service osi4iot_grafana
Creating service osi4iot_traefik
Creating service osi4iot_minio
Creating service osi4iot_dev2pdb
Creating service osi4iot_agent
Creating service osi4iot_mosquitto
Waiting until all services be ready .

```

Figure 20: Local deployment with no cert configuration. Output after correctly installed.

1.4 Documentation

The **OSI4IOT** platform used a web user interface to be managed

1.4.1 Home Screen

The home screen of the platform displays four options

- Platform assistant
- Dashboards
- Digital twin simulator
- Mobile sensors (Only Android devices)

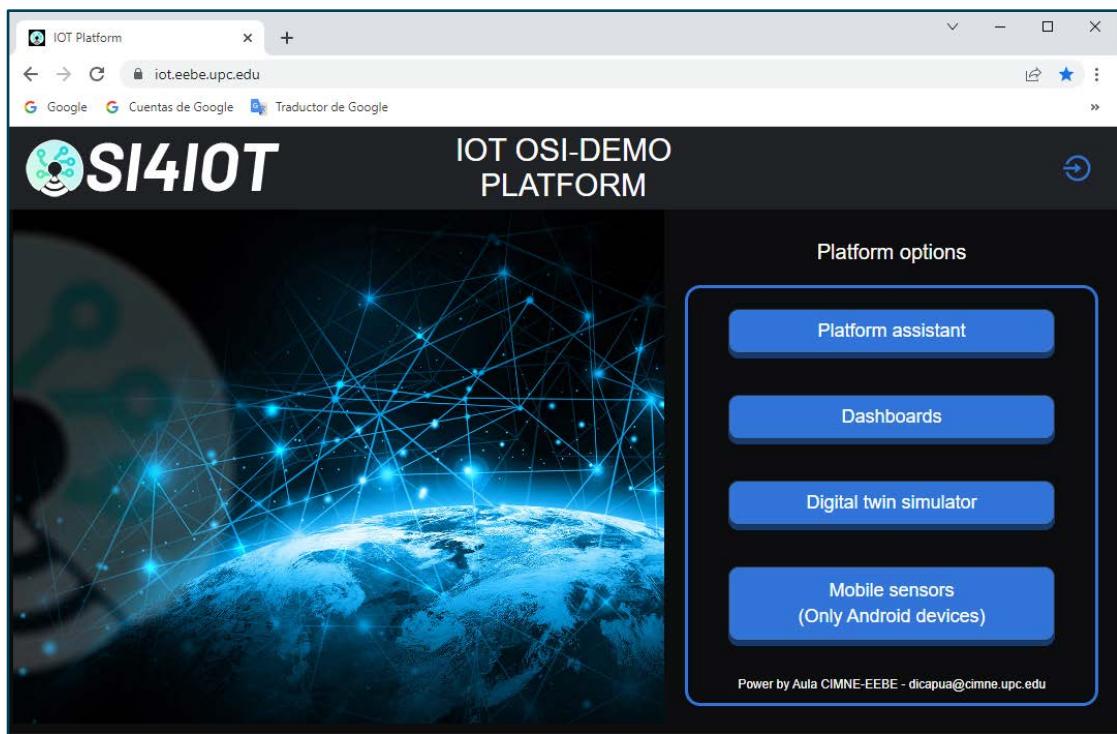


Figure 21: Home screen.

At the top right corner, you can click the **login** icon to log in. Any attempt to access the platform from the home screen without login, will be redirected to the **login page**.

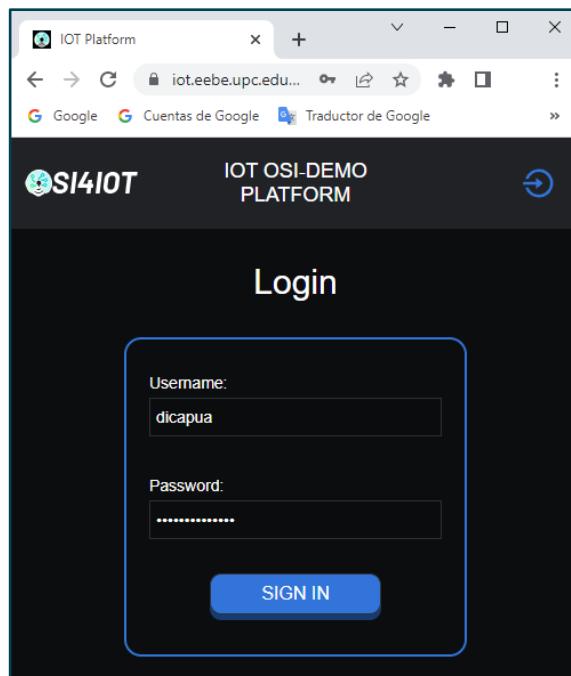


Figure 22: Login page.

1.4.2 Platform Assistant

When clicking on the Platform Assistant button in the Home Page. The user will be redirected to the Platform app. This page is compound of a vertical navigation bar that contains the different roles that a user can access. These are:

- Super Admin
- Organizations Admin
- Groups Admin
- User

The hierarchy of roles is establish as *Super Admin > Organizations Admin > Groups Admin > Users*. The actions that are enabled to a lower level role are accessible to a higher-level role, but not opposite.

The second element in the homepage of the platform assistant is the Geolocation, which is displayed by default. It show the outline of the buildings where the organizations are located.

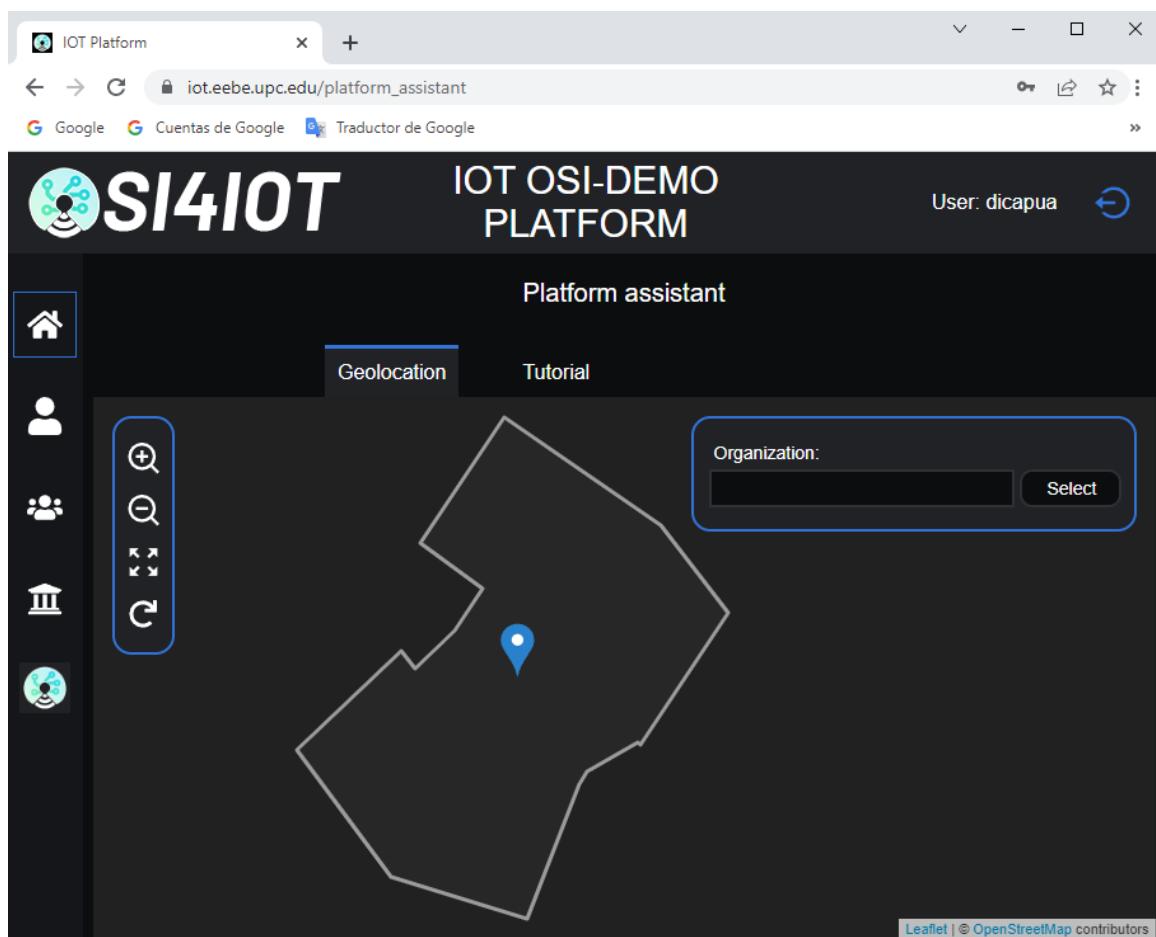


Figure 23: Geolocation screen.

1.4.3 Dashboards

The dashboard panel opens the Grafana application. When clicking the Dashboards button, it prompts the login page of Grafana.

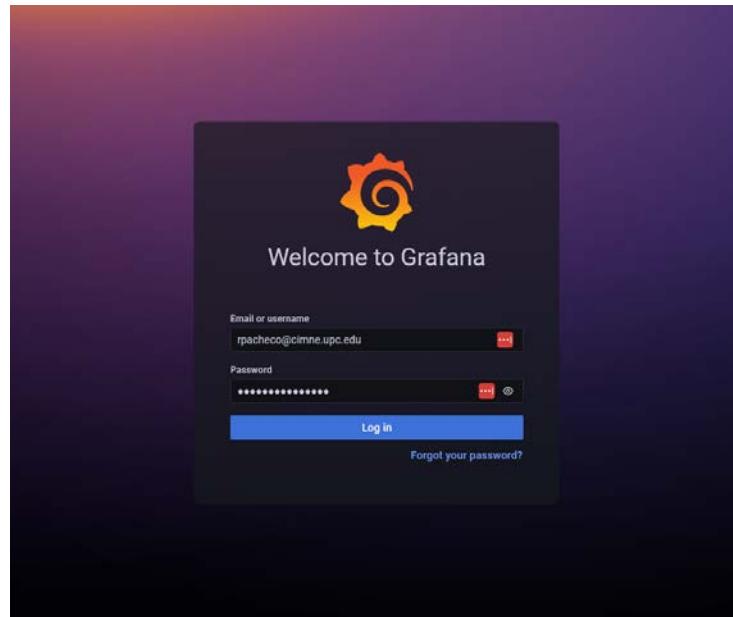


Figure 24: Grafana login page.

1.4.4 Digital Twin Simulator

The digital twin simulator is an application that allows you directly to access the different digital twin simulations available in the platform. This application is useful when needing to control a simulation from another device. For example from a mobile sensor.

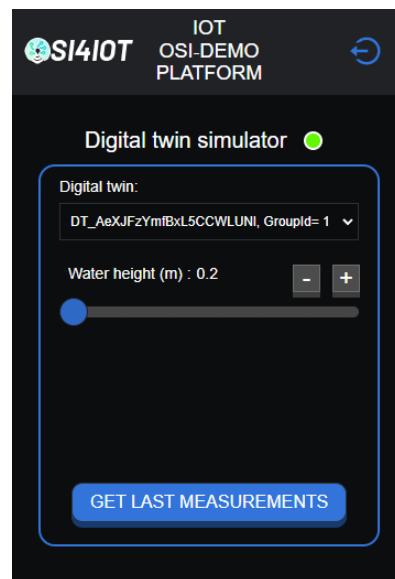


Figure 25: Digital twin simulator page.

1.4.5 Mobile Sensors (Only Android devices)

The option only works on mobile phones running Android. This application inside the platform is useful to demonstrate the capabilities of the platform with portable devices such as mobiles.

There are different types of possibilities with the integration of phones. One is to record the gyroscope of an Android phone and track it in real time. Other more complex features tested have been the acquisition of a snapshot via the phone camera and use techniques of machine learning to identify different objects shown in the snapshot.

1.4.6 OSI4IOT Platform Roles

The vertical navigation bar defines the possible roles that a user has. From bottom to top, the higher roles are displayed.

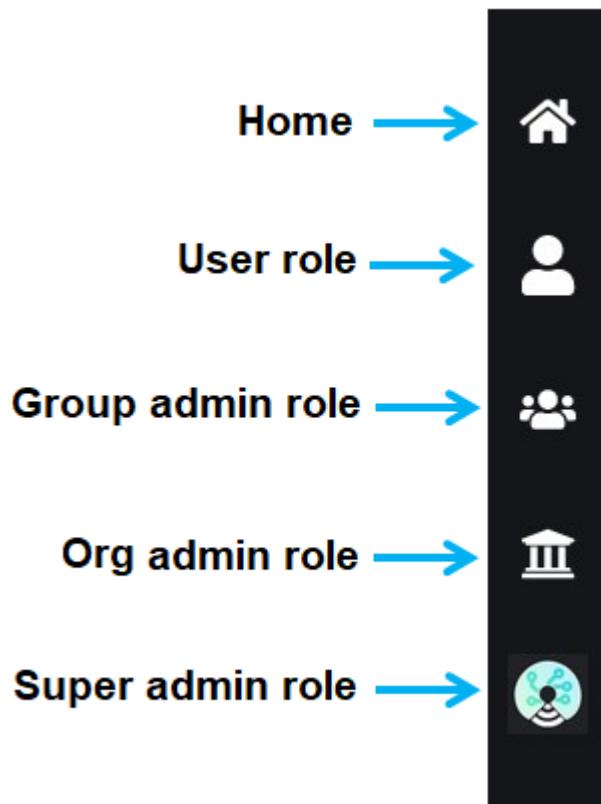


Figure 26: Role navigation bar.

The platform is designed using a hierarchical structure. Within this structure, there are four type of roles: Super Admin, Organizations Admin, Groups Admin and User.

The most important role would be the Super Admin, this role is dedicated to the members that manage the platform and install it through the CLI. The Super Admin role is in charge of creating the entire infrastructure necessary to manage organizations.

Organizations Admin role has access to the rest of options minus those specific of the Super Admin. Normally this role would be associated to those people who are in charge of managing an organization.

Within the organizations, there are groups, and each group has one or various Groups Admin. A group can be the departments within an organization or other organizations within a stakeholder such as vendors or providers that take part in some of the processes integrated in the digital twin model managed in the IOT platform.

Any other role, which does not fit at these levels, would fit in the category of a User member. These members have access to some of the devices, sensors, digital twin models stored in the platform. However, they cannot manage them.

1.4.7 Super admin role

The Super Admin role is the highest role available and it can be accessed when clicking in the Super Admin icon of the Role Navigation Bar. In Figure 27 is shown the default page of this role.

The screenshot shows a web browser window for the 'IOT Platform' at the URL 'iot.eebe.upc.edu/platform_assistant/admin'. The title bar says 'IOT OSi-DEMO PLATFORM'. The top navigation bar includes a 'User: dicapua' dropdown and a 'Logout' button. On the left, there's a vertical sidebar with icons for Home, People, and Organizations. The main content area is titled 'Platform assistant for superadmin' and features a 'Buildings' tab (which is selected), 'Floors', 'Organizations', 'Global users', 'Refresh tokens', and 'Tools'. Below this is a search and filter section with 'Page 1 of 1' and 'Show 10'. A table lists a single building entry:

ID	Name	Longitude	Latitude	Time from creation	Time from last update
1	Escola d'Enginyeria de Barcelona Est	2.2222129951594387	41.41368209515797	10m 18s 977.53ms	38.756ms

A blue button labeled 'New building' is located in the top right corner of the table area.

Figure 27: Super Admin role default page.

In the Super Admin role can be found the following tabs:

- Buildings
- Floors

- Organizations
- Refresh token
- Tools

1.4.7.1 Buildings

The default option is the buildings options. This is used to create the geolocation and geometry of the building where the organization is placed. Generally one building per organization is recommendable, since having buildings separated by a considerable length makes the geometry difficult to be displayed.

To add a building just click on **New Building**. A pop-out window will appear with the title create building. The fields to be completed are Building name, Longitude, Latitude and GeoJson data.

Longitude and Latitude can be left to 0 by default because these are updated automatically when inserting the GeoJson data. There is also the possibility to upload a local file to fill the GeoJson data field.

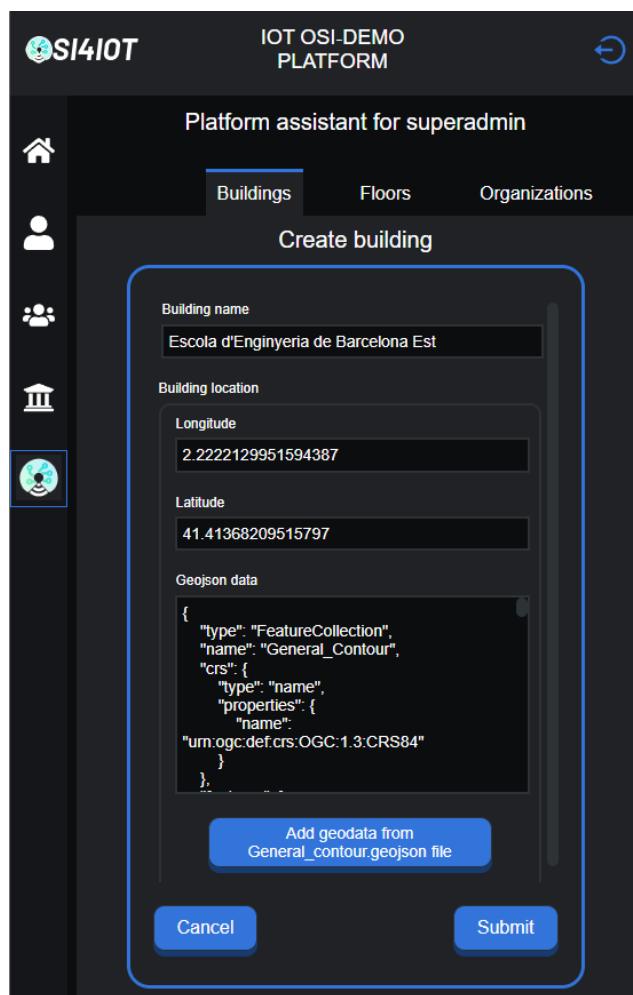


Figure 28: Creating a building.

To create the building geometry, use the <https://geojson.io/> web app to complete the GeoJson data field.

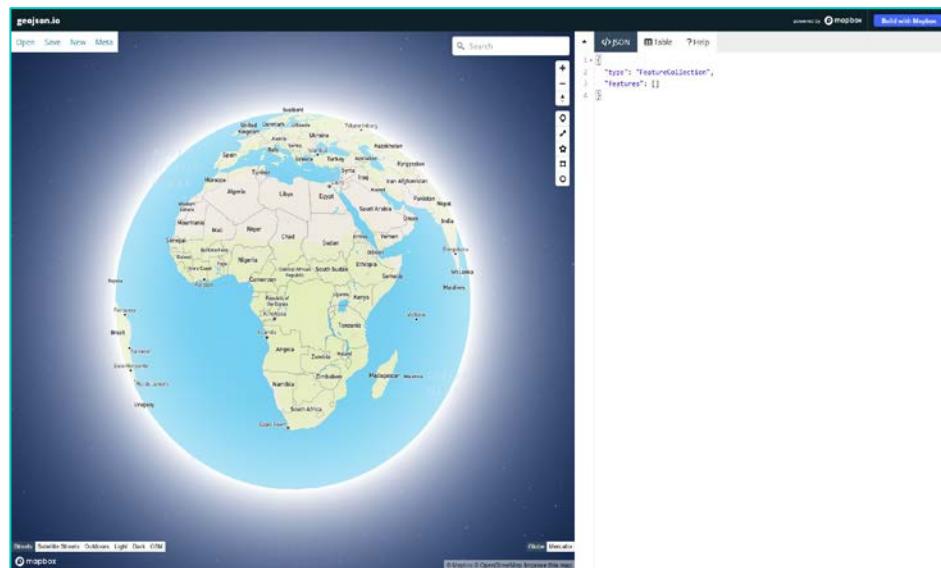


Figure 29: GeoJson.io home screen.

Create a polygon and copy the output JSON code.

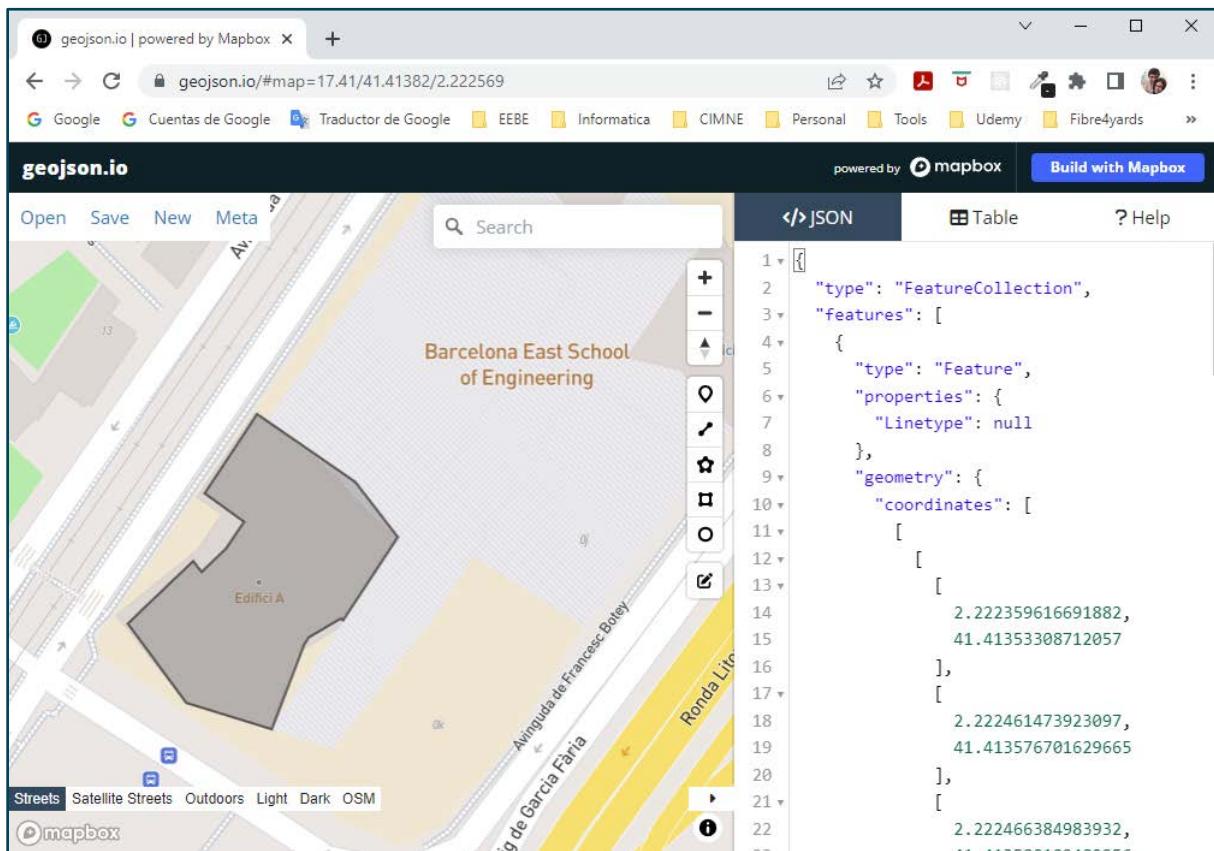


Figure 30: Creation of a polygon representing the outline of the building.

Paste the JSON code on the GeoJson data text area and then click Submit bottom. A green message should appear if the building has been correctly created, otherwise appear a red message.

If the **Edit icon** is pressed, the following pop-out window appears. Note that the Longitude and Latitude fields are updated automatically.

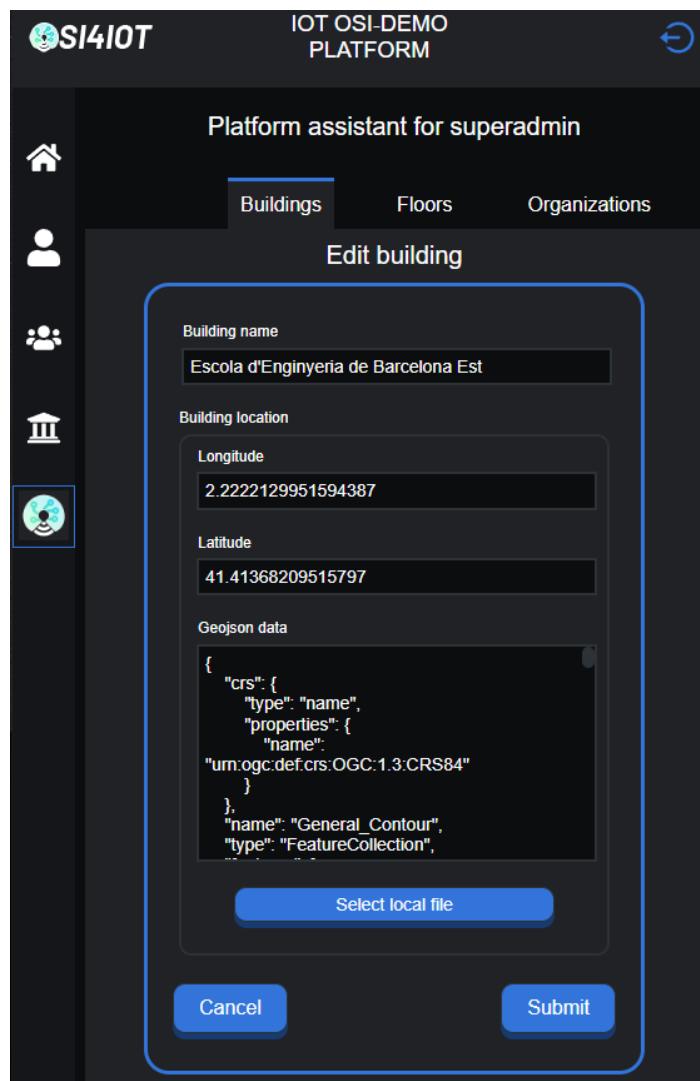


Figure 31: Building edition.

If anything should be changed, click submit. A green message should appear if the building has been correctly updated, otherwise appear a red message.

The parameters of Building table are the following:

1. Id: enumerator associated to the building created.
2. Name: the name of the building.
3. Longitude: longitude of the building. Autofilled by default when created.
4. Latitude: latitude of the building. Autofilled by default when created.

5. Time from creation: the time at which the building was created.
6. Time from last update: the time at which the building was updated. Floors

Id	Name	Longitude	Latitude	Time from creation	Time from last update
1	Escola d'Enginyeria de Barcelona Est	2.2222129951594387	41.41368209515797	17d 16h 20m 9s 53.017ms	17d 16h 9m 47s 563.675ms

Figure 32: Parameters of Buildings table.

1.4.7.2 Floors

The floors tab displays the list of floors present in the different created buildings as can be seen in Figure 33

Id	Building Id	Building name	Floor number	Time from creation	Time from last update
1	1	Escola d'Enginyeria de Barcelona Est	0	15h 3m 32s 923.065ms	14h 53m 4s 956.728ms
2	1	Escola d'Enginyeria de Barcelona Est	1	14h 32m 47s 115.647ms	14h 32m 47s 115.647ms
3	1	Escola d'Enginyeria de Barcelona Est	2	14h 32m 10s 561.065ms	14h 32m 10s 561.065ms

Figure 33: Floors table.

To create a floor, can be used <https://geojson.io/> web app. It must be added the field "name": "Floor_0" to determine the ground floor. Note that the outline of the building and the floor 0 must be the same.

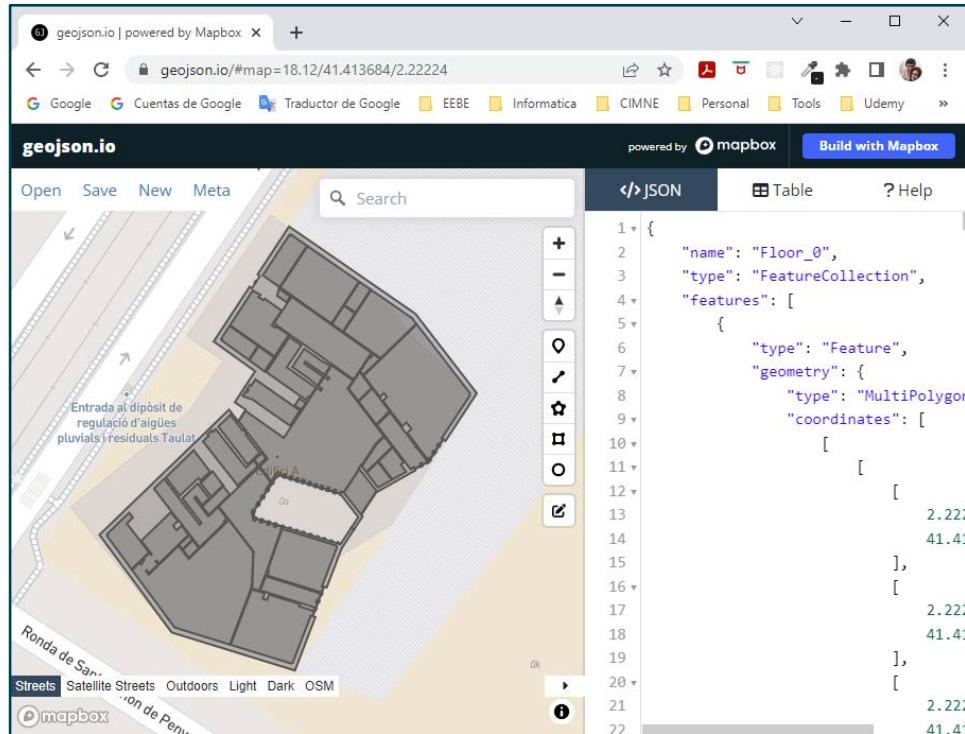


Figure 34: Creation of a floor in geojon.io application.

Insert the output JSON code into the Create floor pop-out window. In here select the Building Id and in the floor location, input the Floor number and the GeoJson data or upload it from a local file. Click submit.

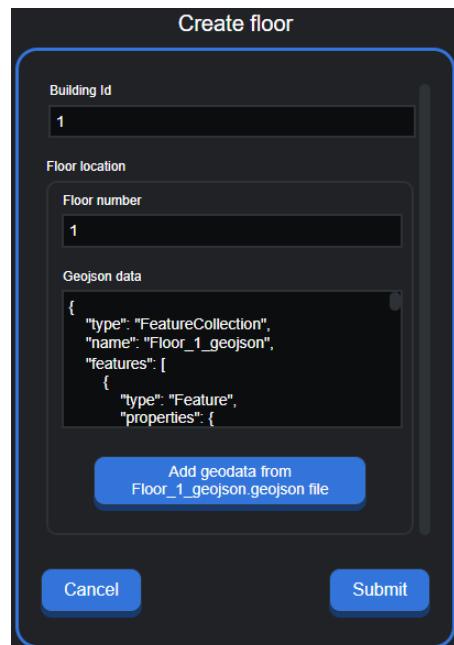


Figure 35: Create floor with GeoJSON data.

In the Figure 36 can be seen the parameters of Floors table:

1. Id: enumerator associated to the floor created.
2. Building Id: enumerator associated to the building where the floor is placed.
3. Building name: the name of the building.
4. Floor number: the number of the floor.
5. Time from creation: the time at which the floor was created.
6. Time from last update: the time at which the floor was updated.

1	2	3	4	5	6
<input type="text"/> Search	<input type="text"/> Search	<input type="text"/> Search	<input type="text"/> Search		
1	1	Escola d'Enginyeria de Barcelona Est	0	31m 22s 553.572ms	20m 54s 587.235ms
2	1	Escola d'Enginyeria de Barcelona Est	1	36s 746.154ms	36s 746.154ms
3	1	Escola d'Enginyeria de Barcelona Est	2	191.572ms	191.572ms

Figure 36: Parameters of Floors table.

1.4.7.3 Organizations

Organizations are created and edited in the **CLI**. In the WUI can be see the table of created organizations.

Id	Name	Acronym	Address	City	Zip code	State	Country	Building Id	Org hash	Mqtt acc
<input type="text"/> Search	<input type="text"/> Search	<input type="text"/> Search								
1	Escola d'Enginyeria de Barcelona Est	EEBE	Av. d'Eduard Maristany, 16	Barcelona	08019	Catalunya	Spain	1	TXAglyspzSMeMcwu	Pub & Sub

Figure 37: Organizations table.

The parameters of Organizations table are the following:

1. Id: enumerator associated to the organization created.
2. Name: the name of the organization.
3. Acronym: the acronym used to identify the organization.
4. Address: the address where the organization is located.
5. City: the city of the address.
6. Zip code: the postal code.
7. State: the state or province where the organization is located.
8. Country: the country where the organization is located.
9. Building Id: the enumerator of the building associated to the organization.
10. Org hash: a unique identifier hash of the organization.
11. Mqtt acc: type of access control of the MQTT protocol for the organization.

IOT OSI-DEMO
PLATFORM

User: dicapua

Platform assistant for superadmin

1	2	3	4	5	6	7	8	9	10	11
<input type="text"/> Search	<input type="text"/> Search	<input type="text"/> Search								<input type="text"/> Mqtt acc
1	Escola d'Enginyeria de Barcelona Est	EEBE	Av. d'Eduard Maristany, 16	Barcelona	08019	Catalunya	Spain	1	TXAglyspzSMeMcwu	Pub & Sub

Figure 38: Parameters of Organizations table.

There are four types of access control for the MQTT protocol, which can be applied to any element (organization, group, device or topic):

- Subscribe & Publish: The element can subscribe or publish to its related topics.
- Subscribe: The element can only subscribe to its related topics.
- Publish: The element can only publish to its related topics.
- None: The element can do nothing.

1.4.7.4 Global Users

In the global users tab, a list of the global users is defined. A global user is a generic user of the platform that can be assigned to one or more organizations. A platform user who does not belong to any organization is still a global user of the platform.

A global user can have the role of platform admin or not.

Id	First Name	Surname	Username	Email	Role	Seen
2	Daniel	Di Capua	dicapua	dicapua67@gmail.com	Admin	10y 15h
5	Rafael	Pacheco	rpacheco	rpacheco@gmail.com		10y 13h

Figure 39: Global user table.

To create a new global user, click on the **New global user** button. Then fill the fields: *Name*, *Surname*, *Email*, *Username* and *Password*.

Figure 40: Creation of a new global user.

If you need to add more global users, click the **Add global user** button or if you have them defined in a local file, click the **Select local file** button. Once you are done click the **Submit** button.

If you need to edit a global user, click the Edit icon and the following pop-up window will appear as shown in Figure 41.

Edit global user

First name	Rafael
Surname	Pacheco
Email	rpacheco@gmail.com
Username	rpacheco
Password	[REDACTED]
Platform admin	No
<input type="button" value="Cancel"/> <input type="button" value="Submit"/>	

Figure 41: Edit global user.

The parameters of Global users table are the following:

1. Id: enumerator associated to the user created.
2. First Name: the name of the user created.
3. Surname: the surname of the user created.
4. Username: the username of the user created.
5. Email: the email of the user created.
6. Role: the type of role associated to the user created.
7. Seen: The last time the user was logged.

IOT OSI-DEMO PLATFORM

User: dicapua [Logout](#)

Platform assistant for superadmin

Buildings Floors Organizations Global users Refresh tokens Tools

Create new global user [New global user](#)

1	2	3	4	5	6	7
Id	First Name	Surname	Username	Email	Role	Seen
2	Daniel	Di Capua	dicapua	dicapua67@gmail.com	Admin	10y 1h
5	Rafael	Pacheco	rpacheco	rpacheco@gmail.com		10y 5s

[Edit global user](#) [Delete global user](#)

Figure 42: Parameters of Global users table.

1.4.7.5 Refresh Tokens

When a user is logged in, a token is created using the JSON Web Token (JWT) encryption technology. To renew periodically this token from the browser, a refresh token is created simultaneously. These tokens are stored in the browser local storage. In the Refresh tokens tab can be seen the table of the refresh tokens

Platform assistant for superadmin				
	Buildings	Floors	Organizations	Global users
	Refresh tokens			
1	2	eyJhbGciOiJIUzI1Ni... NzAzMDAxMzIzQ.ME2w2eIKFwGldoPiVxZ9Rv51gXgAXFRSNNDNzNbZUw	14h 56m	14h 56m
2	5	eyJhbGciOiJIUzI1Ni... DMwMDkyOTJ9.FpFw0O4ZLadsBugDU25Ldmxo2zDe_08rkFoGoXSZsEM	12h 43m	12h 43m

Figure 43: Refresh token table.

The parameters of Refresh tokens table are the following:

1. Id: enumerator associated with the refresh token.
2. Userid: enumerator associated with the user that is associated with the refresh token.
3. Refresh tokens: a generated hash representing the refresh token.
4. Created: time of creation of the refresh token.
5. Updated: time when the refresh token was updated.

Platform assistant for superadmin					
	Buildings	Floors	Organizations	Global users	
	Refresh tokens				
1	2	3	4	5	
2	5	eyJhbGciOiJIUzI1Ni... DMwMDkyOTJ9.FpFw0O4ZLadsBugDU25Ldmxo2zDe_08rkFoGoXSZsEM	2h 13m	2h 13m	

Figure 44: Parameters of Refresh tokens table.

1.4.7.6 Tools

List of the tools available inside the platform.

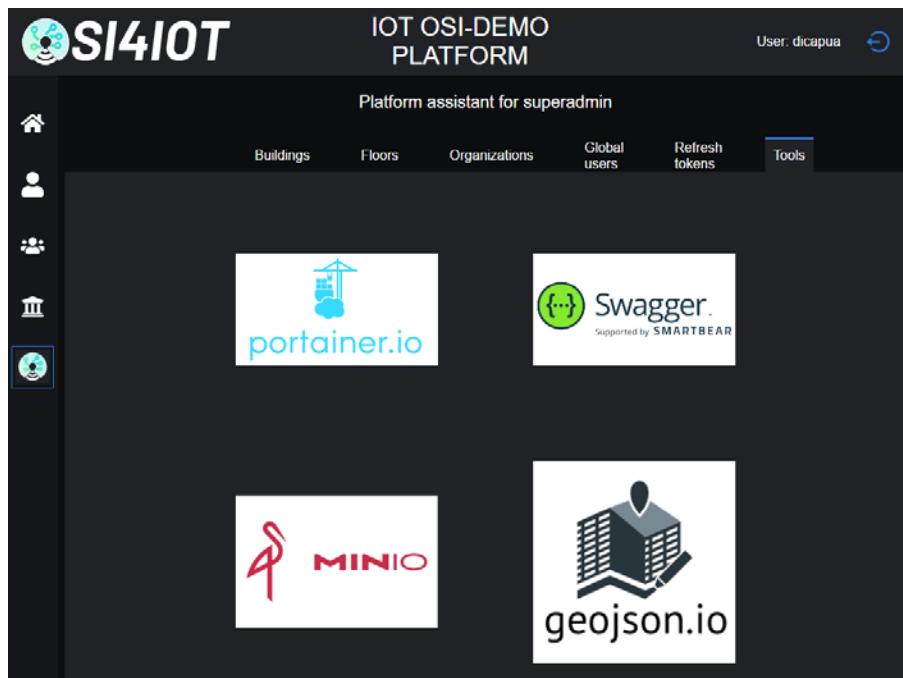


Figure 45: Tools page.

- Portainer.io: a web app for manage the Docker containers of the platform.
- Swagger: a platform to document the API.
- Minio: an S3 storage application.
- geojson.io: a platform that aids in the creation of geolocation data in JSON format.

1.4.8 Organization admin role

The Org admin role is the second highest role available and it can be accessed when clicking on corresponding icon of the Role Navigation Bar. The Org admin role page has the following tabs as shown in Figure 46:

- Orgs managed
- Org users
- Groups
- Nodered instances in group

ID	Name	Acronym	Address	City	Zip code	State	Country	Org hash	Mqtt acc	Add users	Remove all users
1	Escola d'Enginyeria de Barcelona Est	EEBE	Av. d'Eduard Maristany, 16	Barcelona	08019	Catalunya	Spain	TXAglypzSMeMcwu	Pub & Sub		

Figure 46: Default page of the Org admin role.

1.4.8.1 Organizations Managed

The first tab shows the list of organizations that the logged manages. This screen shows relevant information about the organization and it offers the possibility to add or remove all users by clicking the icons.

The important feature found in this tab is the possibility to add users to a certain organization and to remove all of them.

The parameters of Orgs Managed table are the following:

1. Id: enumerator associated to the association.
2. Name: the name of the association.
3. Acronym: The identifier of the association.
4. Address: the address where the organization is located.
5. City: the city of the address.
6. Zip code: the postal code.
7. State: the state or province where the organization is located.
8. Country: the country where the organization is located.

9. Org hash: a unique identifier hash of the organization.
10. Mqtt acc: type of access control of the MQTT protocol for the organization.
11. Add users: click on the icon to add users to the organization.
12. Remove all users: click on the icon to remove all users of the organization.

IOT OSI-DEMO PLATFORM

User: dicapua

Platform assistant for orgs admin

Orgs managed

Org users Groups Nodered instances in orgs

Page 1 of 1 | Go to page: 1 Show 10 Global search

Id	Name	Acronym	Address	City	Zip code	State	Country	Org hash	Mqtt acc	Add users	Remove all users
1	Escola d'Enginyeria de Barcelona Est	EEBE	Av. d'Eduard Maristany, 16	Barcelona	08019	Catalunya	Spain	TXAGiyjspzSMcMwu	Pub & Sub		

Figure 47: Parameters of Orgs managed table.

1.4.8.2 Organizations Users

The second tab shows the user list for the organizations listed in the Organizations Users tab. Here you can edit or erase a particular organization user and its roles within the organization hierarchy. An organization user can has the role of admin or viewer.

IOT OSI-DEMO PLATFORM

User: dicapua

Platform assistant for orgs admin

Orgs managed

Org users Groups Nodered instances in orgs

Page 1 of 1 | Go to page: 1 Show 10 Global search

OrgId	UserId	First Name	Surname	Username	Email	Role	Seen
1	2	Daniel	Di Capua	dicapua	dicapua67@gmail.com	Admin	10y 15h
1	5	Rafael	Pacheco	rpacheco	rpacheco@gmail.com	Viewer	10y 14h

Figure 48: Organizations user table.

The parameters of Orgs user table are the following:

1. OrgId: enumerator associated to the Organization that the user belongs to.
2. UserId: enumerator associated to the user.
3. First Name: the name of the user.
4. Surname: the surname of the user.
5. Username: the username of the user.
6. Email: the email of the user.
7. Role: the type of role associated to the user.
8. Seen: The last time the user was logged.

1	2	3	4	5	6	7	8
1	2	Daniel	Di Capua	dicapua	dicapua67@gmail.com	Admin	10y 15h
1	5	Rafael	Pacheco	rpacheco	rpacheco@gmail.com	Viewer	10y 14h

Figure 49: Parameters of Org users table.

1.4.8.2.1 Groups

The groups tab shows the different groups associated with an organization.

GroupId	OrgId	Name	Acronym	Type	Folder permission	Group Hash	Telegram invitation link	ChatId	Floor number	Feature index	Mgt acc
1	1	Escola d'Enginyeria de Barcelona Est general	EEBE_GRAL	Default	Viewer	f4b0e58a1_0d8c_4cde_b66c_a716ca845fad	https://t.me/joinchat/AibDfEmLxwDMmT7_-	-537434692	0	12	Pub & Sub
2	1	Elasticity	ELAS	Generic	Viewer	b30e58fc_bb32_447a_904d_aa1fbba2a5563	https://t.me/+d-polD18602RTV38	-597813945	0	36	Pub & Sub

Figure 50: Groups table.

When creating or editing a group appear the pop-out window shown in Figure 51. The first field to fill in is the ID of the organization to which the group belongs. Then complete the name of group, acronym and the folder permission (admin, editor or viewer). The MQTT access control for the group also must be defined.

From the tutorial of the telegram bot, follow the indications to obtain the telegram invitation link and telegram chat id to fill in this form.

The group location is defined by the floor number and the feature index. The feature index has to be defined in the geojson data for the polygon that represents the group location. Finally, select a user to be the group admin and press submit.

Figure 51: Creation of a Group.

The parameters of Groups table are the following:

1. GroupId: enumerator associated to the group created.
2. OrgId: enumerator associated to the Organization that the group belongs.

3. Name: the name of the organization.
4. Acronym: The identifier associated to the group.
5. Type: The type of group can be default or generic. A default group is automatically generated when an organization is created.
6. Folder permission: the privileges associated to the group.
7. Group Hash: the hash associated to the group that is used in the MQTT protocol.
8. Telegram invitation link: invitation link to the telegram channel of the notification bot.
9. ChatId: identification number associated to the chat where the telegram bot is.
10. Floor number: is the floor number where the group is located.
11. Feature index: is the feature index found in the geojson data, for the particular GroupId.
12. Mqtt acc: type of access control of the MQTT protocol for the group.

1	2	3	4	5	6	7	8	9	10	11	12
Groupid	Orgid	Name	Acronym	Type	Folder permission	Group Hash	Telegram invitation link	ChatId	Floor number	Feature index	Mqtt acc
1	1	Escola d'Enginyeria de Barcelona Est general	EEBE_GRAL	Default	Viewer	f4b568a1_0d8c_4c8e_b66c_a716ca845fad	https://t.me/joinchat/AbDEmLxwDMmT7_-537434892	0	12		<input checked="" type="checkbox"/> Pub & Sub
2	1	Elasticity	ELAS	Generic	Viewer	b30e69c_bb32_447a_904d_aafba2a5563	https://t.me/+dpolD18602RTV38	-597813945	0	36	<input checked="" type="checkbox"/> Pub & Sub

Figure 52: Parameters of Groups table.

1.4.8.2.2 Nodered Instances in Organizations

The **Nodered Instances** (NRI) are instances of the Node-Red application, which are associated to an organization. In the Nodered instances are developed the logic of the digital twin models.

The number of NRI assigned to an organization is defined when this organization is created by the **CLI**.

The screenshot shows a web browser window titled 'IOT Platform' with the URL 'iot.eebe.upc.edu/platform_assistant/org'. The page header includes the 'SI4IOT' logo, the title 'IOT OSi-DEMO PLATFORM', and a user session 'User: dicapua'. On the left, there is a vertical sidebar with icons for Home, User, Groups, and a database. The main content area is titled 'Platform assistant for orgs admin' and features a table titled 'Nodered instances in orgs'. The table has columns: Id, Org Id, Group Id, Nodered instance hash, Longitude, Latitude, and Deleted. The data in the table is as follows:

Id	Org Id	Group Id	Nodered instance hash	Longitude	Latitude	Deleted
1	1	1	Tuxu6Wm7UF	2.2225182346099928	41.413806953914396	No
2	1	2	5jsz12iWUv	2.221995766428904	41.41348068619644	No
3	1	-	ej5cR3yIWo	0	0	No

Figure 53: Table of Nodered instances in orgs.

The parameters of the NRI in orgs table are the following:

1. Id: enumerator associated to the device.
2. Org Id: enumerator associated to the Organization that the group belongs to.
3. Group Id: enumerator associated to the group id that the device belongs to.
4. Nodered instance hash: the hash associated tot he node red instance.
5. Longitude: longitude where the nodered instance is placed.
6. Latitude: latitude where the nodered instance is placed.
7. Deleted: Indicate if the NRI has been deleted or not.

This screenshot is identical to Figure 53, but the columns are numbered 1 through 7 above the table header. The table data remains the same as in Figure 53.

Figure 54: Table of Nodered instances in orgs.

1.4.9 Group admin role

The Groups Admin role is the third highest role available and it can be accessed when clicking on the corresponding icon of the Role Navigation Bar. In the Group admin role can be found the following tabs:

- Groups managed
- Group members
- Devices
- Topics
- Measurements
- Dashboards
- Digital twins

In Figure 55 can be seen the default page of the Group admin role.

GroupId	OrgId	Name	Acronym	Type	Folder permission	Group hash	Mqtt acc	Floor number	Add members	Remove all members	Change hash
1	1	Escola d'Enginyeria de Barcelona Est general	EEBE_GRAL	Default	Viewer	f4b588a1_0d8c_4c8e_b86c_a716ca845fad	Pub & Sub	0			
2	1	Elasticity	ELAS	Generic	Viewer	b30e68fc_bb32_447a_904d_aafba2a5563	Pub & Sub	0			

Figure 55: Group admin role tabs.

1.4.9.1 Groups managed

This tab shows the list of groups managed by the Group Admin role. The Group Admin can edit the parameters of the groups that is in charge of managing.

The parameters of Groups managed table are the following:

1. GroupId: enumerator associated to the group created.
2. OrgId: enumerator associated to the Organization that the group belongs to.
3. Name: the name of the organization.
4. Acronym: The identifier associated to the group.
5. Type: The type of group. A default group is automatically generated when an organization is created.
6. Folder permission: the privileges associated to the group.

7. Group Hash: the hash associated to the group that is used in the MQTT protocol.
8. Mqtt acc: type of access control of the MQTT protocol for the group.
9. Floor number: is the floor number where the group is located.
10. Add members: click the icon to add members to the group.
11. Remove all members: click the icon to remove all members of the group.
12. Change hash: click to regenerate the group hash.

1	2	3	4	5	6	7	8	9	10	11	12	
GroupId	OrgId	Name	Acronym	Type	Folder permission	Group hash	Mqtt acc	Floor number	Add members	Remove all members	Change hash	
1	1	Escola d'Enginyeria de Barcelona Est general	EEBE_GRAL	Default	Viewer	f4b588a1_0d8c_4c8e_b86c_a716ca845fad	Pub & Sub	0				
2	1	Elasticity	ELAS	Generic	Viewer	b30ef8fc_bb32_447a_904d_aafbbba2a5563	Pub & Sub	0				

[Edit group managed](#)

Figure 56: Parameters of Groups managed table.

When the edit icon is clicked, the pop-out window show in Figure 57 is open. It allows to modify the group identifier, the organization identifier, the name of the group, the acronym associated to it, the folder permission, and the telegram invitation link and chat id of the notification bot.

Figure 57: Edition of a Group managed.

The node-red instance icon location and size can be modified.

1.4.9.2 Group members

The group member tab show the list of members associated with a group. A user with group admin privileges can edit or remove users.

Groupid	Userid	First Name	Surname	Username	Email	Role
1	2	Daniel	Di Capua	dicapua	dicapua67@gmail.com	Admin
2	2	Daniel	Di Capua	dicapua	dicapua67@gmail.com	Admin
1	5	Rafael	Pacheco	rpacheco	rpacheco@gmail.com	Viewer

Figure 58: Group members table.

When the edit button is clicked, the following pop-out window appears. In here, the first name, surname, email, username and the member role in the group can be modified. The member role in the group can be selected from viewer, editor and admin.

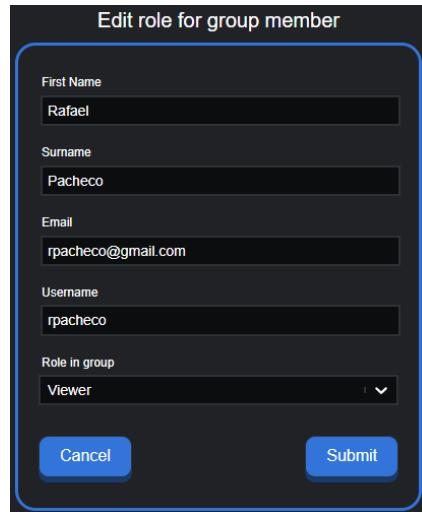


Figure 59: Edit role for group member.

The editor role allows to the group member to create and edit devices, topics, dashboards and digital twins. However, with editor role a group member cannot add new members to the group.

The parameters of Group members table are shown in Figure 60:

1. GroupId: enumerator associated to the group created.
2. UserId: enumerator associated to the user.
3. First Name: the name of the user.
4. Surname: the surname of the user.
5. Email: email of the user.
6. Role: member role in the group.

 A screenshot of the IOT OSI-DEMO PLATFORM interface. The top navigation bar includes the logo, "IOT OSI-DEMO PLATFORM", and a user session. Below is a sidebar with icons for Home, Groups managed, Group members (selected), Devices, Topics, Measurements, Dashboards, and Digital twins. The main content area is titled "Platform assistant for groups admin". It shows a table of group members with columns: GroupId, UserId, First Name, Surname, Username, Email, and Role. The table has 3 rows of data. To the right of the table are edit and delete icons for each row. Red numbers 1 through 7 are overlaid on the table and its controls. Red arrows point from the numbers to specific elements: 1 points to the first column header "GroupId", 2 to the second column header "UserId", 3 to the third column header "First Name", 4 to the fourth column header "Surname", 5 to the fifth column header "Username", 6 to the sixth column header "Email", and 7 to the seventh column header "Role".

GroupId	UserId	First Name	Surname	Username	Email	Role
1	2	Daniel	Di Capua	dicapua	dicapua67@gmail.com	Admin
2	2	Daniel	Di Capua	dicapua	dicapua67@gmail.com	Admin
1	5	Rafael	Pacheco	rpacheco	rpacheco@gmail.com	Viewer

Figure 60: Parameter of Group members table.

1.4.9.3 Devices

The devices tab shows the list of devices that are inside a group.

DeviceId	OrgId	GroupId	Name	Description	Type	Icon	Longitude	Latitude	Mqtt acc	Device hash	Change hash	SSL certs
1	1	1	EEBE_GRAL_default	Default device of group EEBE_GRAL	Generic	radio	2.2225182346099926	41.41377997430348	Pub & Sub	pS9dM08qvdmYhbarDjMH		
2	1	2	ELAS_default	Default device of group Elasticity	Generic	radio	2.221995766428904	41.41345370658553	Pub & Sub	Kpz8WtYAV5LT312alqd		

Figure 61: Devices table

A user with group admin privileges can change the device hash, download the SSL certificates, create, edit or delete devices. When creating a device, the first field is to select to which group the device belongs to, the device name and description. Then type of device can be selected (Generic or Maser). The type of Mqtt access control can be selected and finally the device location and size.

Create device

GroupId
1

Device name
Device for machine 1

Description
Device for machine 1

Type
Generic

Mqtt access control
Subscribe & Publish

Device location and icon size

Longitude
2.222567237913609

Latitude
41.4137793990375

Cancel **Submit**

Create device

Description
Device for machine 1

Type
Generic

Mqtt access control
Subscribe & Publish

Device location and icon size

Longitude
2.222567237913609

Latitude
41.4137793990375

Icon ratio
1

Select location

Cancel **Submit**

Figure 62: Creation of a device.

Clicking in the Select location bottom the device location inside the group can be defined.

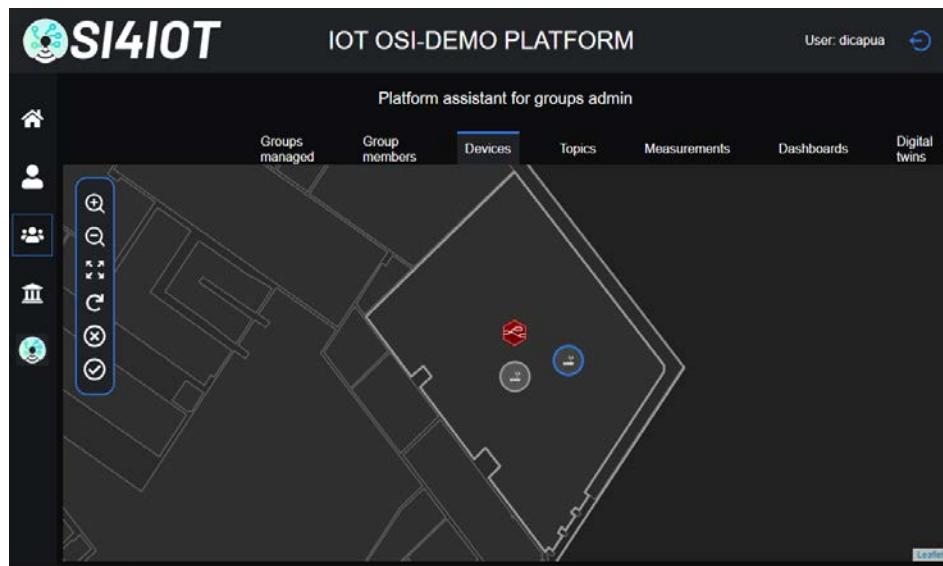


Figure 63: Defining the device location.

The parameters of Groups managed table are the following:

1. DeviceId: enumerator associated to the device listed.
2. OrgId: enumerator associated to the organization.
3. GroupId: enumerator associated to the group.
4. Name: the name of the device.
5. Description: the description of the device.
6. Type of the device: Generic;
7. Icon radio: radius in meters of the circle shown in the map.
8. Longitude: longitude where the device is placed.
9. Latitude: latitude where the device is placed.
10. Mqtt acc: type of Mqtt access control for the device.
11. Device hash: the hash code associated with the device.
12. Change hash: click this icon to change the device hash.
13. SSL certs: click to download the SSL certificate used to encrypt the message using the Mqtt protocol.

DeviceId	OrgId	GroupId	Name	Description	Type	Icon	Longitude	Latitude	Mqtt acc	Device hash	Change hash	SSL certs
1	1	1	EEBE_GRAL_default	Default device of group EEBE_GRAL	Generic	radio	2.2225182346099928	41.41377997430348	Pub & Sub	ps9dm08qvdyMhbarDjMH		
2	1	2	ELAS_default	Default device of group Elasticity	Generic	radio	2.221995766428904	41.413453706588553	Pub & Sub	kpz8WtYAV5LT312alqd		

Figure 64: Parameters of devices table.

1.4.9.4 Topics

The Topics tab show a list of the different Mqtt topics managed for the logged in user. A user with group admin privileges can create, edit or delete a topic and can change the topic hash.

TopicId	OrgId	GroupId	DeviceId	Type	Topic name	Description	Mqtt acc	Topic hash	Change hash
1	1	1	1	dev2pdb	EEBE_GRAL_Temperature	Temperature sensor for EEBE_GRAL_default device	Pub & Sub	NcY_XqkJv9bGDbw2xqu7	
2	1	1	1	dev2pdb_wt	EEBE_GRAL_Accelerometer	Mobile accelerations for EEBE_GRAL_default device	Pub & Sub	E5wgINpBucT_9Ycurl	
3	1	1	1	dev2dtm	EEBE_GRAL_Photo	Mobile photo for default for EEBE_GRAL_default device	Pub & Sub	TcmNkD9EMGFxa01arhox	
4	1	2	2	dev2pdb	ELAS_Temperature	Temperature sensor for ELAS_default device	Pub & Sub	KNWlg70Cs10Ver51kmO	
5	1	2	2	dev2pdb_wt	ELAS_Accelerometer	Mobile accelerations for ELAS_default device	Pub & Sub	mc2ZnkCtMKWeZ_mQXcc	
6	1	2	2	dev2dtm	ELAS_Photo	Mobile photo for ELAS_default device	Pub & Sub	pq5D8g2UHoMVLd5DROkT	

Figure 65: Topics table.

To create a new topic click the New Topic button and fill the fields. You need to define the group id that the device belongs to, its own id in the device id field, the type of topic, topic name and description. The Mqtt access control type also need to be defined. Finally, the payload format is a description in JSON format of the different parameters to be used as

payload of the topic. This description can include the name, the variable type and the units of the parameters.

The image displays two identical 'Create topic' dialog boxes side-by-side, both titled 'Create topic'. They are used for configuring a new MQTT topic, specifically for a device to platform DB.

Left Dialog Box:

- GroupId:** 1
- DeviceId:** 1
- Topic type:** Device to platform DB
- Topic name:** Temperature topic of device 1
- Description:** Temperature topic of device 1
- Mqtt access control:** Subscribe & Publish
- Payload format:**

```
{ "temperature": { "type": "number", "unit": "C" } }
```

Right Dialog Box:

- Device to platform DB**
- Topic name:** Temperature topic of device 1
- Description:** Temperature topic of device 1
- Mqtt access control:** Subscribe & Publish
- Payload format:**

```
{ "temperature": { "type": "number", "unit": "C" } }
```

Both dialogs feature 'Cancel' and 'Submit' buttons at the bottom.

Figure 66: Creation of a new topic.

If an already created device is edited the pop-out window shown in Figure 67.

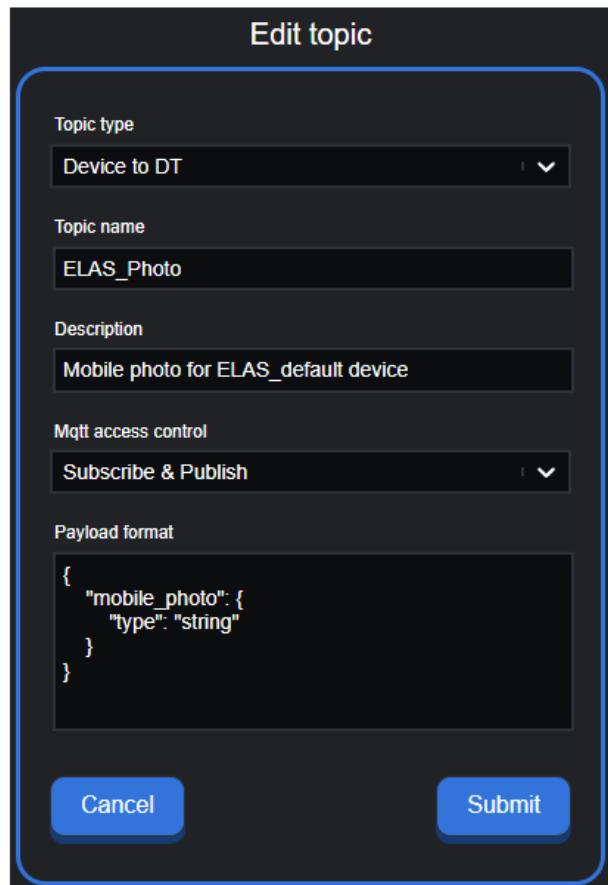


Figure 67: Edition of topic.

The parameters of the Topics table are the following:

1. TopicId: enumerator associated to the topic listed.
2. OrgId: enumerator associated to the organization.
3. GroupId: enumerator associated to the group.
4. DeviceId: enumerator associated to the device
5. Type: type of topic.
 1. dev2pdb: Device to platform database
 2. dev2pdb_wt: Device to platform database with timestamp
 3. dev2dtm: Device to Digital Twin Model (DTM).
 4. dev_sim_2dtm: Simulated device to DTM
 5. dtm_as2pdb: DTM assets state to platform database
 6. dtm_sim_as2dts: DTM simulated assets state to Digital Twin Simulator (DTS).
 7. dtm_fmv2pdb: DTM fem modal value to platform database.
 8. dtm_sim_fmv2dts: DTM simulated fem modal value to DTS.
6. Topic name: the label of the topic.
7. Description: the description of the topic.
8. Mqtt acc: type of Mqtt access control for the topic.
9. Topic hash: the hash code of the topic.
10. Change hash: click this icon to regenerate the topic hash.

TopicId	OrgId	GroupId	DeviceId	Type	Topic name	Description	Mqtt acc	Topic hash	Change hash
1	1	1	1	dev2pdb	EEBE_GRAL_Temperature	Temperature sensor for EEBE_GRAL_default device	Pub & Sub	Ncy_XqkJv9bGDbw2xqu7	<input type="checkbox"/> <input type="button" value="Edit topic"/> <input type="button" value="Delete topic"/>
2	1	1	1	dev2pdb_wl	EEBE_GRAL_Accelerometer	Mobile accelerations for EEBE_GRAL_default device	Pub & Sub	E5xgINOpBucT_9YcunL	<input type="checkbox"/> <input type="button" value="Edit topic"/> <input type="button" value="Delete topic"/>
3	1	1	1	dev2dtm	EEBE_GRAL_Photo	Mobile photo for default for EEBE_GRAL_default device	Pub & Sub	TCmNxD9EMGFxa01arhox	<input type="checkbox"/> <input type="button" value="Edit topic"/> <input type="button" value="Delete topic"/>

Figure 68: Topics table parameters.

1.4.9.4.1 Measurements

This tab shows a list of the sensor measurements for topics managed for the logged user.

OrgId	GroupId	DeviceId	Type	Topic name
1	1	1	dev2pdb	EEBE_GRAL_Temperature

Timestamp Payload

2022-12-22 08:58:17.420808+00	{"temperature":24}	<input type="checkbox"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/>
2022-12-22 08:58:16.447042+00	{"temperature":24}	<input type="checkbox"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/>
2022-12-22 08:58:15.259337+00	{"temperature":24}	<input type="checkbox"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/>
2022-12-22 08:58:14.222078+00	{"temperature":24}	<input type="checkbox"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/>
2022-12-22 08:58:10.287803+00	{"temperature":23}	<input type="checkbox"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/>
2022-12-22 08:58:09.398884+00	{"temperature":23}	<input type="checkbox"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/>
2022-12-22 08:58:08.711561+00	{"temperature":23}	<input type="checkbox"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/>
2022-12-22 08:58:02.272060+00	{"temperature":25}	<input type="checkbox"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/>

Figure 69: Measurements table.

When a topic and time range are selected, it is shown the history of the corresponding measurements with its payload and the timestamp.

1.4.9.5 Dashboards

This tab shows a list of the dashboards created in Grafana application that can be managed for the logged user.

DashboardId	OrgId	GroupId	Slug	Title	Uid
2	1	1	home_eebe	Home EEBE	2d89c6cb-f7f6-4e16-8917-ddef88cd3515
3	1	1	eebe_gral_temp_demo	EEBE_GRAL_Temp_demo	e3c79d41-32c8-4100-8748-97bcf7c5a926
4	1	1	eebe_gral_accel_demo	EEBE_GRAL_Accel_demo	bec50bba-0a59-4c2a-8d3f-cc9941229a93
6	1	2	elas_temp_demo	ELAS_Temp_demo	c0981c7d-6721-4e89-9c4a-22889432014b
7	1	2	elas_accel_demo	ELAS_Accel_demo	76b2ea23-93f2-4461-af03-5df35dc80738

Figure 70: Dashboards table.

The parameters of the Dashboards table are the following:

1. DashboardId: enumerator associated to the dashboard listed
2. OrgId: enumerator associated to the organization.
3. GroupId: enumerator associated to the group.
4. Slug: the label of the dashboard.
5. Title: the name of the dashboard.
6. Uid: Dashboard universal identification.

1	2	3	4	5	6
DashboardId	OrgId	GroupId	Slug	Title	Uid
1	1	1	home_eebe	Home EEBE	2d89c6cb-f7f6-4e16-8917-ddef8cd3515
2	1	1	eebe_gral_temp_demo	EEBE_GRAL_Temp_demo	e3c79d41-32c8-4100-8748-97bcf7c5a926
3	1	1	eebe_gral_accel_demo	EEBE_GRAL_Accel_demo	b6c50bba-0a59-4c2a-8d3f-cc9941229a93
4	1	2	elas_temp_demo	ELAS_Temp_demo	c0981c7d-6721-4e89-9c4a-22889432014b
5	1	2	elas_accel_demo	ELAS_Accel_demo	76b2ea23-93f2-4461-af03-5df35dc80738
6	1	2	elas_accel_demo	ELAS_Accel_demo	76b2ea23-93f2-4461-af03-5df35dc80738

Figure 71: Parameters of Dashboards table.

1.4.9.6 Digital Twins

This tab shows a list of the digital twin models managed by the logged user. A user with group admin privileges is able to create, edit or delete digital twin models.

Id	OrgId	GroupId	DeviceId	Reference	Description	Type
1	1	1	1	DT_KRXl0o4Bv3aX7UsSbAwS	Temperature dashboard for EEBE_GRAL group	Grafana dashboard
2	1	1	1	DT_PS66KFE87e7woNcRMS06	Accelerations dashboard for EEBE_GRAL group	Grafana dashboard
3	1	2	2	DT_ycMVJyyczsyZalfAxk6	Temperature dashboard for ELAS group	Grafana dashboard
4	1	2	2	DT_l8ICBy6VfBDEJUs8gysE	Accelerations dashboard for ELAS group	Grafana dashboard

Figure 72: Digital twins table.

To create a digital twin model click the New digital twin button. Then appear pop-out window shown in Figure 73. The GroupId and Deviceld need to be indicated. The DigitalTwinUid is a reference of the digital twin model. This reference is used in Nodered instances. The other two fields are the description and the type of the digital twin model. The type can be either Grafana dashboard or Gltf 3D model.

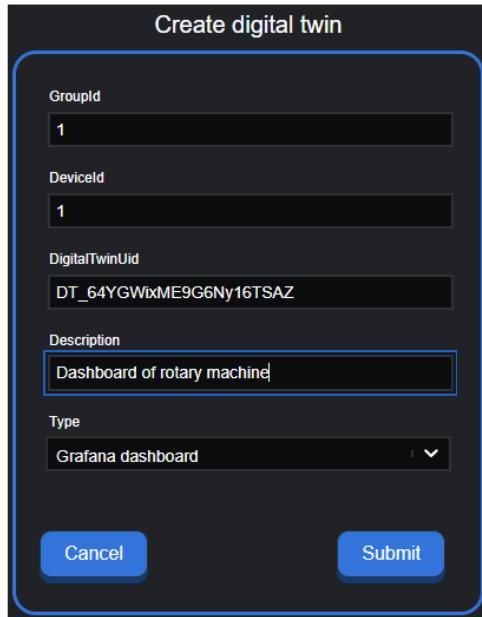


Figure 73: Creation of a new digital twin with Grafana dashboard type.

When Gltf 3D model is selected as type, the menu window changes in appearance as is shown in Figure 74.

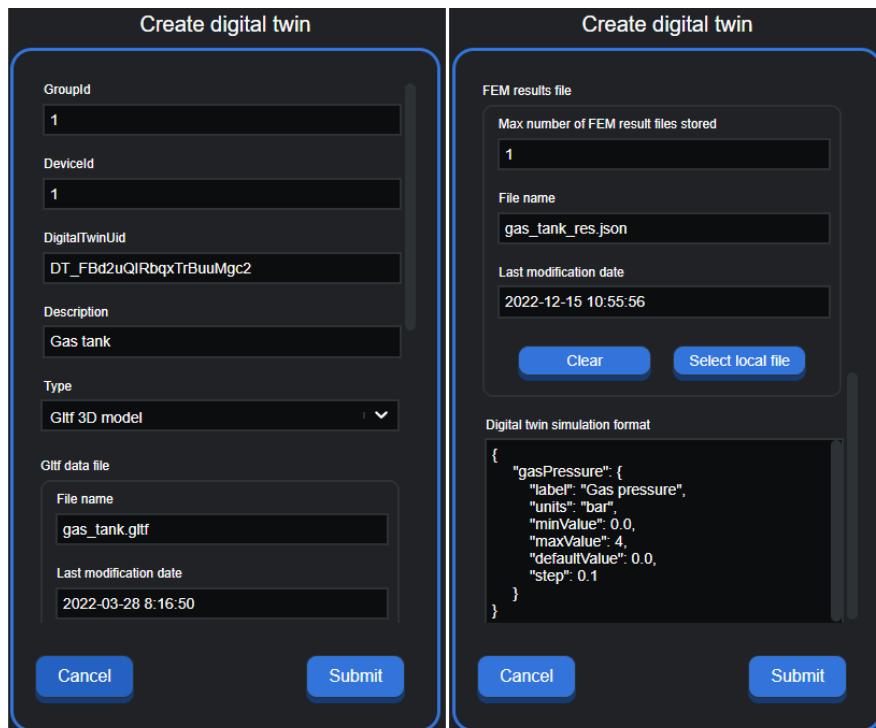


Figure 74: Creation of a new digital twin with Gltf 3D model type.

The geometrical definition of 3D model is defined in a gltf file that need to be uploaded. The other part of the digital twin model is the FEM simulation data file. You need to upload a valid JSON file, when uploaded, the fields file name and last modification date will be populated automatically. The last field to fill is the Digital twin simulation format.

The parameters of the Digital twin table are the following:

1. Id: enumerator associated to the digital twin model listed.
2. OrgId: enumerator associated to the group.
3. DeviceId: enumerator associated to the device.
4. Reference: the label of the digital twin model.
5. Description: the description of the digital twin model.
6. Type: the type of digital twin model.

1	2	3	4	5	6	7
<input type="text"/>	<input type="text"/>	<input type="text"/>				
1	1	1	1	DT_KRX10o4Bv3aX7UsSbAwS	Temperature dashboard for EEBE_GRAL group	Grafana dashboard
2	1	1	1	DT_PS66KFE87e7woNcRMS06	Accelerations dashboard for EEBE_GRAL group	Grafana dashboard
3	1	2	2	DT_ycMVJyyczsyZalfAxk6	Temperature dashboard for ELAS group	Grafana dashboard
4	1	2	2	DT_l8ICBy6VfBDEJUs8gysE	Accelerations dashboard for ELAS group	Grafana dashboard

Figure 75: Parameters of Digital twins table.

1.4.10 User role

The User role is the lowest role available and it can be accessed when clicking on the corresponding icon of the Role Navigation Bar. In the Use role can found the following tabs:

- User profile
- Membership in orgs
- Membership in groups

The default page for the User role is shown in Figure 76.

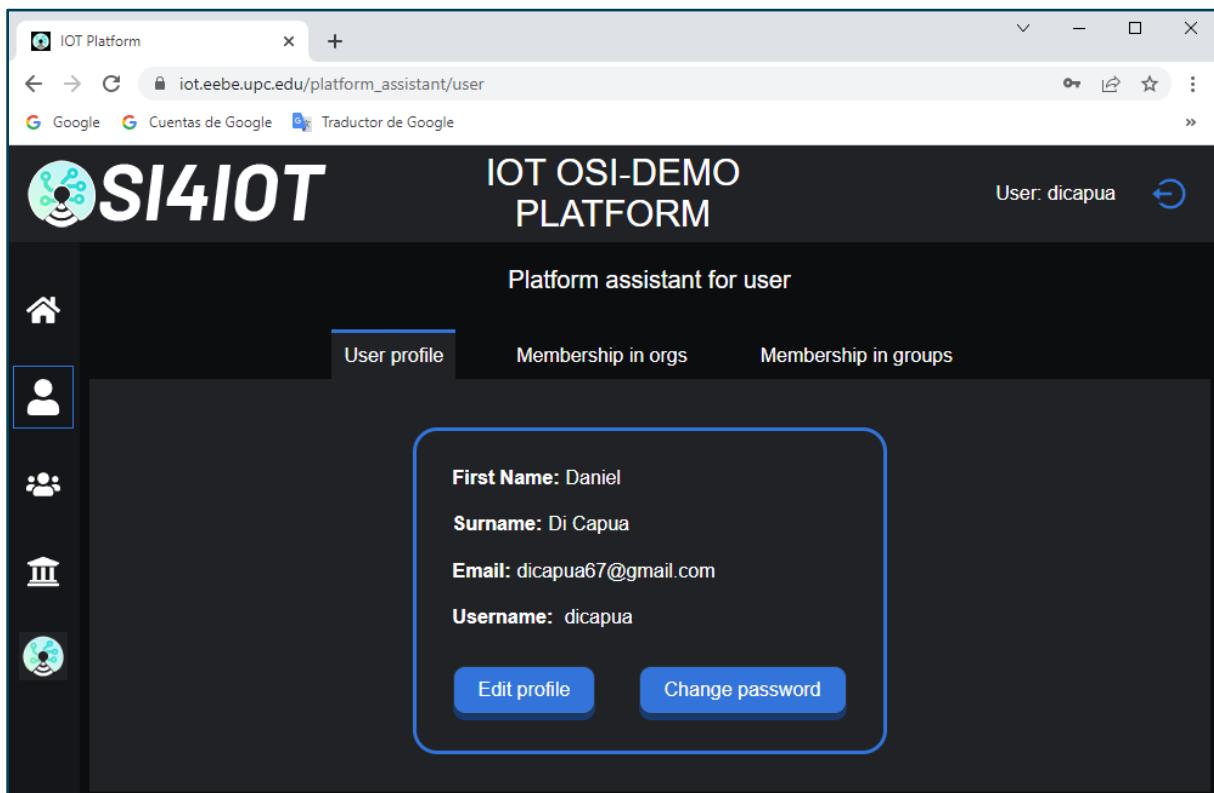


Figure 76: Default page of User role.

1.4.10.1 User Profile

This tab displays a summary of the identification data of the logged user. The fields displayed in this tab can be edited with the Edit profile button. The user password can be modified by clicking the Change password button.

1.4.10.2 Memberships in Orgs

The Membership in Orgs tab displays a list of organizations that the logged in user is a member of.

The screenshot shows a web browser window titled 'IOT Platform' with the URL 'iot.eebe.upc.edu/platform_assistant/user'. The main title is 'IOT OSI-DEMO PLATFORM'. On the left, there is a sidebar with icons for Home, User profile, Groups, and Help. The 'User profile' tab is active. The main content area is titled 'Platform assistant for user' and shows a table titled 'Membership in orgs'. The table has columns: OrgId, Name, Acronym, Address, City, Zip code, State, Country, and Role. A single row is displayed: OrgId 1, Name Escola d'Enginyeria de Barcelona Est, Acronym EEBE, Address Av. d'Eduard Maristany, 16, City Barcelona, Zip code 08019, State Catalunya, Country Spain, and Role Admin. There are search fields for OrgId, Name, and Acronym, and a global search bar at the top of the table.

Figure 77: Memberships in orgs table.

The parameters of the Membership in orgs table are the following:

1. OrgId: enumerator associated to the group.
2. Name: name of the organization.
3. Acronym: acronym of the organization.
4. Address: address of the location of the organization.
5. City: city of the organization.
6. Zip code: postal code of the location of the organization.
7. State: state or province where the organization is located.
8. Country: country where the organization is located.
9. Role: role that the logged user has inside the organization.

This screenshot is identical to Figure 77, but the columns of the 'Membership in orgs' table are numbered 1 through 9 above the header. The first column, 'OrgId', is highlighted with a red border. The data in the table remains the same as in Figure 77.

Figure 78: Memberships in orgs table.

1.4.10.3 Memberships in Groups

The membership in groups tab shows a list of that that the logged in user is a member of.

OrgId	GroupId	Name	Acronym	Telegram invitation link	ChatId	Role
1	1	Escola d'Enginyeria de Barcelona Est general	EEBE_GRAL	https://t.me/joinchat/IAbDEmLxwDMmT7_	-537434892	Admin
1	2	Elasticity	ELAS	https://t.me/+d-potD1860ZRTV38	-597813945	Admin

Figure 79: Memberships in groups table.

The parameters of the Membership in groups table are the following:

1. OrgId: enumerator associated to the group.
2. Group Id: enumerator associated to the group.
3. Name: name of the group.
4. Acronym: acronym of the group.
5. Telegram invitation link: the invitation link to the telegram group where the notification bot is.
6. ChatId: the id of the chat group of telegram.
7. Role: the role of the user in the group.

1	2	3	4	5	6	7
OrgId	GroupId	Name	Acronym	Telegram invitation link	ChatId	Role
1	1	Escola d'Enginyeria de Barcelona Est general	EEBE_GRAL	https://t.me/joinchat/IAbDEmLxwDMmT7_	-537434892	Admin
1	2	Elasticity	ELAS	https://t.me/+d-potD1860ZRTV38	-597813945	Admin

Figure 92: Parameter of Memberships in groups table.

2 References

- [1] Eclipse Mosquitto. [Online]: <https://mosquitto.org/>
- [2] Mosquitto Go Auth. [O
- [3] Node-RED. [Online]: [4] PostgreSQL. [Online]:
- [5] Timescaledb. [Online]: [6] Pgadmin. [Online]: [7] Grafana. [Online]: [8] Grafana Image Renderer. [Online]: [9] Traefik. [Online]: [10] Minio. [Online]: [11] Portainer. [Online]:
- [12] Keepalived. [Online]: [13] Docker. [Online]:[14] Blender. [Online]: