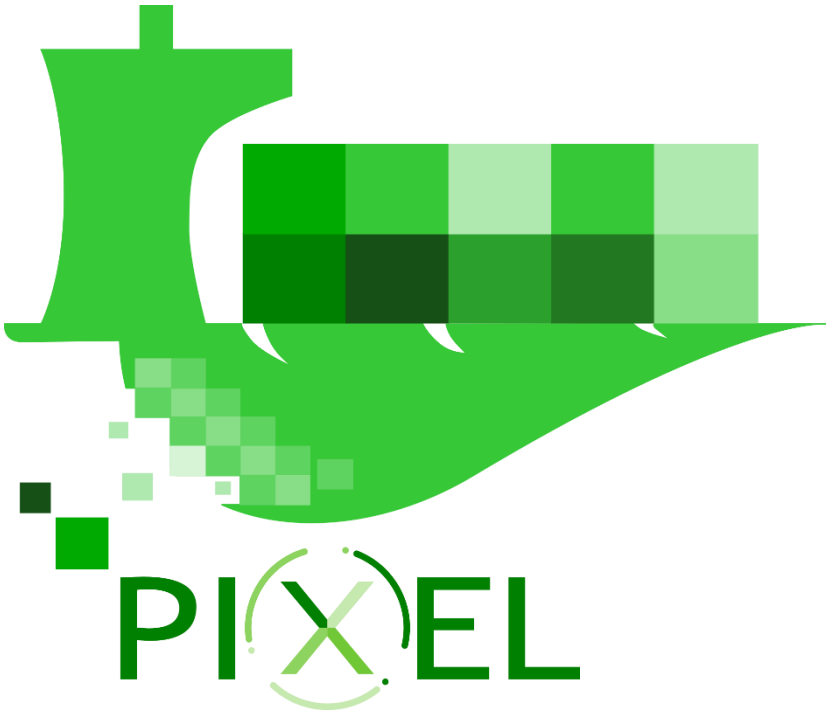
This project has received funding from the European's Union Horizon 2020 research innovation program under grant agreement No. 769355



D7.3- Pilots and Cross Pilot Collaboration Report

Deliverable No.	D.7.3	**Due Date**	30- JUN-2021 (version sent 6-JUL-2021, incurring in a 6 days' delay).
Type	Report	**Dissemination Level**	Public
Version	1.0	**Status**	Release 1
Description	This document reports the work conducted in the inter-pilot and collaboration task (T7.6). It includes new use cases tested in each pilot site, describes the main generalization aspects of the PIXEL Platform and models and lessons learnt regarding the PIXEL framework, in the four use-cases, have been included.		
Work Package	**WP7**		

Authors

Name	Partner	e-mail
Ignacio Lacalle	1- UPV	iglaub@upv.es
Benjamin Molina	1- UPV	benmomo@upvnet.upv.es
Ismael Torres	2- PRO	itorres@prodevelop.es
Miguel Ángel Llorente	2- PRO	mllorente@prodevelop.es
Gilda De Marco	4- INSIEL	gilda.demarco@insiel.it
Paolo Casoto	4- INSIEL	paolo.casoto@gmail.com
Charles Garnier	5- CATIE	c.garnier@catie.fr
Erwan Simon	5- CATIE	e.simon@catie.fr
Marc Despland	6-ORANGE	marc.despland@orange.com
Teodora Milosevic	8-MEDRI	teodora.milosevic@medri.uniri.hr
Stjepan Piličić	8 MEDRI	stjepan.pilicic@gmail.com
Tamara Corsano	9- SDAG	t.corsano@sdag.it
Eleonora Anut	9- SDAG	e.anut@sdag.it
Cinzia Ninzatti	9- SDAG	c.ninzatti@sdag.it
Eirini Tserga	10- ThPA S.A.	etserga@thpa.gr
Grigoris Dimitriadis	10- ThPA S.A.	gdimitriadis@thpa.gr
Dimitrios Spyrou	11- PPA S.A.	dspyrou@olp.gr
Athanasios Chaldeakis	11- PPA S.A.	ahaldek@gmail.com
Stefano Bevilacqua	12-APT	stefano.bevilacqua@porto.trieste.it
Fabrice Klein	13-GPMB	F-Klein@bordeaux-port.fr
George Litainas	14-PEOPLE	glitainas@people-t.com
Leonidas Pitsikas	14-PEOPLE	lpitsikas@peoplethinkbeyond.com

History

Date	Version	Change
01 – MAR – 2021	0.1	ToC v1
01 – APR – 2021	0.2	ToC Final
20 – MAY – 2021	0.3	Task assignment
25 – JUN – 2021	0.4	Initial version with all contributions
01 – JUL – 2021	0.5	Integrated version
6 – JUL – 2021	1.0	Submission to EC after a review of the document by the TC and PC

Key Data

Keywords	Use Case, Cross pilot, Lessons learnt
Lead Editor	Ismael Torres - P02 PRO
Internal Reviewer(s)	CERT, CREO

Abstract

The deliverable D7.3 will present the work conducted in the inter-pilot and collaboration task. It explains new use cases tested that were not originally planned. One of the models not planned is the COVID model. This is a model that the project consortium decided to develop at the beginning of the pandemic to try to help ports control employee distancing measures and avoid crowding. The model arises to help ports to control the number of people who are working in the different areas of the port.

Throughout the project it has always been kept in mind that the results of the project should be easily applicable in other ports, considering this requirement. In this document, it is explained how the platform and models have been designed bearing in mind that they should be sufficiently generic to be easily applicable to other ports.

Finally , after the development and execution of the different pilots carried out throughout the WP7, it is very interesting to gather the opinion of the different users in relation to the usefulness of the platform and the models. This information is very useful to know the strengths and weaknesses of the platform directly from its users, both from the technical and from the port user's point of view. It can be used to improve the platform for the aim of developing new pilots or proof of concept in ports interested in monitoring their environmental impact through tools such as the one proposed in the PIXEL project.

Statement of originality

Table of contents

List of tables

List of figures

List of acronyms

Acronym	Explanation
API	Application Programming Interface
ASPM	CCIAA DI GORIZIA - AZIENDA SPECIALE PER IL PORTO DI MONFALCONE
BTEX	Benzene, Toluene, Ethylbenzene and Xylene
CO	Carbon Monoxide
DAL	Data Acquisition Layer
D&N	Dashboard and Notification
EE	Electric energy
eKPI	Environmental Key Performance Indicator
GPMB	Grand Port Maritime de Bordeaux - Port of Bordeaux
GUI	Graphical User Interface
IH	Information Hub
IoT	Internet of Things
IPD	Individual Protection Device
IS	Information System
JSON	JavaScript Object Notation
KPI	Key Performance Indicators
LNG	Liquified Natural GAS
NGSI	Next Generation Service. Interface
NOx	Nitrogen Oxides
O3	Ozone
OT	Operational Tools
PAS	Port Activity Scenario
PEI	Port Environmental Index
PIXEL	Port IoT for Environmental Leverage
PM10	Particulate Matter 10 micrometres or less in diameter
PV	Photovoltaic
TDS	Total Dissolved Solids
ThPA	Thessaloniki Port Authority
SO2	Sulfur Dioxide
SDAG	Stazioni Doganali Autoportuali Gorizia S.p.A.

1. About this document

The scope of the deliverable is to report about the work conducted in the inter-pilot and collaboration task. This task was devoted to enable the interaction, communication, and alignment between the PIXEL pilots. The main activities conducted in this task were focused in: (i) new use cases tested in each pilot site, leveraging the PIXEL Information System technology and enabling a better environmental management thanks to the experience and knowledge of the collaborating ports and (ii) enable the generalization of the PIXEL Platform and models to easily apply the PIXEL solution in other small and medium European ports. Moreover, lessons learnt regarding the PIXEL framework, in the four use-cases, have been included.

1.1. Deliverable context

Table 1: Deliverable content

Keywords	**Lead Editor**
Objectives	Objective 1: Describe new use cases tested that were not originally planned.
	Objective 2: Explain how the platform and models have been designed to meet the requirement of genericity and portability.
	Objective 3: Lessons learnt during the execution of the pilots
Exploitable Results	This deliverable presents the PIXEL platform and de models that are the main outcomes of the PIXEL project.
Work plan	This deliverable reports about the works performed mainly in the task 7.6 and task T7.7, without forgetting the other tasks in the WP7 to complete the lessons learnt section
Milestones	MS9
Deliverables	Detected inputs: ● D7.1 is the first version of the integration report. ● D7.2 is the second version of the integration report. Detected outputs: ● D8.3. The pilots developed in wp7 will be evaluated in WP8.
Risks	**Risk N°18** - Delay in technical developments. A substantial delay in the completion of the technical developments of the platform and models implies a delay in the validation of the developments and in the development of the different pilot. *Mitigation Measures*: Firstly, conducting weekly meetings to have a better control of the state of the developments and minimize time deviations. Secondly, making incremental deliveries of software artifacts that allow pilots to advance without waiting for the latest version.

	Risk Nº19 - Technical documentation delayed or incomplete. A delay in documentation or poor-quality documentation will force both technical partners and ports to devote more effort to the development of pilots. *Mitigation Measures*: Involve the technical leaders of the pilots in the technical follow-up meetings so that they know first-hand the state of the developments.
	Risk 20 - Delay in the availability of the infrastructure in the pilots. If the infrastructure is not on time, the start of the technical part of the pilot will be delayed and the data and models cannot start to be integrated. *Mitigation Measures*: be flexible in the minimum requirements so that they have no problems in providing the minimum infrastructure.
	Risk 22 - insufficient or untimely pilot data available. The data required to develop the scenarios for the different ports are not on time or the quality of the data is not sufficient. *Mitigation Measures*: work with the ports in the definition of data sources and in the acquisition of the sensors, months before the beginning of the implementation of the pilots. Moreover, it could be possible to test the scenarios with simulated data or look for alternative data sources.

1.2. The rationale behind the structure

The deliverable has been structured in several sections.

- "About this document" and "Motivation" that clarifies the objectives and the motivation for writing this deliverable.
- COVID Pilot. This section describes a new model that can help ports in complying with the social distancing measures applied to minimise virus' spread, looking after the safety of workers by monitoring the number of people in different areas of the port.
- Cross-pilot report: introduction of models from other use-cases. This section describes the models applied in the ports that were not initially considered and that endorse the flexibility and scalability of PIXEL.
- Generalization. This section describes all the decisions made and strategies adopted during the design and implementation of the PIXEL platform in the pilots so that models and scenarios will be easily portable/applicable to other ports.
- Lessons learnt. This section describes the lessons learnt during the execution of the pilots from the technical and end user perspective.

Finally, the conclusion section summarises the deliverable with the most important conclusions and objectives achieved by the end of WP7.

1.3. Version-specific notes

The D7.3. presents the work conducted in the inter-pilot and collaboration task. It explains new use cases tested that were not originally planned and how the platform and models have been designed bearing in mind the "Generalization" non-functional requirement. Finally, the lessons learnt during the execution of the pilots have been included.

2. Motivation

During the proposal stage of PIXEL back in 2017, a set of objectives related to the platform and the elaboration of the different pilots were defined. These objectives were later detailed during the execution of the WP3 "Requirements and use cases" which redounded on a list of functional and non-functional requirements for both the platform and the different technical elements as well as for the different pilots. This document explains the work carried out in the different pilots that were not initially planned.

During the design of the platform and the integration of the different models and data sources, it has always been kept in mind that the solution should be generic enough to be easily applicable to different pilots. The aim is that once the PIXEL project is completed, it will be easy to implement Pixel in ports interested in measuring the impact. It is important that the document includes all the activities carried out at the technical level to facilitate the use of PIXEL once the project is completed. It is important that these activities do not only focus on the platform but should also include integration aspects, without forgetting the effort carried out in various models such as the PEI and PAS, which are the models that have been used in all the pilots and which will most likely be used in future pilots.

Other of the points included in the document is the lessons learnt from the development of the pilots. To this end, it has been considered interesting to compile this information both from the technical point of view and from the point of view of the end users of the port. This information will be very useful to obtain first-hand feedback and impressions from the pilots.

3. COVID Pilot

3.1. Introduction / motivation

The COVID-19 outbreak that started engulfing various nations across the globe forced governments, national and international authorities to take unprecedented measures such as lockdowns and restricting the movement of people (and workers) to check and control the exponential spread of the pandemic. The freight and logistics sector have been on the frontline since the very beginning of the COVID-19 crisis.

Most countries restricted or stopped international flights and the shipping sector has also been hit as vessels were placed under quarantine for weeks before being allowed into the ports. Shipping containers and trucks were stuck at the ports and at state borders for checks. The first- and last-mile transportation and intermodal connectivity of goods came to a standstill especially during the lockdown, causing overcrowding of people who were waiting. Increased border controls and customs regulations resulted in longer waiting times, and the lack of capacity for long-haul and last mile fulfilment created extreme challenges for freight transport and especially for road transport.

Ports and inland terminals are essential nodes in the multi-modal supply chain, and COVID-19 crisis has significantly impacted on them also due to the national and international dispositions that forced social distances among people and so, also workers (for example limiting the density both indoor and outdoor), and nevertheless also many sanitary obligations which inevitably impact on the safety and security of workers.

Therefore, the conditions (both sanitary and regarding safety and security) are fundamental. Truck drivers, and all workers, need to be able to wait in a safe and secure environment that also provides at least minimum levels of service and a sufficient level of security and safety of the area (as a matter of fact, in most of the cases the access to toilets was even forbidden).

This is also relevant according to the new European regulation on the Mobility Package[1] also referring to safe and secure truck parking areas, in which drivers' conditions are the starting basis. Being witness of overcrowded parking areas, where trucks were parked, and people were waiting on each corner was unfortunately common during COVID-19 crisis. It is not only extremely dangerous in terms of road safety, but also highly uncomfortable and worrying for workers' health.

All of those problems lead to the objective of the COVID Pilot in PIXEL project, which is to leverage the development of an IoT-based platform (PIXEL's overarching goal) that, drawing from the previous, could help in evaluating how the COVID-19 social distancing measures can be introduced in the scheduling of port operations avoiding overcrowding of workers and truck drivers in one area of the terminal during a particular set of time, including also impacts in productivity and energy efficiency the measures can have. Therefore, helping port managers know how and when there would be a risk of violating the social distance restrictions were identified as an interesting feature that could drive a terminal operator and to take proper measures consequently to address density of workers in inland terminal areas such as SDAG. This goal could then also be applied to a generic scenario for the benefit of the community. However, the scope of this pilot has been just to provide a simulation (decision support tool) based on the real data of incoming vessels and expected operations in the terminal.

Finally, because of these preliminary remarks, the Port of Monfalcone together with SDAG, as an inland terminal, was interested on evaluating how the COVID-19 social distancing measures can be introduced in the scheduling of port operations and realise which impacts in terms of productivity and energy efficiency would those measures have.

In terms of the actual execution of this pilot in the work plan, it is worth mentioning that it was not planned in the original Description of Work. The addition of the task associated with this pilot (T7.7) came into force after the signature of the second amendment of the project.

[1] Regulation (EU) 2020/1054 of the European Parliament and of the Council of 15 July 2020 amending Regulation (EC) No 561/2006 as regards minimum requirements on maximum daily and weekly driving times, minimum breaks and daily and weekly rest periods and Regulation (EU) No 165/2014 as regards positioning by means of tachographs.

3.2. Description

3.2.1. Technical Approach

The idea behind the technical approach for this pilot was the following:

- To take advantage of the already developed components/models of PIXEL to deliver the results. This fact would allow to (i) maximise effort efficiency and (ii) demonstrate the scalability, adaptability, modularity, flexibility and usefulness of PIXEL architecture and models;
- To make the global system as generic as possible so that it can be adapted to other ports (see task T7.6 and application to Thessaloniki).

The different actions that have been performed at the technical level are:

- PAS instantiation and fine-tuning: a new instance of the PAS with the purpose of just implementing this use-case has been used. The work on the PAS model (deployment, integration, etc.) in the normal PIXEL flow has not been affected. The work in this new instance has consisted in associating a new Resource in the definition of the Areas, Supply Chain and Machinery while modeling the Port of Monfalcone. The introduction of new rules such as limitation of people and prioritisation of type of vessels have also been part of the fine-tuning;
- Improvement of the vessel calls retrieval agent: the NGSI agent that was developed for inserting vessel calls of the Port of Monfalcone into PIXEL's Information Hub was enhanced to include a series of additional parameters/features to comply with the requirements of this pilot:
    - Separation of the registers of vessels in separated registries in case they are occupying more than one berth;
    - Reinforcement of the berth information, enriching it;
    - Fine-tuning of several fields including loading tonnage, unloading tonnage and inclusion of advanced filtering specially on the goods' type category.
- PAS fulfilment: port of Monfalcone performed the fulfilment of the PAS forms (enhanced and tailored for this pilot) to define their supply chain considering the new constraints;
- Modelling: The execution of the PAS provides the same usual results: timelines of steps, machinery usage, area occupancy, supply chain duration and energy consumption per process, all of that referred per vessel. For this COVID pilot, the human has been included as output parameter:
    - A new doc_id is generated within the index pas-outputs-results in the Information Hub to deliver the results associated with the COVID pilot (occupancy, density, limit of workers).
    - A parallel index pas-output-results-covid is generated after each execution of the PAS that includes a scenario which assumes that no area in the port could exceed at any given moment the maximum limitation of workers per square meter. The result, then, is different from what a usual simulation of the PAS would provide (as the latter could result in higher, prohibited occupancy rates).
- Visualisation: a new visualisation has been created: a map of the port with geo-fenced areas informing about density of workers and energy consumption in those areas each timeframe (per hour). This visualisation must allow the user to advance in time to check the evolution of those parameters during the shifts. The visualisation allows to see the results aggregated, per vessel and with/without pandemic constraints. The visualization was decided to be included in the platform as a separate entry in the menu of PIXEL for the sake of clarity and separation of roles. This tool would be useful for the terminal operator.

3.2.2. Implementation

The first action performed in the task was to design a Minimum Viable Product (MVP) for the model so that it could be validated in an agile, straightforward way before proceeding to a final integration in the PIXEL deployment of the Port of Monfalcone (hosted by INSIEL).

The architecture of this MVP relied as much as possible on PIXEL components to guarantee a smoother integration in the final platform. In the image below the infrastructure and architecture used for such MVP is shown:

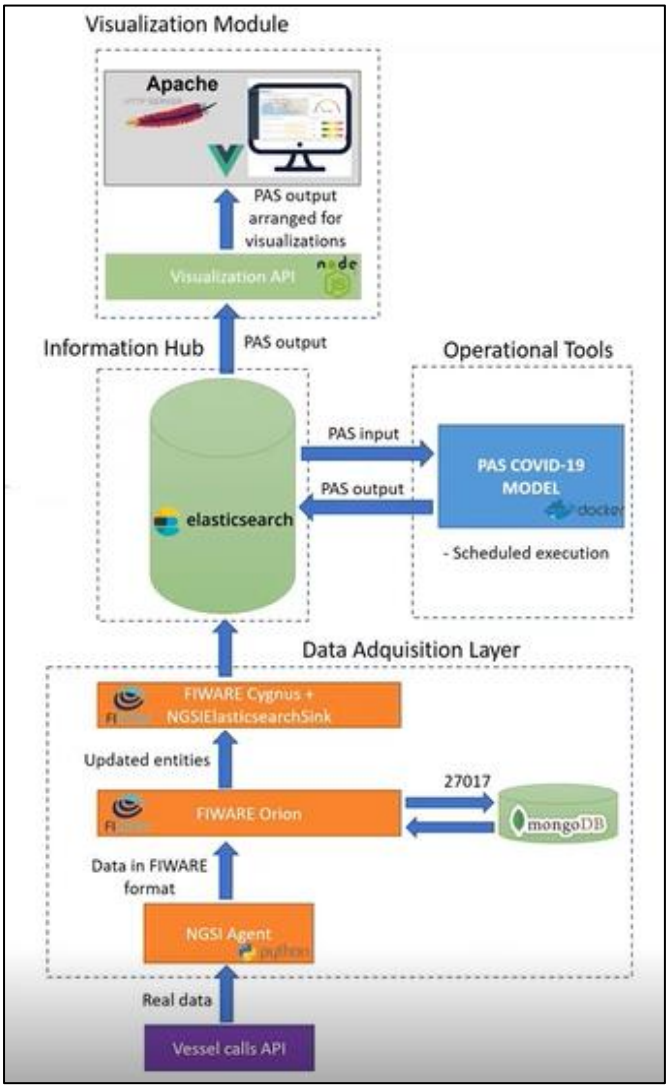


Figure 1: PIXEL COVID architecture

Once this was validated, the MVP was integrated in PIXEL, becoming a new model available for Port of Monfalcone stakeholders, accessible through the Dashboard of the already installed system.

The overall flow (both for the MVP and for the final integration) is related hereafter.

The first work consisted of associating a new resource in the definition of the areas, supply chain, etc. while modeling the PAS forms (already fulfilled) of the Port of Monfalcone to come up with a simulation of activities of the terminal. Furthermore, also the introduction of new rules such as limitation of people and prioritisation of type of vessels, the type of operative activities and occupancy of terminal areas was also part of the fine-tuning. So, allowing an analysis of the situation in a certain period (per hour) and therefore being able to take measures in order to guarantee a certain level of safety and security to all workers in terms of density of people per area.

Doing so, the PAS allows to define a maximum number of workers per area, process and machine used in the terminal (*see the image below).*

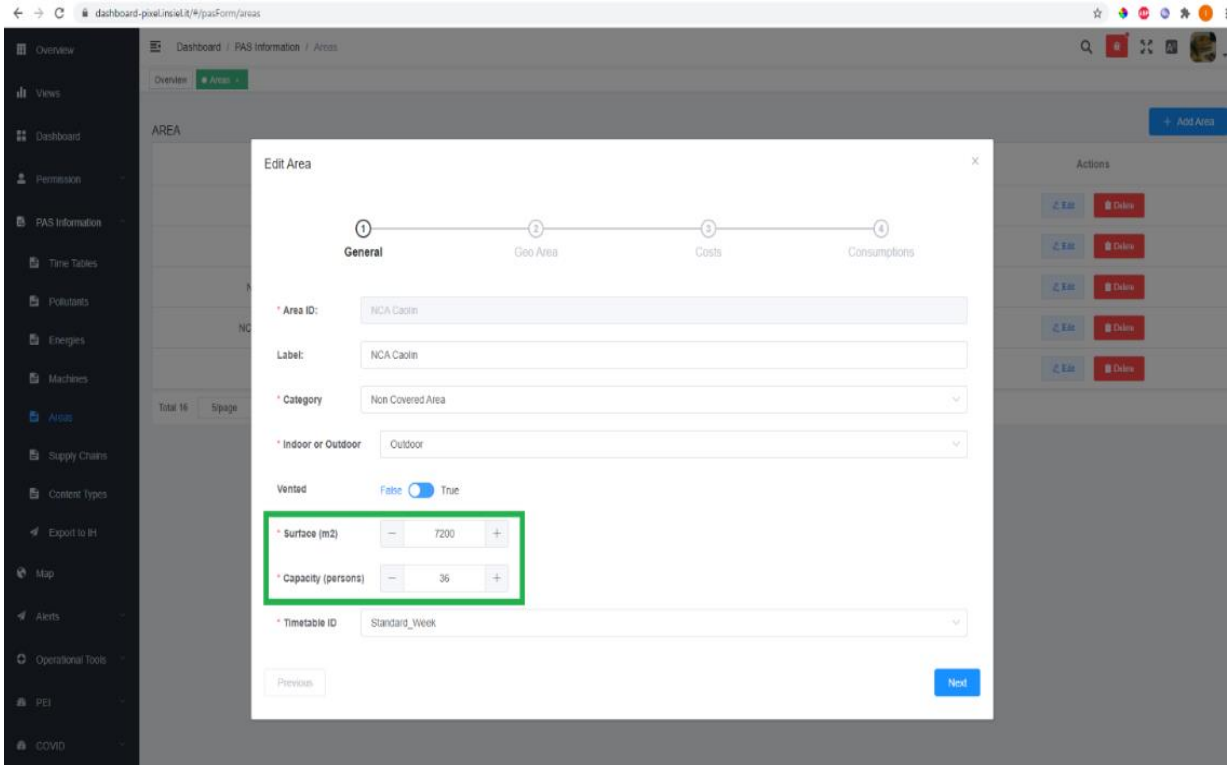


Figure 2: Definition of maximum number of workers per area

The figure below shows the visualisation of the model execution:

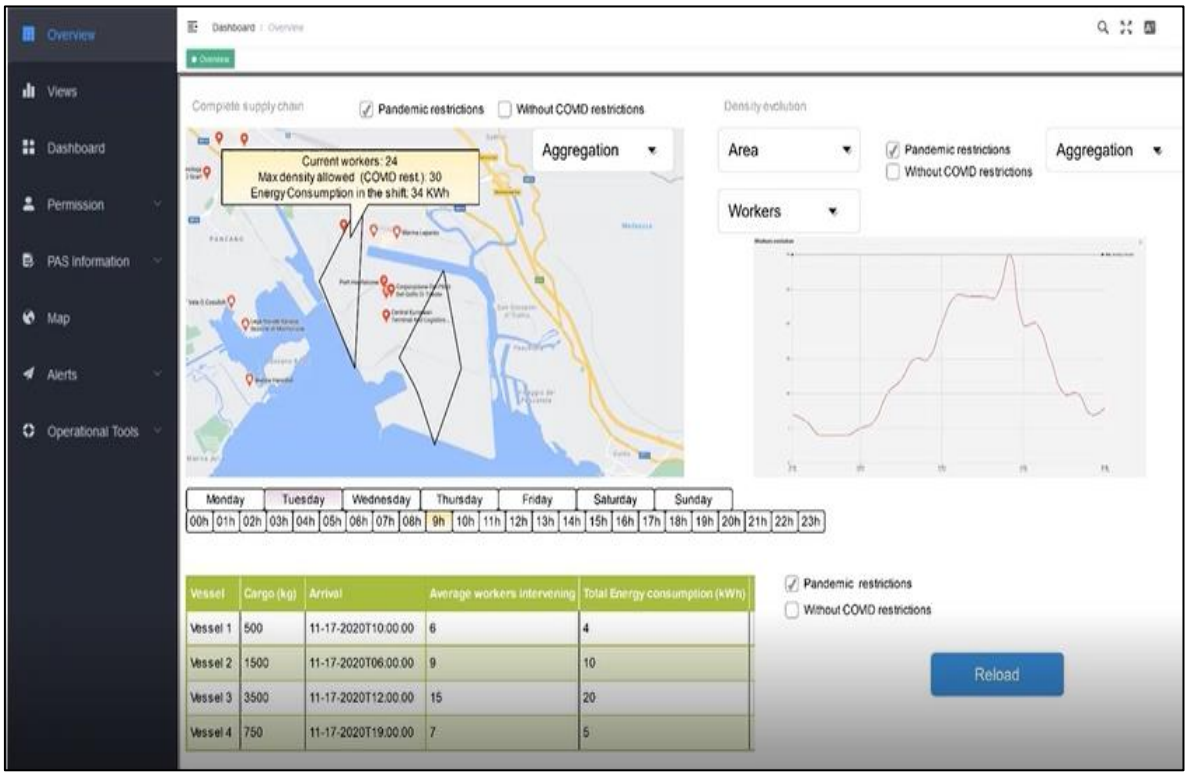


Figure 3: Mock-up of the visualisation associated to the COVID pilot.

Due to the fact of hugely relying on already-developed PIXEL artifacts, the implementation flow has been like the rest of pilot cases exposed in this deliverable.

The first step is the acquisition of data sources from the vessel calls API from the Port of Monfalcone (including different types of vessels coming to the Port, for example slabs and cars that are parameterised inside the PAS). This step is done by retrieving the information from the automated service of "sailing list" on the website of the Port of Monfalcone. An example can be seen below.

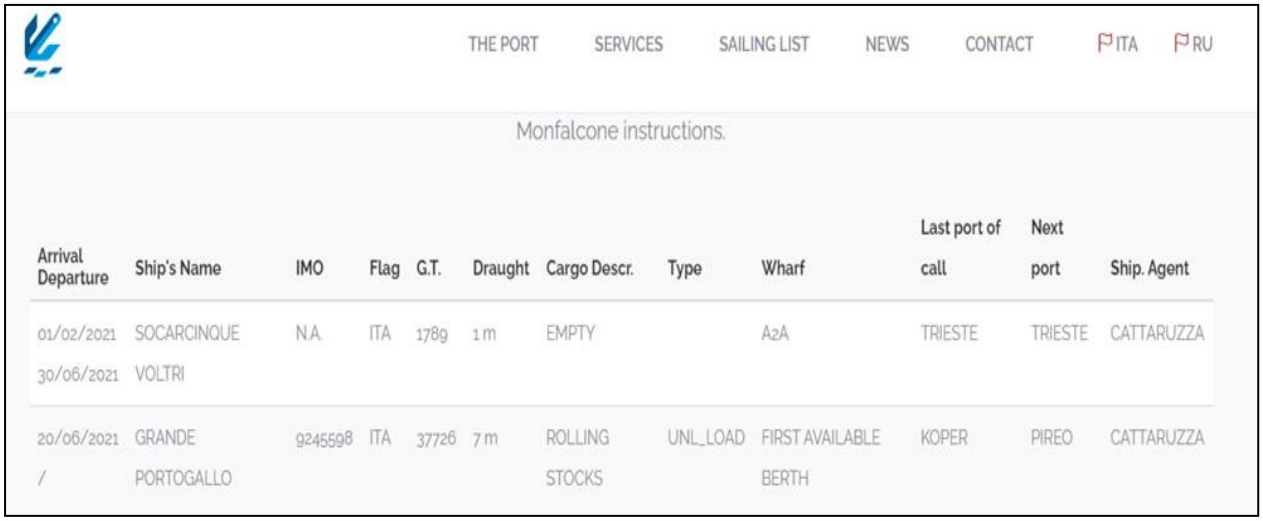


Figure 4: Port of Monfalcone sailing list interface (origin of data retrieval)

The NGSI agent then processes the entries and generates enough information (following the established data model in PIXEL) to represent that info about vessel calls in a proper way. An example of the register of one vessel arriving to Monfalcone is the following:


```
"data": {
  "departure_dock": 1621382400000,
  "loading_cargo_type": "Slabs",
  "IMO": "//",
  "loading_berth": "2",
  "source": "urn:pixel:DataSource:itmnf:VesselCall_COVID",
  "loading_cargo_fiscal_type": "",
  "journeyid": 0,
  "loading_dangerous": false,
  "unloading_tonnage": 0,
  "arrival_dock": 1621296000000,
  "unloading_cargo_fiscal_type": "",
  "loading_tonnage": 1100,
  "unloading_cargo_type": "",
  "unloading_agent": "",
  "unloading_berth": "",
  "name": "AB2+FLORA P",
  "unloading_dangerous": false,
  "location": {
    "lat": 45.783333,
    "lon": 13.55
  },
  "dataProvider": "http://www.porto.monfalcone.gorizia.it/sailinglist.asp",
  "loading_agent": "FRIULTRANS"
},
```


Figure 5: NGSI agent example

Subsequently the building of an enhanced version of the PAS Model for the COVID use-case took place that consisted in introducing new parameters in the business logic of the model for processing the new conditions for the area (such as mentioned in the paragraph before, the number of workers and sanitary restrictions).

The procedure followed created such constraints and includes them in the supply chain definition of the operations of the Port of Monfalcone: to each process, machine and area were assigned a fixed and maximum number of workers.

Then, the upgraded PAS model was published, installed, and run within the platform of PIXEL. The system was designed so that the new results would include enough output data for the representation of the designed interface. In that sense, the solution adopted was the replication (in another index) of the results of the PAS (all documents generated under the same index after a usual PAS execution) in a different index, including the with/without COVID scenarios simulation result:

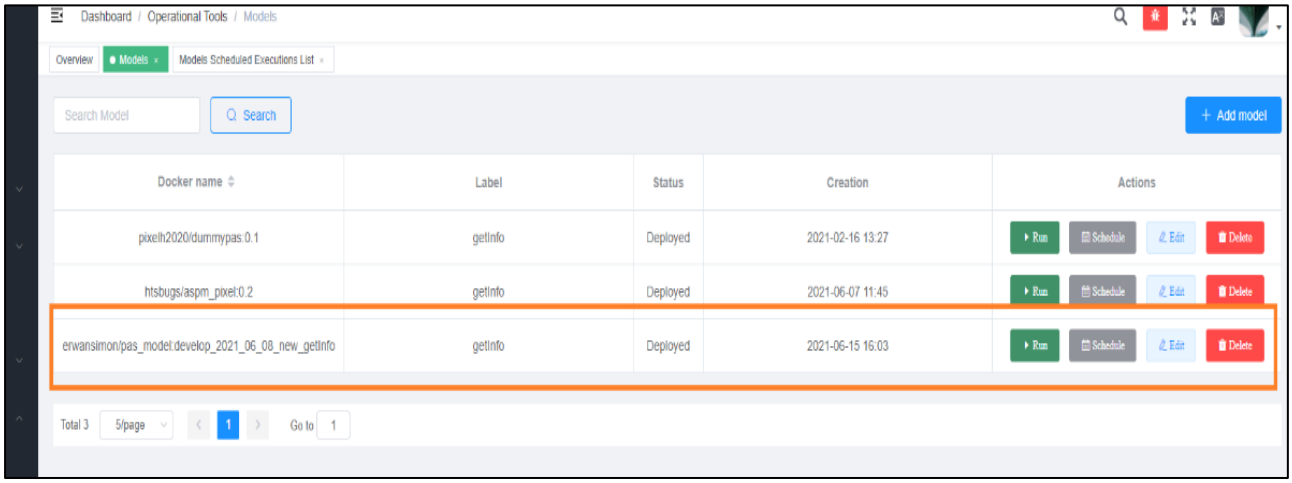


Figure 6: Model installed and ready to be executed.

Afterwards, the model was scheduled so that the tool will be automatically fed with the most recent data. It was decided to schedule the model to be run each week (at the end of the week), including as input the list of vessel calls announced to arrive at the port within the last week.

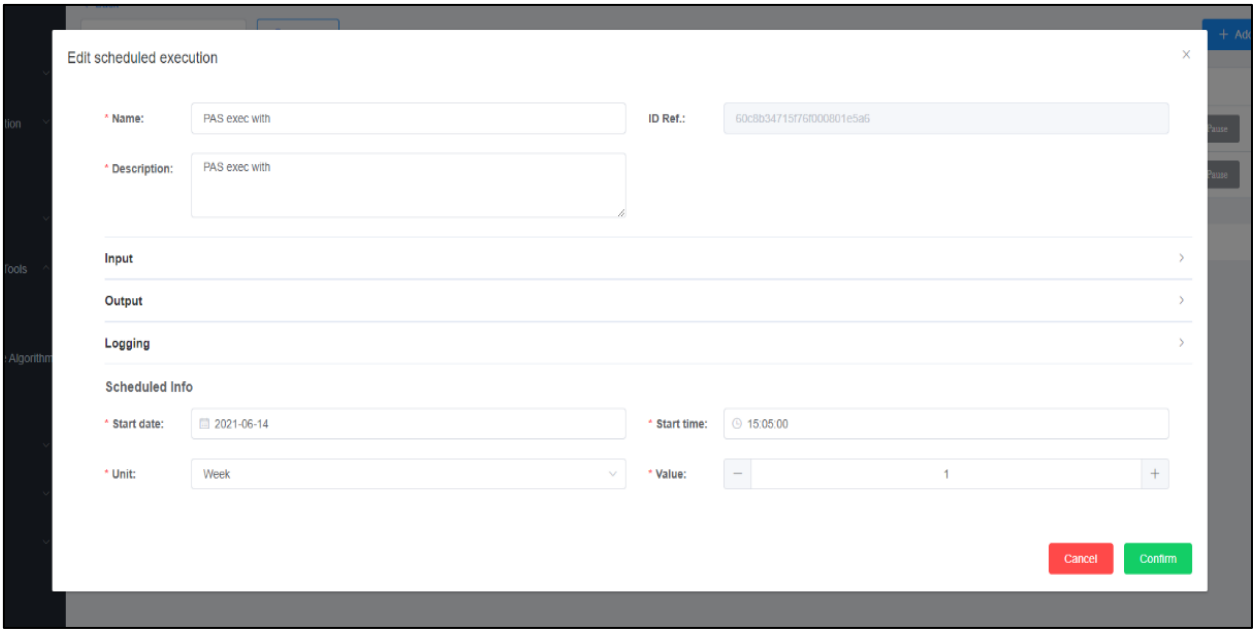


Figure 7: Model schedule Example

A summary of the presented flow can be observed in the following image:

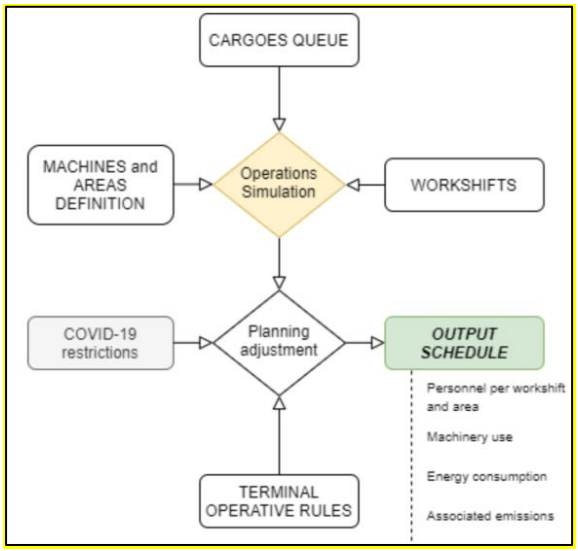


Figure 8: COVID Flow example

From this procedure, once the architecture was finalised (requiring different adaptations and fine-tuning), it was possible to create an ad hoc Visualization Module for this Pilot.

3.2.3. Visualization

The result of the execution can be observed in a user-friendly interface, as stated below.

For the port managers, the visualisation expected is a map with the areas of the port representing the density of workers in a timeframe (decided in the pilot: per hour). Thus, the end user in the port recognises how the operations have been rescheduled to meet the density limitations, while observing the evolution during the week and being able to compare with a not-pandemic scenario. The user can also see in advance in a specific timeframe the evolution of the parameters.

To provide a better look of the interface, the different areas of the terminal have been displayed in different colours depending on if there is a surpassing of the threshold of the restrictions in each hour shift.

Figure 9: Visualization of COVID platform with colours

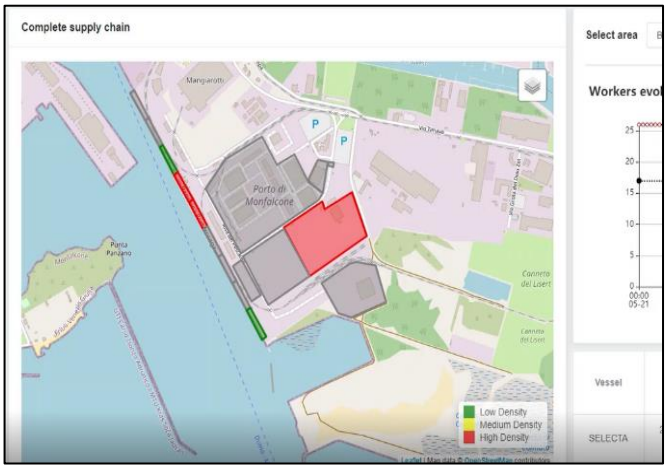


Figure 10: Visualization of COVID Platform with colours

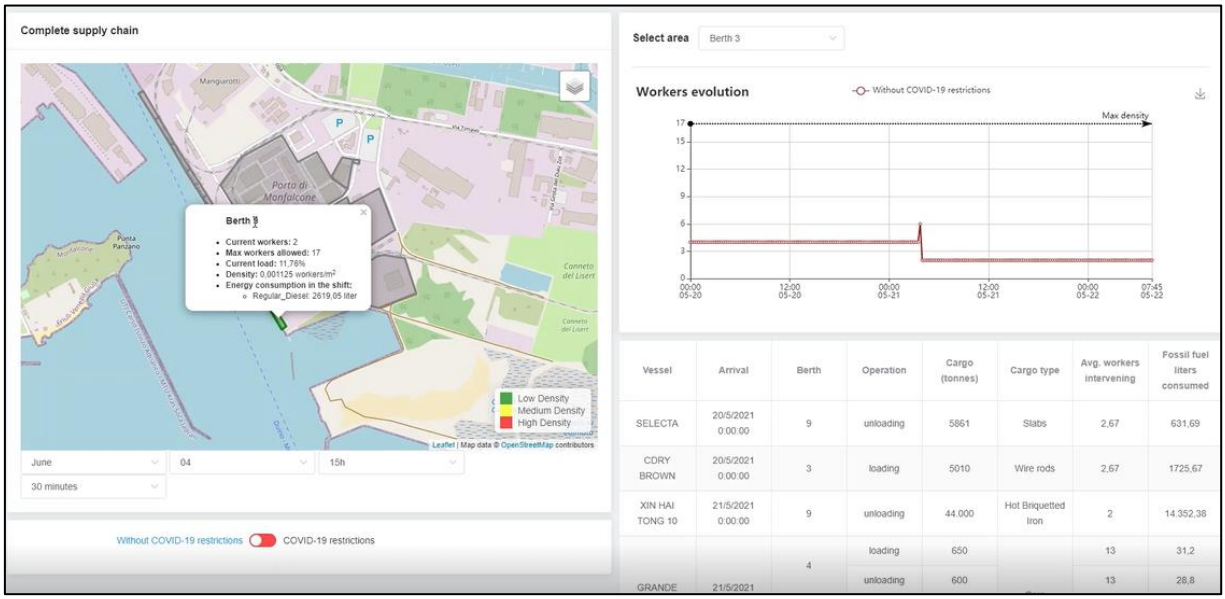


Figure 11: Visualization of Covid platform with data

The results take advantage of the already-developed technology in PIXEL and include energy consumption estimations as well (*see figure below*).

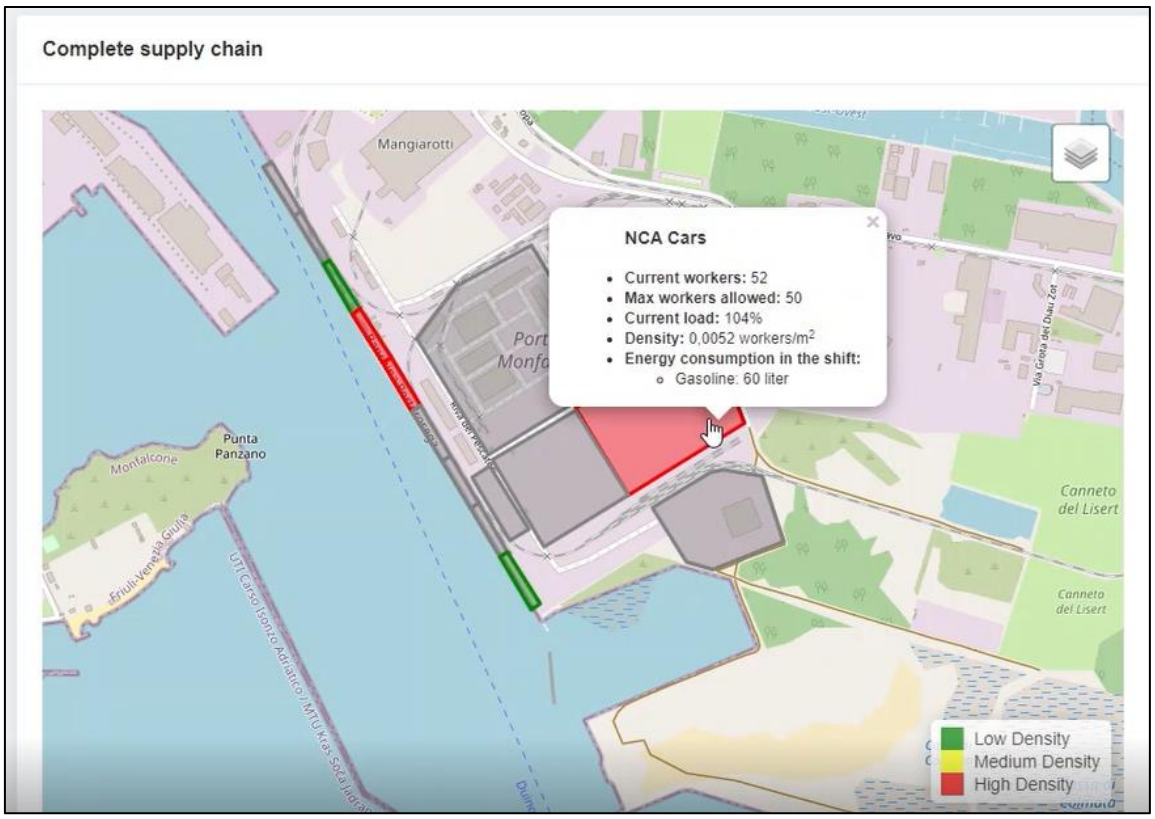


Figure 12: Visualisation with focus on data

Considering the simulation created, with this tool able to monitor the density of workers in the port, port operators have the potential to manage their activities. However, the main scope of the pilot is to provide a simulation.

For the extrapolation to other ports, actual actions in the port may be changes in shift plans for personnel and machinery so that the work would be redefined to minimise social aggregation at different areas of the port, therefore decreasing contact and virus spread among workers and, at the same time protect drivers' needs for example suggesting a re-routing to other truck parking areas in inland terminals that have all the necessary services.

The different information included in the visualization interface is the following:

- A map with the density of workers per area and per work shift. Clicking over each area a pop-up appears that provides information about the current workers, limit, energy consumption, density per square meters;
- A time evolution graph that includes the threshold of the max density that can be reached per area. A dropdown selector allows the user to specify which area wishes to be observed.
- A table including the vessels used in that calculation period. This table contains the name of the vessels, the berth where they will arrive, operation (loading or unloading), amount and type of cargo, average of workers intervening and the associated energy consumption to process the cargo of those vessels;
- The tool allows the user to shift from "without pandemic scenario" and "with pandemic scenario". The latter forces the simulation to adjust to the allowed density at any moment.

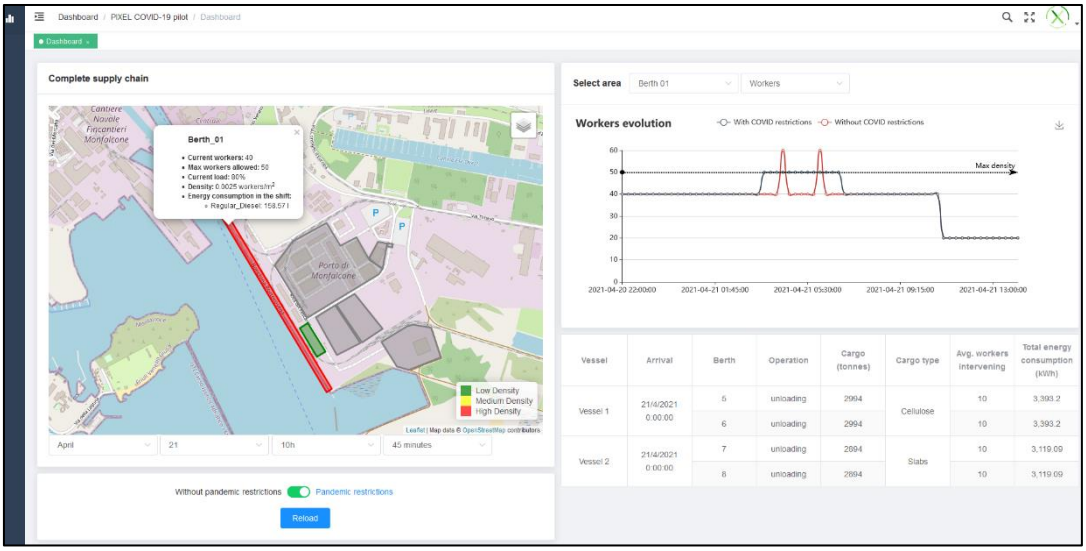


Figure 13 Visualization. Detail of a specific area

3.3. Results

The system implemented allows to cope with the following requirements:

1. Definition of a level of density of the areas so to ensure physical distancing in accordance with the WHO World Health Organization provisions;
2. Calculation of the real-time level of density;
3. Alert in case of density exceeding the level defined;
4. Estimations based on vessel calls and scheduled operation.

At the moment, the system is not designed to trigger automated actions in case of alerts; the port operators can monitor the density level and put in actions the required measures in the case of deviations.

The system therefore allows not only to ensure physical distancing but also to implement further measures to protect users' health and safety in case of emergency, both sanitary and of any other nature. For example, when used by a maritime or health Authority or a private terminal, this tool can be useful to:

1. Isolate and put into quarantine one of the ports "blocks".
2. Increase physical distancing by decreasing the maximum level of density of the areas allowed.
3. Block and/or cordon off users coming from a specific vessel.
4. Re-route a specific section of users/truck drivers to inland terminals.

These are both ordinary and extraordinary measures that can prevent the risk of spread of viruses and/or solve emergencies related to users' health and safety.

COVID-19 restrictions in Italy defined a social distance of at least 1 meter among workers. When this distance cannot be achieved, workers have to use specific IPD's.

Taking in consideration this limit, the maximum number of workers allowed in each port area is one worker per square meter (without IPD's).

After the implementation of the COVID pilot in the Porto of Monfalcone, it can be noticed that the model never highlighted the surpassing of the threshold defined by national law in terms of social distancing. Even introducing a stricter restriction, such as considering 2 square meters for each worker, the model did not highlight critical scenarios.

This is related mainly to three factors:

- port activities are (almost) carried out in wide areas, where social distancing is not a critical aspect;
- the tonnage of goods handled by the port (mainly dry bulk and vehicles), decreased by almost 30% since the spread of COVID-19;

- The operative surfaces considered by the model are related to the IOS data available for the PAS that identifies the single berth as the smallest port section available. This is not adequate to measure the density of workers in small spaces where social aggregation can occur, such as the ship hold.

These aspects will be further monitored in the ongoing tasks related to the evaluation of the project results also with reference to the serological survey promoted, on a voluntary basis, by the Port System Authority of the Eastern Adriatic Sea on the workers of the Port of Trieste and Monfalcone.

So, considering that the system implemented is compliant with the requirements obtained from the ports to solve the different needs, the lesson learnt from this is that PIXEL implemented a platform that can be further upgraded with many potential additions and future variables. For example, in a technological manner, some added-value technologies can be added to the architecture (e.g., Machine Learning features to predict occupancy in the long term, real-time monitoring endorsed by other means of data -cameras, RFID sensors, wearables-, et. al.) while in a usability manner it could be implemented by inserting other options variables to be included in the analysis and so, being able to create new scenarios. It would also be interesting to capture other information (such as air emissions) and relate them to the social distancing measures to have a more complete picture and overview of which effects those aspects may have in the terminals of the port.

4. New Uses Cases Tested

This section describes models applied in the ports that were not initially considered and that endorse the flexibility and scalability of PIXEL. During the development of the models, some of them are related, so that if the first one is applied, the second one can be applied without much effort, this is for example the case of the PAS and Energy model. In other cases, when a model has been described and used in one port, it has been of interest to another port and the latter has decided to use it, e.g., the use of the energy Production and balance Model in Monfalcone and the use of the COVID model in THPA.

4.1. Use of the PAS model in THPA and PPA

The task 7.4 "Port-City integration trial" involves the development of pilots for the port of Thessaloniki and Piraeus. Both pilots focus on the planning and optimisation of urban logistics. Due to the proximity of these ports to large urban centres, measuring the environmental impact of port activity on the city is a major concern. From the beginning of the project both ports focused on measuring the impact on noise and air quality, without forgetting the impact on traffic and congestion. Initially the application of the PAS was not contemplated. But since having this model facilitates the implementation of the PEI (see section 5), as the outputs generated by the PAS can be used for the calculation of the PEI, both ports decided to implement the PEI considering the peculiarities of each port.

4.1.1. Use of the PAS model in THPA

The PAS model has been proficiently applied to ThPA in order to evaluate how port operations (related with vessel calls) impact in terms of both energy consumption and pollutant emissions. PAS has been applied on equipment and tools available at ThPA yards; a research activity has been performed in order to collect technical data about each different equipment involved in loading and unloading activities.

Input data of the PAS model are based on:

1. Vessel planning (arrival date and cargo type). The total of the tonnage of the ship is used for the calculations.
2. Time tables (Containers operations and Other Operations)
3. Pollutants that will be calculated and type of energy used by equipment (Diesel and Electricity Grid).
4. Equipment used by the Port for its cargo handling activities (with averages for year of manufacture, energy consumption rates, throughputs and engines power)
5. Areas (GPS Coordinates also inserted).
6. Content types (names corresponding to vessels' calls)
7. Supply chains (the limitation of not being able to insert more than one instance of the same machine in one operation of the supply chain has been overcome by creating identical operations working in parallel).

4.1.2. Use of the PAS model in PPA

The PAS model has been proficiently applied to PPA in order to evaluate how port operations (related with Container vessel calls) impact in terms of both energy consumption and pollutant emissions. PAS has been applied on equipment and tools available at PPA yards; a research activity has been performed in order to collect technical data about each different equipment involved in loading and unloading activities.

It is noted that the PAS has been applied to Container operations only and not for the Cruise and Passengers operations, due to:

1. Vessel calls data do not contain the number of passengers in Cruise and Passenger ships.
2. Number of passengers in passenger ships fluctuate within different months in a year (so an approximation based on the tonnage of ships would produce results not corresponding to reality).

The fluctuation of passengers carried with Passenger ships can be investigated in the future. During the time of the publishing of this document (and because of the Covid restrictions in the past 16 months), such investigation would not produce usable results.

Input data of the PAS model are based on:

1. Vessel planning (arrival date and cargo type). The total of the tonnage of the ship is used for the calculations.
2. Time tables
3. Pollutants that will be calculated and type of energy used by equipment (Diesel and Electricity Grid).
4. Equipment used by the Port for its cargo handling activities (with averages for year of manufacture, energy consumption rates, throughputs and engine power). The imitation of not being able to insert more than one instances of the same machine in one operation of the supply chain has been overcome by creating virtual machines (bundles of the same machine - e.g., STS_Gantry_Crane-VM-of-3 is a bundle of 3 identical STS Gantry Cranes working in parallel - with multiplied throughputs, energy consumption and engines power).
5. Areas (GPS Coordinates also inserted).
6. Content types (names corresponding to vessels' calls)
7. Supply chains

4.2. Use of the PAS energy demand Models in All ports

Initially designed and developed to be used in the energy use-case of GPMB, the Port Activity Scenario (PAS) and energy models have been integrated in all ports of the PIXEL Consortium. PAS outputs provide, under a simulation perspective, the total amount of energy consumed by the machinery needed to operate one vessel. Using that information, altogether with the type of fuel/power used, the results can be leveraged by the PEI to estimate the air emissions attributable to the terminal activity in a period of time. This functionality has been directly included as a module of the PAS. For every considered cargo that is transiting inside the port, the corresponding sequence of operations across time is provided through the Port Activity Scenario described in the previous section. For each operation, the duration, the machine's energy type and its unit consumption values are available (from input and parameters combination). For each operation, the energy cost is calculated as the product of atomic operation's duration multiplied by its machine's unit consumption. Note that the energy consumption independent of the PAS (e.g., buildings with thermal regulation) could also be considered and add to the energy's consumption time series. Then for every energy's type, all the consumption occurring during the same timeframe can be summed to get the corresponding total consumption. The sequence of all those timeframes constitutes the energy consumption time series. This functionality is directly included as a module of the PAS.

The Port Activities Scenario (PAS), combined with the energy model and emission factors, is a transferable and applicable tool for small and medium European ports that allows to model port supply chains. The output of this model can be used as an input for evaluating the energy consumption and with the use of emission factors to provide an estimation of pollutant emissions. Following the previous, the usefulness of the PAS for the Port Environmental Index (PEI) can be understood two-fold: i) Providing information on the berthing time. Ideally, a port will provide the values of the time a vessel is being berthed, hoteling and manoeuvring within the port maritime area. The PAS outputs can be used to obtain a (approximated) value of how much time is a vessel operated (loaded/unloaded), which can be considered a basis for the berthing/manoeuvring ratio. ii) Providing information on the energy consumption of the terminal to calculate air emissions. Energy consumption is, usually, a complex metric to quantify, in contrast to what logical thinking would indicate.

4.3. Use of the energy Production and balance Model in Monfalcone

During the dissemination activity of PIXEL to one operator of the Port of Monfalcone (a terminal operator), the company expressed the interest in evaluating if a transition from diesel-propelled equipment to Electric Energy (EE)-propelled ones would be feasible considering only the use of photovoltaic (PV) energy. This

was the starting point of the technical activities that involved APT, UPV and CATIE to transfer the model developed in the energy use-case of GPMB, in the port of Monfalcone, to fulfil different needs arose and test the capacity of the system to be adapted to different ports.

Essentially, the objective of this cross-pilot integration was to answer the question: would a series of PV panels installed in the terminal of the port of Monfalcone supply enough energy to cover (desired) equipment demand? To answer that, the problem was divided in two branches to be separately tackled. First, it was necessary to understand how much energy a series of PV panels provide in the Port of Monfalcone (based on estimations). Secondly, an analysis should take place to understand how much energy would be needed to "feed" the desired equipment. To tackle the former, it was decided to leverage the Energy Production balance model that was deployed for the Port of Bordeaux, which relied on the PVGIS tool with a series of input parameters. For the latter, the Port of Monfalcone together with the operator should identify which machinery should be assumed, its consumption and working time. A detail of the two processes and the associated work can be found below:

1 - Energy to be supplied by PV panels (estimation):

Same as for the Port of Bordeaux, the PVGIS tool owned by the European Commission was used (see D4.2). It consisted in an 11-year span (2005-2016) range of values for a series of assumptions. The considerations assumed (parameters introduced in the tool for obtaining an output) were:

- Area size: 500 square meters. This information was based on the total surface that the company can allocate for the installation of photovoltaic panels.
- Reference location: Latitude: 45.789º, Longitude: 13.557º.
- 3 meters of elevation
- Slope of 38º, auto-shade and auto-azimuth of 1º
- Nominal power of the system: 70.8 kWp, while the losses were estimated as 14%.
- PVGIS database used: SARAH.

The previous generated a .CSV file with enough information to make an estimation for that production, including a periodicity of 1 hour.

```
date,PW,Gb,Gd,t2m,WS10m,Int
20050101:0010,0.0,0.0,0.0,7.88,2.29,0.0
20050101:0110,0.0,0.0,0.0,7.69,2.29,0.0
20050101:0210,0.0,0.0,0.0,7.49,2.29,0.0
20050101:0310,0.0,0.0,0.0,7.3,2.29,0.0
20050101:0410,0.0,0.0,0.0,7.35,2.32,0.0
```

Drawing from those outputs, an NGSI agent was generated in order to dynamically incorporate the data into the Information Hub of the Monfalcone's PIXEL instance. Some indications of the agent development are included below, whereas a more complete documentation can be found at the Git portal of the project (now, private, soon-to-(potentially)be public):

- For a Photovoltaic Measurement (predicted) at the port, based on the 11-year span processing by PVGIS provided by results in WP4 of PIXEL project (https://ec.europa.eu/jrc/en/PVGIS/tools/hourly-radiation).
- Following the Smart Data Model PhotovoltaicMeasurement
- Logic of the agent:
    - It periodically stores (each year) the values (estimated) of the Power that would be generated per PV panels at the Port of Monfalcone during the next year (month per month).
    - Each entry corresponds to the estimated value of power associated to a specific month (e.g., July 2021).

- The estimation is a direct calculation from the sum of the values of each month, that is based on the median of every day of the year of the 11 years in the available historic.

2 - Energy to be required from EE propelled equipment (estimation):

The activity started with the contribution of the port operator that provided the necessary information about the type and number of the equipment they are interested in replacing with EE propelled ones.

The information about the equipment was the following:

- 18 tractors model: *Terberg modello YT203-EV* with the following data:
  - An average consumption can be estimated at 12-15 kWh/h (selected: 13.5 kWh/h). The working autonomy, on the other hand, depends on the size of the battery packs which can vary from 100 to 220 kwh, so the autonomy varies from 6 hours to 14 hours while the charging times range goes from 2 to 6 hours depending on the charger used".
  - With the collaboration of the terminal operator, data were gathered about the average work hours performed by those 18 tractors per month.
  - The two previous data were used to feed the model.
- 20 forklifts model Kalmar Electric 9-18T Forklift trucks with the following data:
  - The average operative consumption of the forklift family capable of lifting from 13 to 16 tons, can be estimated in 20 KWh/h to 28 KWh/h (selected 24 kWh/h).
  - With the collaboration of the terminal operator, data were gathered about the average work hours performed by those 18 tractors per month.
  - The two previous data were used to feed the model.

Afterwards, both branches needed to be merged into an actionable tool for the stakeholders of the port (the Port Authority and the terminal operator). In that regard, there was the need to create a visualization to represent together both (i) the energy -estimated- to be produced by the PV panels and (ii) the consumption of the machinery. The fact of putting those data together would allow end users to analyse the viability/return of investment in advance.

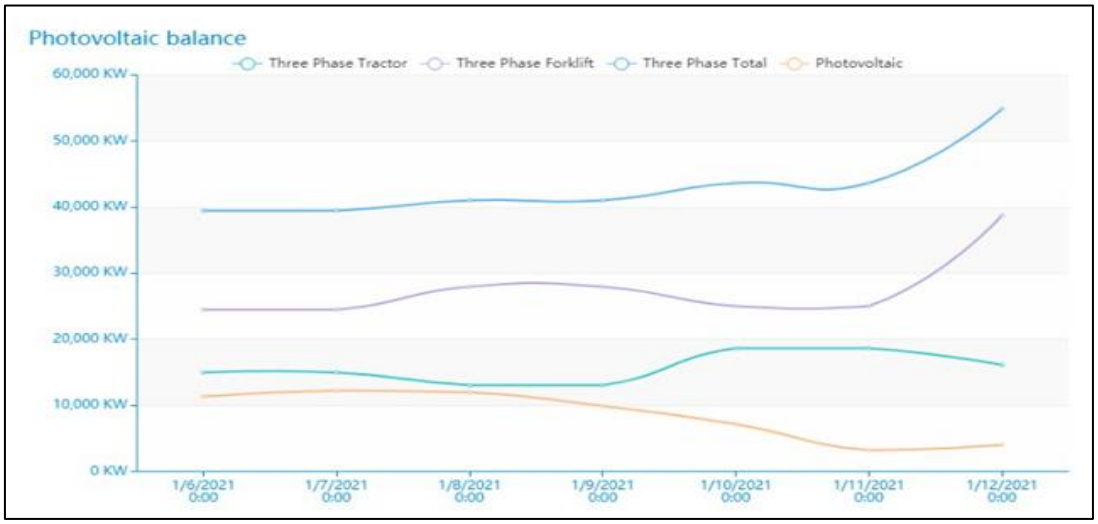
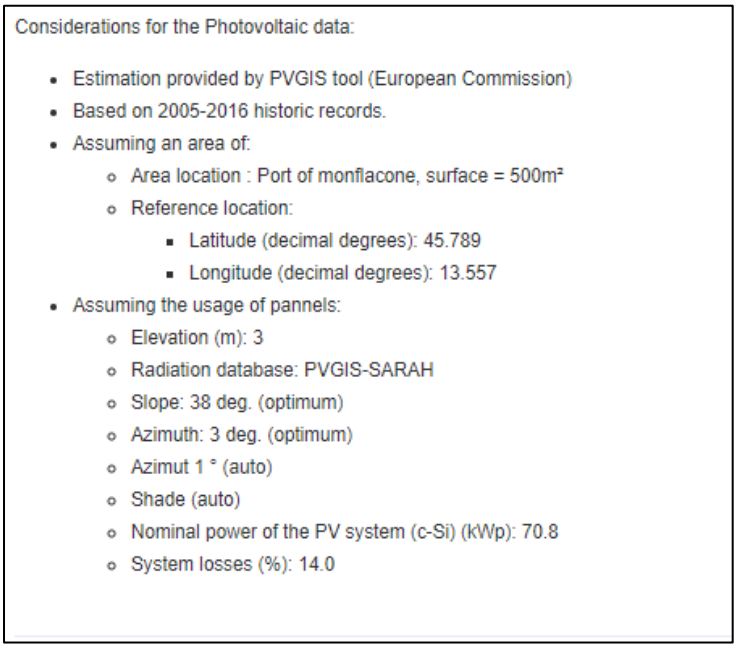
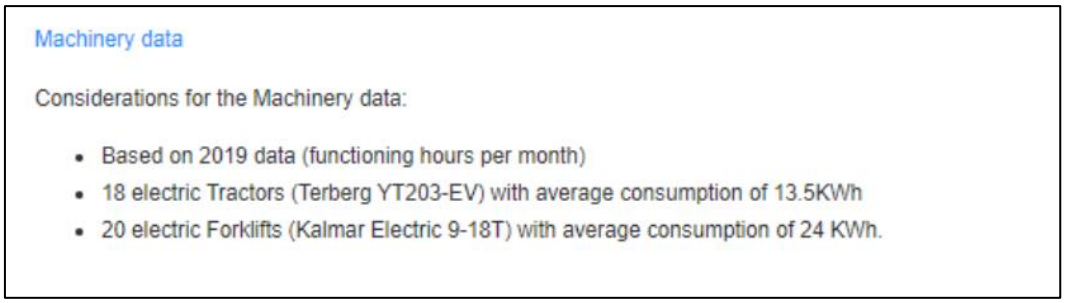


Figure 14: Photovoltaic Balance


Considerations for the Photovoltaic data:

- Estimation provided by PVGIS tool (European Commission)
- Based on 2005-2016 historic records.
- Assuming an area of:
  - Area location : Port of monflacone, surface = 500m²
  - Reference location:
    - Latitude (decimal degrees): 45.789
    - Longitude (decimal degrees): 13.557
- Assuming the usage of pannels:
  - Elevation (m): 3
  - Radiation database: PVGIS-SARAH
  - Slope: 38 deg. (optimum)
  - Azimuth: 3 deg. (optimum)
  - Azimut 1 ° (auto)
  - Shade (auto)
  - Nominal power of the PV system (c-Si) (kWp): 70.8
  - System losses (%): 14.0

Figure 15: Photovoltaic data


Machinery data

Considerations for the Machinery data:

- Based on 2019 data (functioning hours per month)
- 18 electric Tractors (Terberg YT203-EV) with average consumption of 13.5KWh
- 20 electric Forklifts (Kalmar Electric 9-18T) with average consumption of 24 KWh.

Figure 16: Machinery data

These graphs were made available on the PIXEL dashboard to summarize the results of the model.

The provisioning of this tool (cross-pilot result) in the Port of Monfalcone drove the stakeholders in the port to state the following:

After an intense activity to collect the missing data needed, such as the consumption of the new generation EE propelled equipment, and the integration of the model previously implemented, a new feature is available in the PIXEL platform that is devoted to highlighting the energetic balance related to the green transition proposed by the operator is sustainable with photovoltaic energy.

The model, even if it is at its first release, can answer the question made by the terminal operator, but it has plenty of potential for further development and fine-tuning. One of the most troublesome peculiarities of the EE equipment, for example, is that their battery has to be charged rather frequently than the fuel refill that has to be done in traditional equipment; this activity takes more time than a fuel refill too: this could be a very interesting factor to go deeper in the efficiency analysis that an operator should face in a real scenario.

4.4. COVID Pilot in THPA

Taking advantage of the development of the COVID pilot for the Port of Monfalcone (see section) - which already was intended to demonstrate the flexibility and scalability of the platform - the team of T7.6 decided to go one step ahead and test the same tool in another port of the project, thus validating the cross-fertilisation and usability of the tool with a "close-to" plug-and-play fashion.

As a matter of fact, the actions needed to perform by THPA and the technical partners associated to the deployment of PIXEL in the port were the following:

1. Fulfil the PAS forms including proper information (that was gathered by the port for this task):
    a. Machinery information.
    b. Supply chain information.
    c. Work shifts and priorities data.
    d. Areas of the terminal (including the geo-location of the polygons).
    e. Max. number of workers allows per area.
2. Develop the vessel call agent following the model of APT, inspired from the code of that instance and from the experience of GPMB that created the vessel calls agent being directly fed from the Information Hub.
3. Install the PAS model in the PIXEL instance of THPA, using the Operational Tools.
4. Run the PAS model using the guidance and functionality of the Operational Tools.
5. Schedule the PAS model instance to be run every week using the data of incoming vessels of the past week.
6. Activate the visualization via the database of the Dashboard and granting the proper users with enough permission to access the tool.

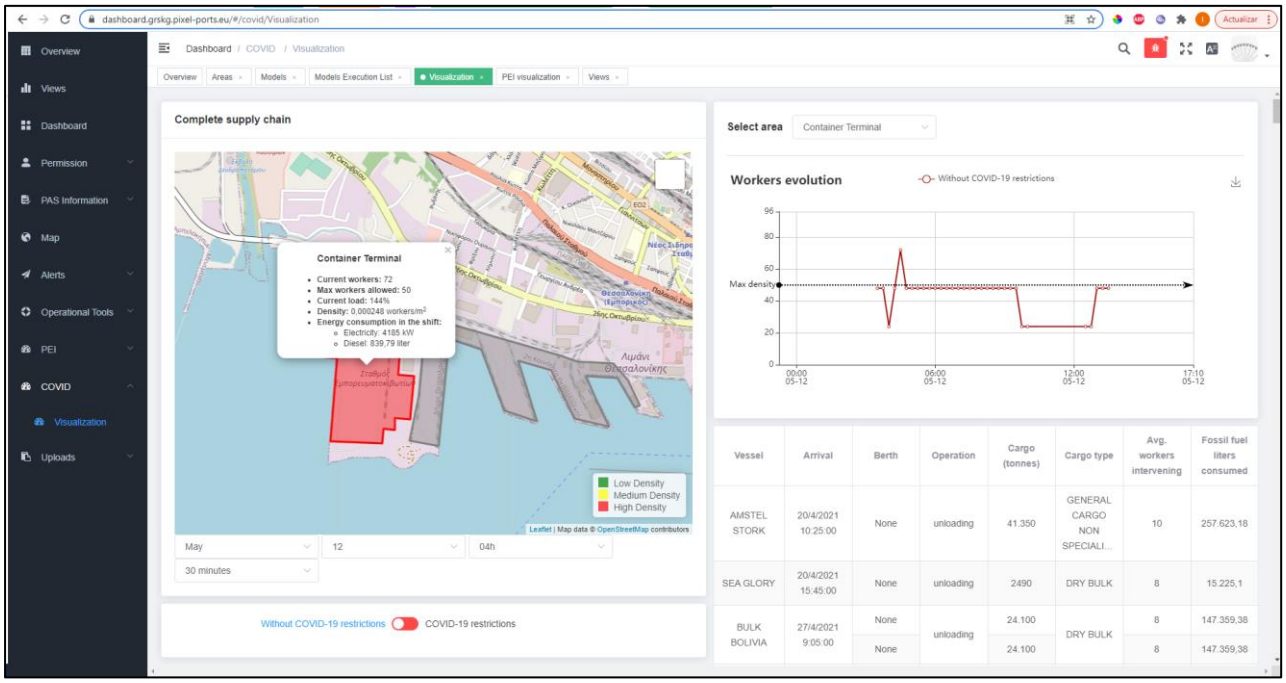


Figure 17: THPA COVID results

5. Generalization of PIXEL

Since the project was created, it has always been kept in mind that the results of the project should be easily applicable in other ports. Considering this requirement, both the platform and the models created were designed to be generic/flexible. This section explains the main relevant aspects included in the development of the platform and models to obtain a generic solution.

The section starts describing how the different components of the PIXEL platform have been designed to be extensible and allow both new data sources and new models to be easily integrated into the platform. Extensibility mechanisms, based on the concept of microservices have been defined to allow both models and services to be integrated by using containers, which makes them independent of the underlying technology and permit them to be developed in any programming language.

Next, the work done on the PEI and PAS-Energy Models, to make them generic and flexible to be applied to the four PIXEL ports, has been described.

Finally, the linkage and cross-utilisation between two results of PIXEL, "Port Activity Scenario" and "Port Environmental Index" has been explained.

5.1. Extendibility (pluggability) and flexibility of the platform

The PIXEL platform has been designed with the requirement of being extensible. This extensibility applies mainly to the capture/import of different data sources, the addition of new models and algorithms based on Artificial Intelligence that provides new functionality and the creation of dashboards and visualisations.

5.1.1. Use of Data Models

The use of "standard" Data Models on PIXEL allows the platform to be agnostic from the format of the data provided by the different Data Source connected. The FIWARE NGSI standard allows us to define extensible Data Models and to rely on the community to validate and propose schemas that are not specific to one use case implementation.

The use of those Data Models makes it possible to quickly add new Data Source to a PIXEL Platform and facilitate the use of those data in the different Models. The format of the data is agnostic from the format of the Data Source but rely on the type of data manipulated.

The use of the Smart Data Models proposed by FIWARE allow us to reuse the agent to collect data from one source to any "powered by FIWARE" projects. It also allows the PIXEL platform to quickly plug FIWARE compatible data sources to the platform.

In some cases, there is not a proper data model defined to represent the new data to be integrated. In that case a new Data Model will be created. The creation of new Data Model should follow the recommendations of the FIWARE community describe in their Github repository:

- https://github.com/smart-data-models/data-models

All data models used on PIXEL have to be published in our own repository:

- https://gitpixel.satrdlab.upv.es/iglaub/Data_Models/src/master/specs

The List of FIWARE Data Models used on PIXEL is presented below:

Table 2: FIWARE Data Models used in PIXEL

Data Model name	Description
Device	An apparatus (hardware + software + firmware) intended to accomplish a particular task (sensing the environment, actuating, etc.).
OffStreetParking	Manage context about off-street parking
TrafficFlowObserved	Manage context about traffic flow observation : averageVehicleSpeed , congested, vehicleType ...
WeatherObserved	An observation of weather conditions at a certain place and time. This data model has been developed in cooperation with mobile operators and the GSMA.

The list of PIXEL specific Data Model is presented below:

Table 3: New Data Models created in PIXEL

Data Model name	Status	Push to the community	Description
VesselCall	new model	In progress	Store vessel calls information on each port (arrival, departure, operation, cargo)
AirQualityObserved	update	No	Extension of the FIWARE model to store Bordeaux specific data
EnergyConsumptionMeasurement	new model	Too specific	Model to store Energy consumption provided by Bordeaux supplier.
MarpolWaste	new model	In progress	Store MARPOL Waste information from the ship
MarpolWasteTerminal	new model	In progress	Store MARPOL Waste information from the terminal
TideSensorObserved	new model	Too specific	Store Tide Sensor data form Bordeaux
WeatherObserved	update	No	Add of some specifics fields provided by the source
WeatherObservedSencrop	update	Too specific	Add of some specifics fields provided by the source
NoiseLevelObserved	update	No	Add of some specifics fields provided by the source
TrafficFlowObserved	update	No	Reduction of the FIWARE - Transportation / TrafficFlowObserved data model for the purpose of PIXEL project
TrafficFlow	new model	Too specific	Special model to store data specifics for one use case
WindObserved	update	To Discuss	Enhancement of the FIWARE - Weather / WeatherFlowObserved data model for the

			purpose of PIXEL project
EnvironmentalKeyPerformanceIndicator	new model	To Discuss	Adaptation of the FIWARE - KPI / KeyPerformanceIndicator data model for the purpose of PIXEL project
PhotovoltaicMeasurement	new model	To Discuss	The Data Model is intended to measure the continuous power transferred by the photovoltaic panel to an Inverter Device

PIXEL will share the Data Models created for the PIXEL platform with the FIWARE community to ensure that those Data Models will be maintained by the community.

5.1.2. Data Acquisition flexibility

Due to the wide variety of data sources and data formats, not all data can be integrated into the platform in the same way.

The preferred and mostly used mechanism for data gathering in PIXEL has been through the use of NGSI Agents. The agents use the FIWARE Data Models approach described in the previous section to define the structure and the semantics of the data to be integrated in the PIXEL Platform. This approach is deeply explained in section 3.4.2. "Data integration Strategy" of the Deliverable D7.2 "Integration Report v2".

These agents can obtain the data from data sources using different techniques:

- API. Most systems and sensors provide an API that allows third-party access to data through a set of specific endpoints to that purpose. Most of the API provides the data in JSON format.
- FTP. For the sensors and systems that do not provide an API, data can usually be exported to different formats, CSV and XLSX are the most common exported formats. This data is later stored on an FTP server. The agent reviews the FTP periodically to read the data and load it into the platform. The XLSX format has caused many problems because this proprietary format is not easily read in some situations.
- MQTT Server. MQTT is the common interchange protocol for IoT devices. In some ports the information coming from the AIR Quality station is obtained through MQTT.

Other mechanisms used in PIXEL for data gathering are:

- Upload functionality for loading multimedia files like images or videos. Because of the type of content and the large size of this type of data, Data Models and agents are not designed to support them.

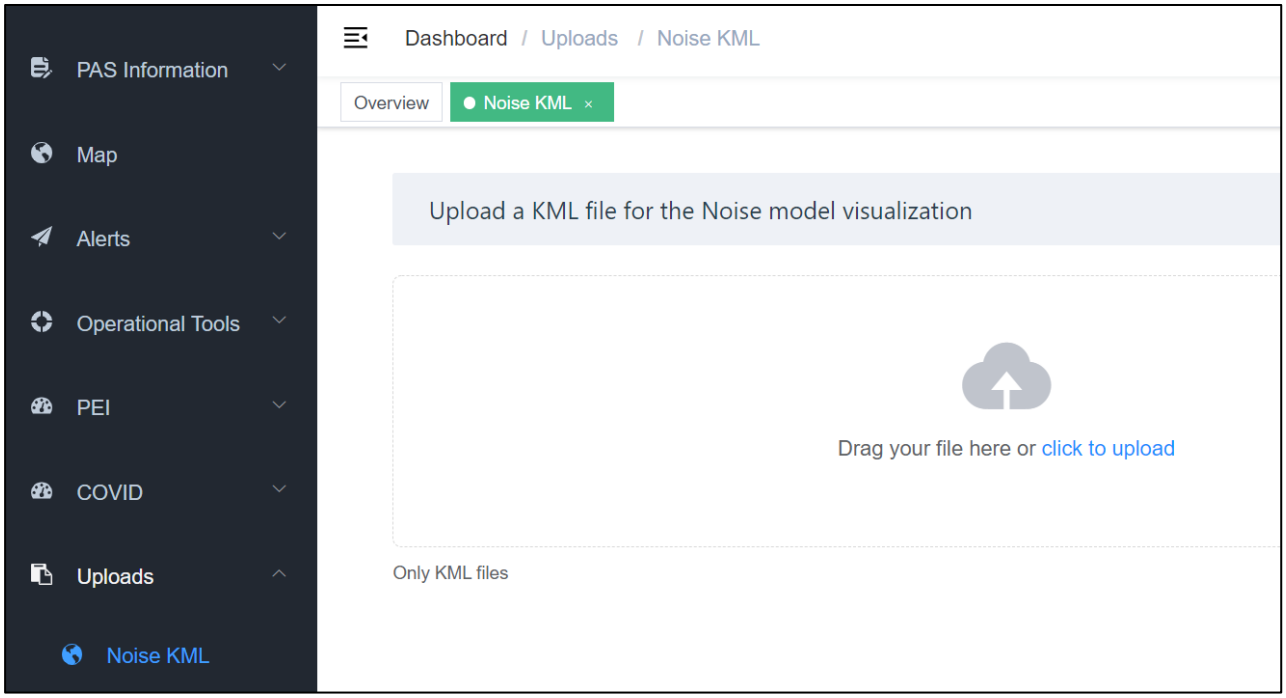


Figure 18: Example of load image functionality

Web Forms to allow users to enter the required information manually. This information is often not available in digital format or simply does not exist. These forms are created explicitly for each type of data to be inserted. The PEI and PAS models use the forms extensively to facilitate the ingestion of the data required to run these models.

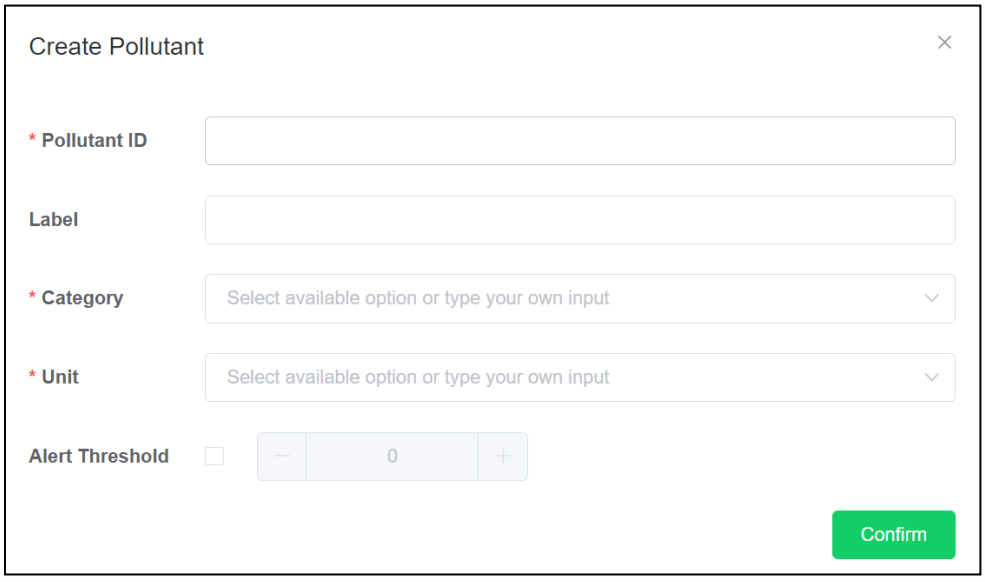


Figure 19: example of a web form for providing Pollutants info for the PAS Model

5.1.3. Integration/installation of new models

To provide a scalable and modular solution the usage of Docker containers has been used for the PIXEL platform, not only for models and predictive algorithms, but also for NGSI agents. This section is focused on the different steps needed to integrate a Docker container representing a model or predictive algorithm within the PIXEL platform through the operational tools.

In order to fully understand the process, the Operational Tools Framework needs to be described.

SPECIFICATION OF A MODEL

A model (referring to both PIXEL models and PIXEL predictive algorithms) can be specified by a set of parameters related to:

- **name, version, description, categories**: name, version, description, and categories of the model.
- **type**: type of model (model or pa). The term 'pa' stands for predictive algorithms.
- **invocation model:** describes how the model is invoked (synchronous execution, asynchronous execution and subscription model).
- **inputs**: describes the different inputs (mandatory or optional) needed by the model to be executed.
- **outputs**: describes the different outputs generated by the model after its execution.
- **logging**: describes the logging capabilities of the model.

The previous fields are specified in a **JSON format** that will be described in the next chapter. This format is included as a Docker LABEL by the time the Docker file of a (PIXEL) model is created, so that once the image is uploaded into the Docker repository (e.g., Dockerhub), it contains all information. A normal user or a software component can perform a *docker inspect* to get the description of the model. With this information, the user or a software component can generate another JSON (known as instance JSON) with specific information to execute the model (e.g., some input parameters within a given timeframe).

PUBLICATION OF A MODEL IN THE PIXEL PLATFORM. GENERAL FLOW

There are several steps to be performed if you want to publish a model in the PIXEL platform, as depicted in the Figure below.

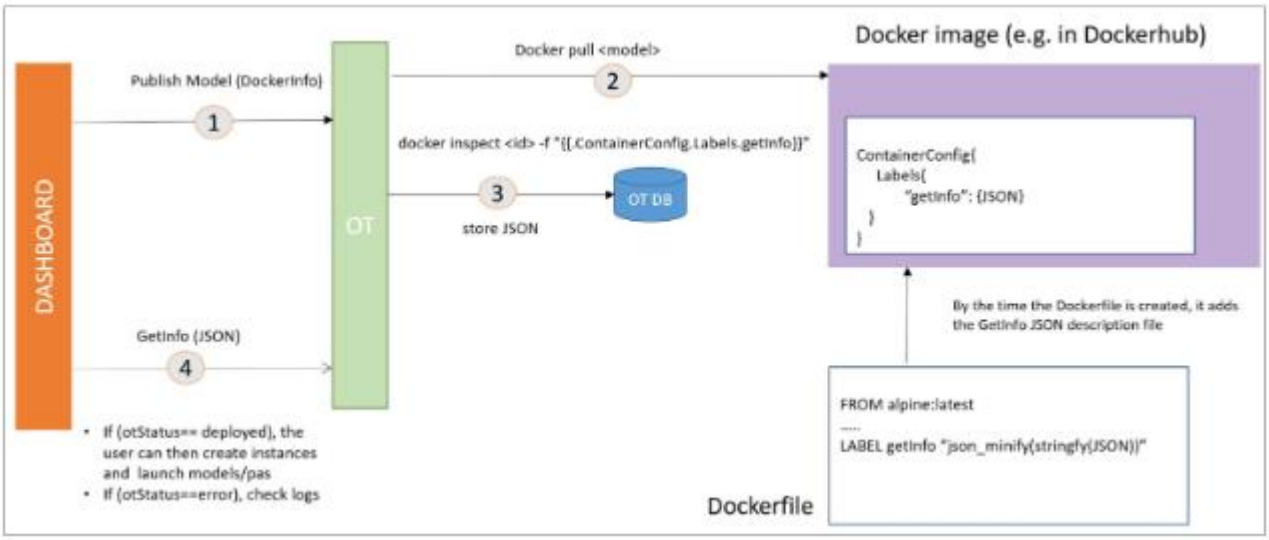


Figure 20: Publication of a model through OT Framework

Step 1: publish the model providing essential information, basically the one related to *DockerInfo* (see the *OT Framework overview*). Here the operation is performed by the Dashboard, but for testing purposes one can also use the OT API (in fact the Dashboard uses it)

- **Step 2**: Get the Docker image from the public repository (*docker pull*). In PIXEL Dockerhub under the index pixelh2020 is being used.
- **Step 3**: Inspect the Docker to get the description of the information (*docker inspect*). This is stored under the label *getInfo*.
- **Step 4**: Now the Dashboard can request the OT to get information from available models, e.g., before building an instance for executing the model.

For step *3*, it is supposed that the model developer has included a JSON structure as *label* in the *Docker file*, which will be commented in the next chapter. As it is included as label, it is important to minimize it, thus the model developer can use an online service (e.g., via **https://jsonformatter.org/json-minify**) to make the conversion from a standard JSON file.

In order to better understand the JSON format an example is provided in Annex 1.

EXECUTION OF A MODEL

An instance can be seen as a particularization of a model with specific values that allow the execution of such model. The set of parameters is very similar to the ones of the model, but here the different *value* fields are filled according to user's needs:

All fields are specified in a **JSON format** that will be described in the next chapter, but you may expect a similar structure as for models, with some remarks.

Execution of a model (instance) in the PIXEL platform. General Flow

There are several steps to be performed if you want to run a model in the PIXEL platform, as depicted in the Figure below. The main important part of the process is the interaction between the OT and model, but before running a model, either the user or the Dashboard passes the **instance.json** structure to the OT. The Docker image is composed of two main parts: (i) the *core model* by itself as developed within WP4, and (ii) the *OT adaptor* responsible for interacting with external entities (OT, IH).

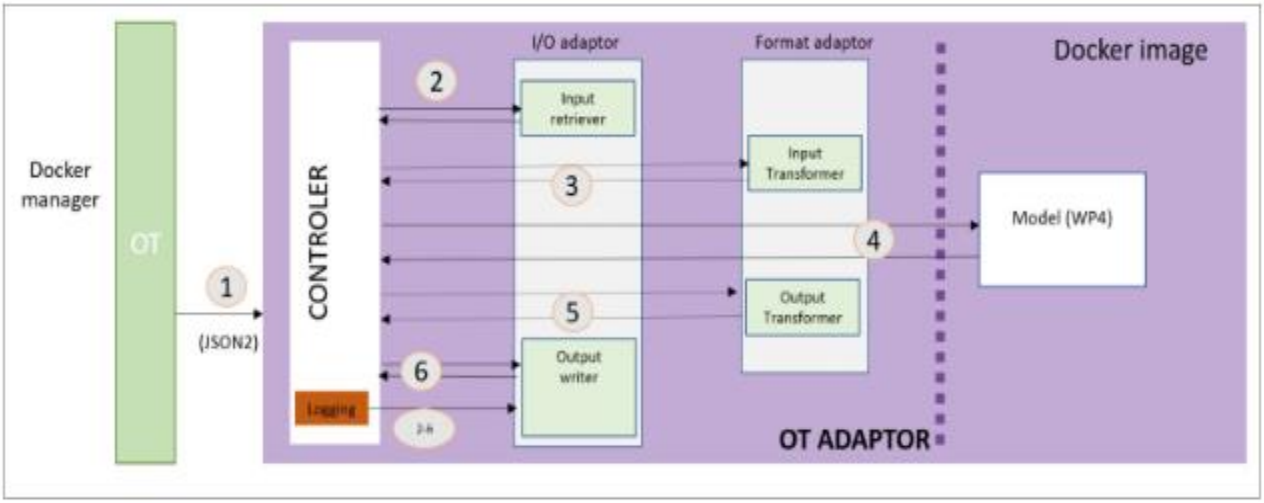


Figure 21: Execution of a model in the OT Framework

Note that the *OT adaptor* drafted in the previous Figure is a scheme of all logical components needed to correctly process the execution request, but it is up to the model developer to decide how to implement them (they can appear in different independent blocks or not)

- **Step 1**: the *instance.json* file is passed to the *controller* of the OT adaptor. It is responsible to parse the JSON file and control the flow of the execution. The JSON file is passed as a string (argv[1]). The OT will try to escape all potential strange parameters.
- **Step 2**: Once the JSON has been parsed (and checked), the *input retriever* of the *I/O adaptor* is responsible to obtain the inputs, which will typically be stored in the IH. For forced inputs, the data will be available directly in the *instance.json*
- **Step 3**: Depending on the format of the obtained input, an *input transformer* might be needed to adapt it to the native input format of the *core model*. In PIXEL, it is preferred to work with (FIWARE) data models so that the conversion is avoided or at least minimized.
- **Step 4**: The model is executed invoking the *core model* with the (transformed) input parameters.
- **Step 5**: Output data might need to be transformed by the *output transformer* if it has to be stored in a specific format.
- **Step 6**: The *controller* uses the *output writer* to store the result, typically in the IH.

For all steps (2-6), the *controller* also includes a *logging* module able to log all activity. At least it should log the start and end of the execution.

In order to better understand the JSON format an example is provided in Annex 2 (normal execution) and Annex 3 (with forced inputs).

Scheduling executions of a model (scheduledInstance) in the PIXEL platform.

A scheduled instance is an extension of an instance (see previous section) that can be launched several times according to a certain schedule. The set of parameters is identical to an instance, but additionally there is a new element called *scheduleInfo*:

```
"scheduleInfo" : {
        "start" : "2021-01-20T11:11:11+02:00",
        "unit" : "minute",
        "value" : 5
        }
```

This element is telling the OT when the model needs to be run: - **First execution**: will start at *start* time (in the example 2021-01-20T11:11:11+02:00). The date-time value is given in ISO 8601 format in the new version v0.2 (in previous version -v0.1 - it was UNIX time).

- **Next executions**: will be launched every 5 minutes starting from *start* time.

If the start time had already passed by the time the scheduledInstance is created, the OT will act as if it had started at that time and will launch the first execution in the nearest 5-minute interval (according to the example) from the current date.

To better understand the JSON format an example is provided in Annex 4.

5.2. Models Generalization

When developing the models, the requirement that they must be generic has been considered from their conception. The purpose of genericity is that they can be easily applied to other ports and not only that they work for the port for which they have been conceived.

This section describes the work done on the PEI and PAS-Energy Models, to make them generic and flexible, as proof of this is that these models have been applied in the four pilots of the project.

5.2.1. Pas Model Generalization/Customization

Thanks to the work done in WP4, PAS models have been designed in order to be a modular and adaptable tool for modeling port activities. The Port Activities Scenarios have been used to link different models and guarantee interoperability. Depending on the available data via the PIXEL Information Hub, a scenario can be completed with models related to energy, environmental pollution, and area occupancy. The Port Activities Scenario (PAS), combined with the energy model and emission factors, is a transferable and applicable tool for small and medium European ports that allows to model port supply chains. The output of this model can be used as an input for evaluating the energy consumption and with the use of emission factors to provide an estimation of pollutant emissions.

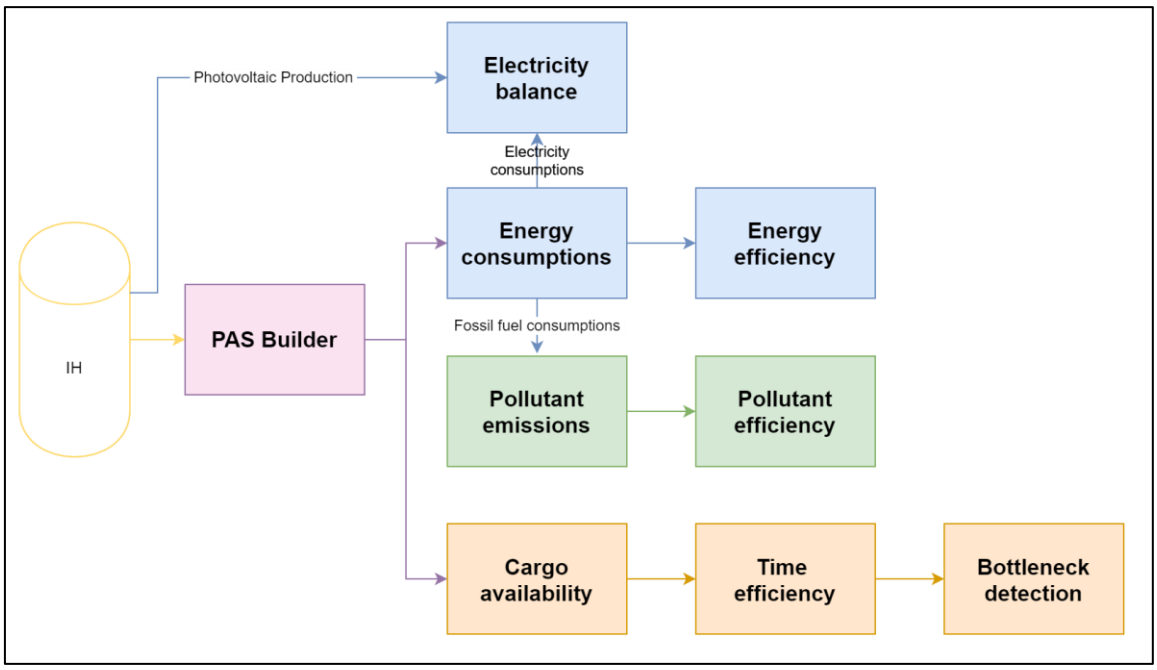


Figure 22: PAS Outcomes and interoperability of models

Another important issue for modeling port activities in an efficient way lies in the ability to deal with various degrees of data completion, precision and accuracy. Indeed, even if the PIXEL platform provides a tool to collect data from a large scope of sensors, some information could be missing or incomplete. This is due to the complexity and variety of agents composing the supply chain; internal competencies and methodologies besides the integration degree of operational and field information in real time. Thus, ports can use the PAS model with comprehensive data (detailed model and low uncertainty), screening data (some local input and external data leading to a significant uncertainty) and scaled data (average and non-specific input data leading to great uncertainty and just giving an order of magnitude).

PAS and energy models have been used to be able to simulate the energy consumption related to the port activities and the related pollutant emissions. Thus, the output of the PAS and energy models can be used as an input for the Port Environmental Index in order to calculate air emissions eKPIs. To deploy the PAS and energy models in all ports the following steps have been follow :

- Develop a specific and unique data model for vessel calls (inputs of the PAS)
- Develop a specific and unique data model for port parameters (description of type of energy, machines, supply chain, inputs of the PAS)
- PAS model has been developed with specific modules (each module can be activated or not by the end users).
    - "handling_converter",
    - "auto_emission_factors",
    - "supplychain_assignation",
    - "activities_scheduling",
    - "energies_consumptions",
    - "areas_occupancy",
    - "pollutants_emissions
- Adapt the PAS models to be as generic as possible and as specific, as necessary. A setting file has been designed on purpose to allow ports to choose the specific module of the port they want to activate.
- Develop a user-friendly interface to allow ports to define the inputs of the models.
- Provide a useful visualization of port activities (Gantt View) and energy consumption (instant or cumulative view)
- Provide an NGSI agent that uses PAS outputs to calculate air emissions eKPIs

PAS MODEL EXECUTION

To run the PAS model the following prerequisites are necessary:

- Deploy an NGSI agent for the vessel calls that is compliant with the associated Data Model (https://gitpixel.satrdlab.upv.es/marc.despland/DataModels/src/master/DataModels/VesselCall.md). Once the NGSI agent has been developed, nothing is expected from the end users. The PAS model will then automatically retrieve the vessel call from the Information Hub for the time period defined by the user.


```
{
  "_index": "arh-lts-vesselcall",
  "_type": "doc",
  "_id": "FR_BAS:9454838:20210405",
  "_version": 4,
  "_score": null,
  "_source": {
    "sourceId": "urn:pixel:DataSource:frbod:VesselCall",
    "sourceTypeId": "VesselCall",
    "sourceRegId": "b6f966777d1a4f7ce225b737ab73303d",
    "timestamp": 1624968010380,
    "data": {
      "loading_cargo_type": "E.COPEAUX%2CPLAQ.%2CDECH.BOIS",
      "scheduled_departure_dock": 1625007600000,
      "IMO": "9454838",
      "source": "urn:pixel:DataSource:frbod:VesselCall",
      "loading_berth": "417",
      "ship_type": "SOL.V",
      "journeyid": 20210405,
      "loading_cargo_fiscal_type": "",
      "loading_dangerous": false,
      "unloading_tonnage": 0,
      "arrival_dock": 1624645800000,
      "pixel_etd": 1624827432061,
      "unloading_cargo_fiscal_type": "SOL.V",
      "loading_tonnage": 2500,
      "unloading_cargo_type": "",
      "unloading_agent": "SURSOL",
      "name": "REIDERDIJK",
      "unloading_berth": "417",
      "unloading_dangerous": false,
      "location": {
        "lon": -0.5472,
        "lat": 44.863
      },
      "dataProvider": "http://frbod/vcall",
      "loading_agent": "SURSOL"
    }
```


Figure 23: PAS inputs - vessel calls for GPMB

Push the "setting file" in the Information Hub. This setting file is used to define the timestamp format of vessel call, PAS module to activate. For example, it is used to activate or not the delay of operations based on constraints defined by users (number of machines, area occupancy, …).


```
{
  "_index": "pas_inputs_settings",
  "_type": "_doc",
  "_id": "pas_inputs_settings_with_gantt",
  "_version": 2,
  "_score": 0,
  "_source": {
    "pipeline": [
      "machine_auto_emission_factor",
      "handling_converter",
      "handling_assignment",
      "handling_scheduler",
      "handling_delayer",
      "out_gantt_view",
      "out_energy_consumption",
      "out_pollutant_emission",
      "out_area_occupancy"
    ],
    "handling_converter": {
      "filters": {
        "ET_consistency": true,
        "content_amount": true,
        "content_type": true,
        "handling_direction": true,
        "handling_dock": true,
        "stopover_ETA": true
      },
      "TS_format": "epoch_miliseconde"
    },
    "handling_assignment": {
      "default_SC": false,
      "discart_unassigned": true,
      "restrictions": {
        "amount_max": false,
        "amount_min": false,
        "direction": false,
        "dock": false
      }
    },
```


Figure 24: PAS inputs - Setting file

- Fulfil the PAS forms. For doing this each port has modeled their port activities by describing type of energy used, machine specifications, sequence of operations associated with a type of cargo, areas where the operation happened, … This description of the port activities takes some time and has to be done by port agents with expertise on how the port works. However, it only has to be done once.

The main effort to be done to run the PAS model is to be able to model the port activities. Depending on the level of accuracy needed for the PAS output, the accuracy of the port activities description must be adapted. Once the vessel calls agent is running and once the PAS forms have been fulfilled, the PAS model is able to run in an automatic way. However, if the input data are poor in quality the PAS output will just give some order of magnitude. On the contrary if the PAS forms have a fine level of detail, the PAS outputs will be able to give more realistic results.

SUPPLY CHAIN

+ Add Supply Chain

ID	Category	Actions
S2	Import/export of containers	Edit Delete
S3_1	Import of handled solid bulks - Seeds	Edit Delete
S3_2	Import of handled solid bulks - Wood	Edit Delete
S3_3	Import of handled solid bulks - Coal and coke	Edit Delete
S3_4	Import of handled solid bulks - Crushed glass	Edit Delete
SC3_5	Import of handled solid bulks - Fertilizer	Edit Delete
S3_6	Import of handled solid bulks - Laitiers	Edit Delete
S3_7	Import of handled solid bulks - Clinker	Edit Delete
S3_8	Import of handled solid bulk - Granulat	Edit Delete
S3_9	Import of handled solid bulk - Refractory earths	Edit Delete
S3_10	Import of handled solid bulk - Ores	Edit Delete
S3_11	Import of handled solid bulk - Minerals	Edit Delete
S3_12	Import of handled solid bulk - Salt	Edit Delete
S3_13	Import of handled solid bulk - Other	Edit Delete
S4_1	Export of handled solid bulks - Oil seeds	Edit Delete
S4_2	Export of handled solid bulk - Quartz	Edit Delete
S4_3	Export of handled solid bulk - Tourteaux	Edit Delete

Figure 25: PAS forms - Supply chain description for GPMB

MACHINE

+ Add Machine

ID	Category	Timetable	Actions
Gantry_Crane	Gantry crane	Container	Éditer Effacer
Straddle_Carrier	Other	Container	Éditer Effacer
Truck_Container	Other	Container	Éditer Effacer
Dry_Bulk_Loader	Other	Conventional	Éditer Effacer
Truck_Dry_Bulk	Other	Conventional	Éditer Effacer
Forklift_General_Cargo	Forklift	Conventional	Éditer Effacer
Truck_General_Cargo	Other	Conventional	Éditer Effacer
Electric_Crane	Crane	Conventional	Éditer Effacer

Total 8 10/page 1 Aller à 1

Figure 26: PAS forms - Machines description for THPA

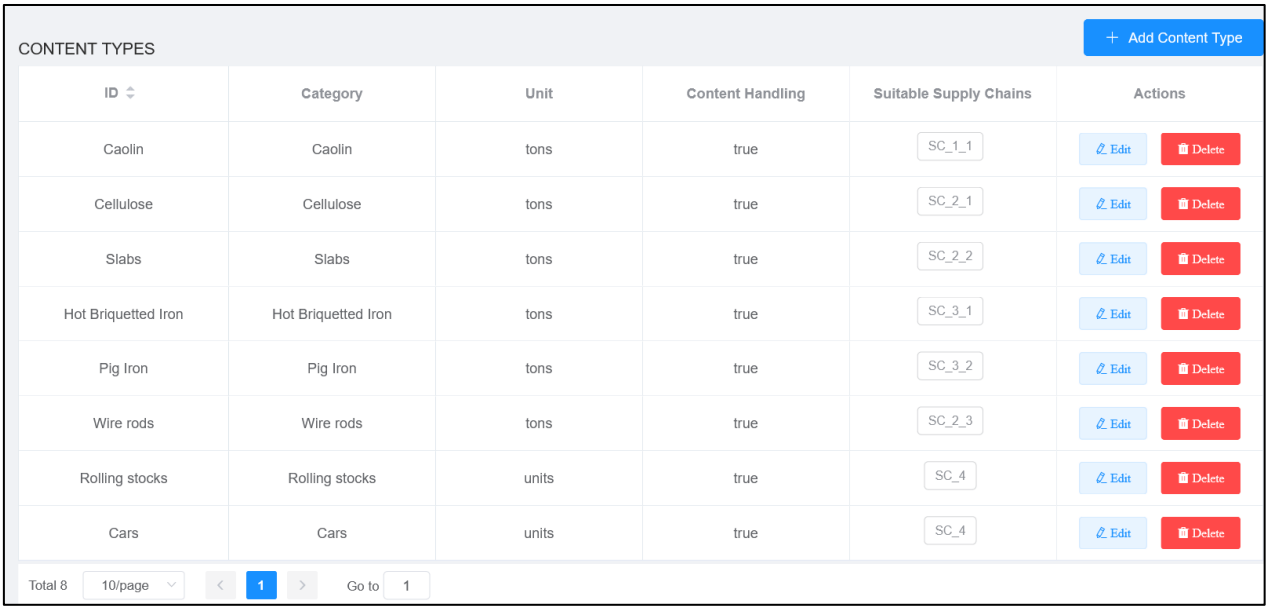


Figure 27: PAS forms - Content types description for Monfalcone

Having these two types of inputs, the PAS models have been run (or scheduled) in each port. A GUI interface has been developed to choose the scenario (period, port parameters and vessel calls list) to use for running the PAS model.

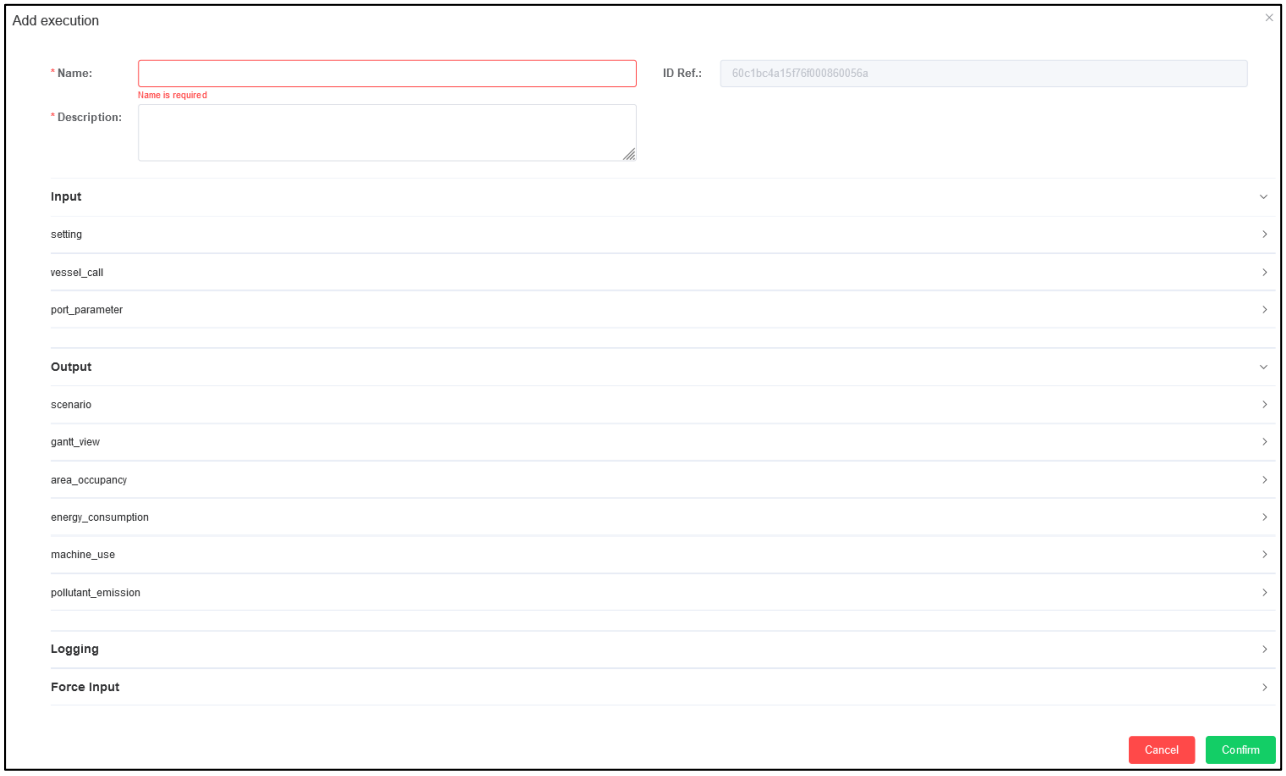


Figure 28: PAS Model execution - GUI to run the PAS model

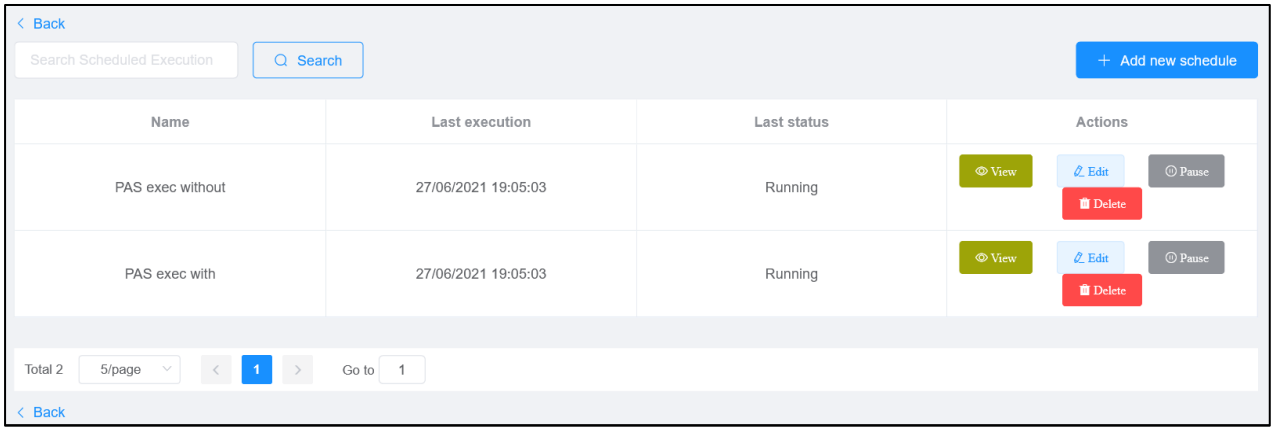


Figure 29: PAS Model execution - Scheduled execution in Monfalcone (with and without constraints due to area occupancy)

```
================================================================
PAS modeling starts
Loading current PAS instance: success (from given argument)

================================================================
INPUTS LOADER STARTS

IH input: setting
A doc_id is provided
Call IH at url http://172.24.1.11:9200, for item setting in index pas_inputs_settings for doc settings_without_delay
Successfully retrived

IH input: vessel_call
Filtering on date range (will be use to filter arrival_dock)
A start time is provided 1619827200000
An end time is provided 1622505600000
Call IH at url http://172.24.1.11:9200, for item vessel_call in index arh-lts-vesselcall for doc None
Successfully retrived

IH input: port_parameter
A doc_id is provided
Call IH at url http://172.24.1.11:9200, for item port_parameter in index pas_input for doc LTTwUnoBTXv-LsH-v7QL
Successfully retrived

All the required items are present and valide.
Number of valid handlings remaining :399
```

Figure 30: PAS Model execution Log

Once the PAS model has been configured and ran, the following outputs are available:

- Instant and cumulative energy used.
- Area occupancy.
- List of handling and operations.
- Emissions quantification.

The sequence of actions from a vessel call to, as an example, a quantification of emissions is as follows:

1. The port agent, terminal operator or the port authority described the different supply chain with their associated machines and specifications. Figure 4 shows the GUI that have been developed to help ports fulfil these data. This work is just done once (and updated by the user as needed with the addition of new equipment procured or the deletion of obsolete machinery). Then the PAS model is configured to be run in an automatic way through the Operational Tools of the PIXEL platforms.

2. A ship approaching a port sends a vessel call and a FAL form (convention on Facilitation of International Maritime Traffic) to the port management information system. Thanks to the PIXEL platform, this vessel call is automatically stored in the PIXEL Information Hub and made available as an input for the PAS model.

3. The PAS model considers the following item to create an ordered list of operations:

    a. The cargo.

    b. The estimated time of arrival.

    c. The port description.

d. The rules and priorities set for the handling of similar cargo.

4. The energy consumption corresponding to the PAS predicted list of operations is calculated by the energy model.

5. With the use of emission factors, the energy consumption calculated is translated to emissions quantification by the PAS.

6. The emissions quantification can be used by the Environmental Pollution Models and by the Port Environmental Index.

The figures below are some examples of how the outcomes of the PAS model can be used: to see and understand the density of workers in a specific area, to simulate the instant (or cumulative) in energy consumption (kWh of electricity, L of diesel, …). Ports will use the PAS outcome in order to better understand how the energy is consumed (when the peak is reached, when the consumption is low, what happened if one fuel machine is changed for an electric one), how many times a machine is used per month, per year, how many people worked in the same area, ...

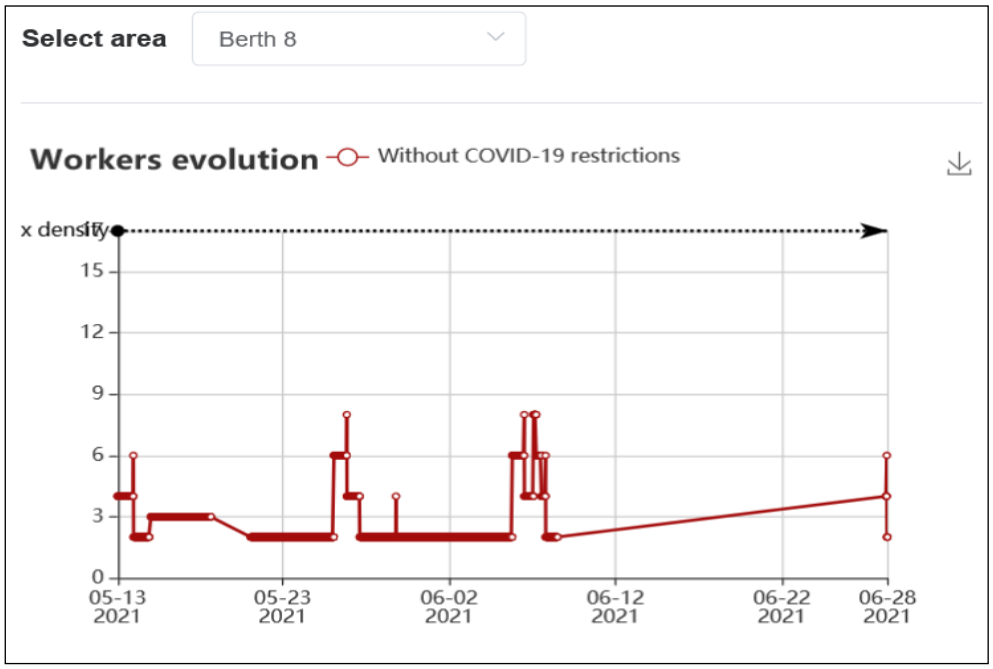


Figure 31: PAS Model output - Area occupancy (number of workers) in Monflacone (Use in the COVID use -case)

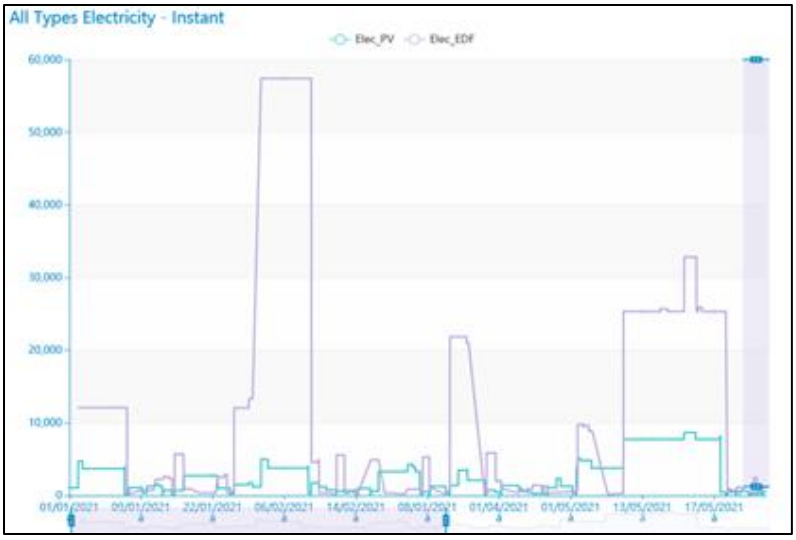


Figure 32: PAS Model output - Instant electricity consumption (kWh) in Monfalcone (Use in the Energy Use Case)

5.2.2. PEI Generalization/Customization

One of the main characteristics (or qualities) of the Port Environmental Index (PEI) is that it should be applicable to any port. In other words, every port, regardless of its size, main purpose and/or port budget, should be able to calculate the PEI if wanted. There are several main issues with this that had to be addressed during the development of the index and its subindices and environmental Key Performance Indicators (eKPIs):

- Chosen indicators had to be representative for all ports.
- Not all ports have the same measurement equipment (sensors) nor they have the same level of automatization – the index had to be flexible enough to account for those differences and to allow for some alternatives to the first-choice options.
- The PEI creation process should consider different acquisition methods for different eKPIs in different ports (i.e., not all ports use the same form for the acquisition of data on wastewater or air pollution for example).
- Related to the two points above, some ports might not have all the required data. In that case, the eKPIs should be approximated using relevant approximation methods and procedures.
- Finally, even if the previous case was not applicable, PEI should consider allowing -at least- a partial calculation of the index. This was (among others) one of the main reasons behind setting a "tree" structure composed of different layers and sub-indices.

In order to assess those aforementioned points, the PEI development procedure had to be conducted extremely cautiously. As described in other Deliverables (most notably D5.4 and D7.2), choice of the indicators is quite a complex process that also has a direct influence on the universality of the PEI as a whole. To avoid as many inconveniences as possible, the following two approaches were used for the identification of the indicators:

1. "Scientific" approach, which consisted of reading the scientific papers and research reports related to the issue and picking out those that seemed to be most reliable and applicable for the purpose of the PEI development. Also, the frequency of various environmental aspects being described as "important" or "significant" in those papers had a major (although not necessarily conclusive) influence on the final decision.
2. "Practical" approach, which consisted of creating the questionnaire for the assessment of the importance of the individual environmental aspects and forwarding it to the four pilot ports (Ports of Bordeaux, Monfalcone, Piraeus and Thessaloniki). Their answers were carefully examined and, together with the information found in the literature, formed the basis for the final decision on the "significance" of individual environmental aspects and their inclusion in the PEI.
3. "Scalability" approach, which consisted of performing a prospective exercise: how the functionalities designed for PEI should be depicted so that they will be most useful for external ports and for future activities. A part of this rationale translated into the technological separation (in two main blocks) of

the software of the whole PEI system. This separation (which details can be found in D5.3) also responded to a logical differentiation very aligned with the trends in the digitalisation of the maritime transport sector.

Like written before, the process was described in more detail in other Deliverables, but it is important to note that both ports and scientific research were consulted to have the most representative and the most universally applicable set of parameters as possible. Especially remarkable here was the action for completing a Budget Allocation Method for the Reliability Rating (= IoT Readiness) index. In short, BAM relies on the vision of experts in one field to "weigh" how much "worthy" are the different options. In PIXEL, this activity was deemed necessary for establishing a "comparative importance" to the different options of data retrieval. For the sake of clarity, let the reader conceive an example: retrieving the amount of wastewater from a grid of sensors installed in the pipes network is not the same (way better) as typing through a form the values estimated of such disposal (way worse). However, the quantification of "how good is one in comparison to the other" is not trivial and it gets more complicated when there are more options in between. The objective of BAM in PIXEL was to specifically establish that quantitative comparison (per eKPI). For doing so, a series of experts in different fields were called upon to provide their views on the matter using precisely classic BAM methodology, having to assign a total of 15 points dispatched to the different "retrieval options" of the data that is needed to feed those eKPIs[2] .

Other obstacles were found during the execution, which led the team to take decisions that were always approached from flexibility and scalability maximisation perspective. For example, one of the first stumbling blocks was the inclusion/exclusion of odour pollution as a significant environmental aspect, since (for most port types) it is a minor (or even negligible) environmental aspects, while for other port types (fishing ports and the ports in which there is a lot of livestock/cattle transport), it can be among the most important aspects. In order to solve this issue, it was decided to make this one aspect an "optional" aspect – it is used in the calculation of the PEI only for those ports that are severely affected by this type of pollution. On another note, it is true that other (sometimes positive) parameters were also in consideration but needed to be discarded due to lack of broadness. For instance, the Port of Bordeaux (partner GPMB) has been investing resources over the last few years on shifting to more sustainable energy sources; photovoltaic panels are starting to be used and Liquified Natural Gas (LNG)-propelled boats are now in charge of draining the river bottom, just to mention some. Also, specific programs for studying and improving biodiversity were initialised in ports during the lifetime of the task. However, all the previous was considered either too specific or too diluted to be considered of relevance for a wide, short-term evaluation tool such as the PEI.

Other considerations and steps taken to make the PEI calculation possible in any port were much more subtle, but still had a significant influence on the final product. The first issue was the issue of available measurement equipment (sensors) in ports. Based on the experience with pilot ports, it could be concluded that most of the ports do not possess all the sensors (noise and light sensors, as well as electrical noses for odour pollution) that were among the desirable equipment for the calculation of the Index. To overcome this barrier, the Port Environmental Index allows a port to indicate the way how the data is being retrieved (among different options) and proceeds with the calculation. While the use of the sensors is highly recommendable and advisable, all three sensors have alternatives:

- If no measurements conducted by using noise sensors is available, it is possible to use historical measurement data that most ports are obliged to do in some regulated periodical intervals.
- Light pollution can be approximated using the online "light pollution maps" that show levels of light in chosen areas
- Electrical noses (odour pollution sensors) can be replaced by subjective assessment done by a designated person or by conducting a periodical survey among the affected people. Admittedly, both alternatives have significant shortcomings – subjectivity if the port designates a single person and high inconvenience that polling a significant number of people in short periods cause – but they both offer a possible (and passable) alternative for the ports that are unavailable to acquire the required sensor.

[2] This methodology followed all Good practices provided by WP1 of the project

5.3. Cross-pollinated generalization: PAS and PEI

Another remarkable outcome of the task T7.6 has been the linkage and cross-utilisation between two results of PIXEL. Port Activity Scenario (PAS) - explained two sections above - was developed as a global generalization approach assimilable to a small-scale "Digital Twin" of a maritime terminal. On the other hand, the Port Environmental Index (PEI) was built considering the variety of port types, origins, data-gathering methods, mathematical tools, etc.

Drawing from the previous, during the course of the task it was discovered that the outcomes of the former could be used for feeding the latter. In particular, the idea went as follows:

- (i) PAS is able to "simulate" the operations of one terminal in a certain period of time using the incoming vessels and the supply chains' definition as input. The result consists in a "forecasting" of the operations to be conducted at each moment, including the machinery in use, the energy consumed, the occupancy of the areas (see COVID pilot).
- (ii) PAS is also able to estimate (see above the "emissions quantifications") the emissions to be produced by each machinery in the terminal each unit of time (depending on the timestamp of operations).
- (iii) The previous could be used as input for the PEI, as one of the "subindices" of the index is, precisely, the emissions to the air attributable to a maritime terminal (thus, allocable to the machinery used in the operations of the terminal).

However, the following barriers were encountered that prevented the data to be used "as-is":

(i) PEI needs all the data to be fed to the platform (associated index in the Information Hub) in the form of eKPI, following the specific format that was defined on purpose. Therefore, a conversion was needed to shift from one format (PAS output) to the other (PEI-compliant).

(ii) Timing/periodicity of timestamps. The outputs of the PAS are generated depending on the operations, but can be split into (e.g.,) hourly records. However, the PEI eKPIs must follow a clear periodicity (daily, weekly, fortnightly, monthly, quarterly, or yearly are the commonly used in the project). In addition, this may depend on how the rest of eKPIs is included in the platform.

(iii) Last, but probably the most important: whereas PAS is a prospective model (i.e., forecasts what should happen in the terminal in the future), the PEI is a retrospective model (i.e., is based in past data and provides the result once that period has passed). This posed a series of challenges in order to consolidate the data of "predictions for the future" to be used as "certainties of the past" so that the results of the PAS could be properly used in the PEI.

For coping with the previous, the team of T7.6 decided to develop an NGSI agent that would (first) store the results of consecutive executions of the PAS, (second) would store the maximum value of emissions for a timeframe and (third) would, with a certain frequency, generate the eKPI entities and (fourth) send them to the Context Broker of the platform.

The agent was developed and was deployed in the ports. Documentation of the agent is available here:

Although it is recommended to check the documentation for a full reference of the agent, here below the most relevant aspects can be found summarised:

- The agent was developed to cover the polluting factors: CO, CO2, NOx, HC, PM10, PM2.5 and S02.
- The process of obtaining the results will be carried out as shown in the following diagram:

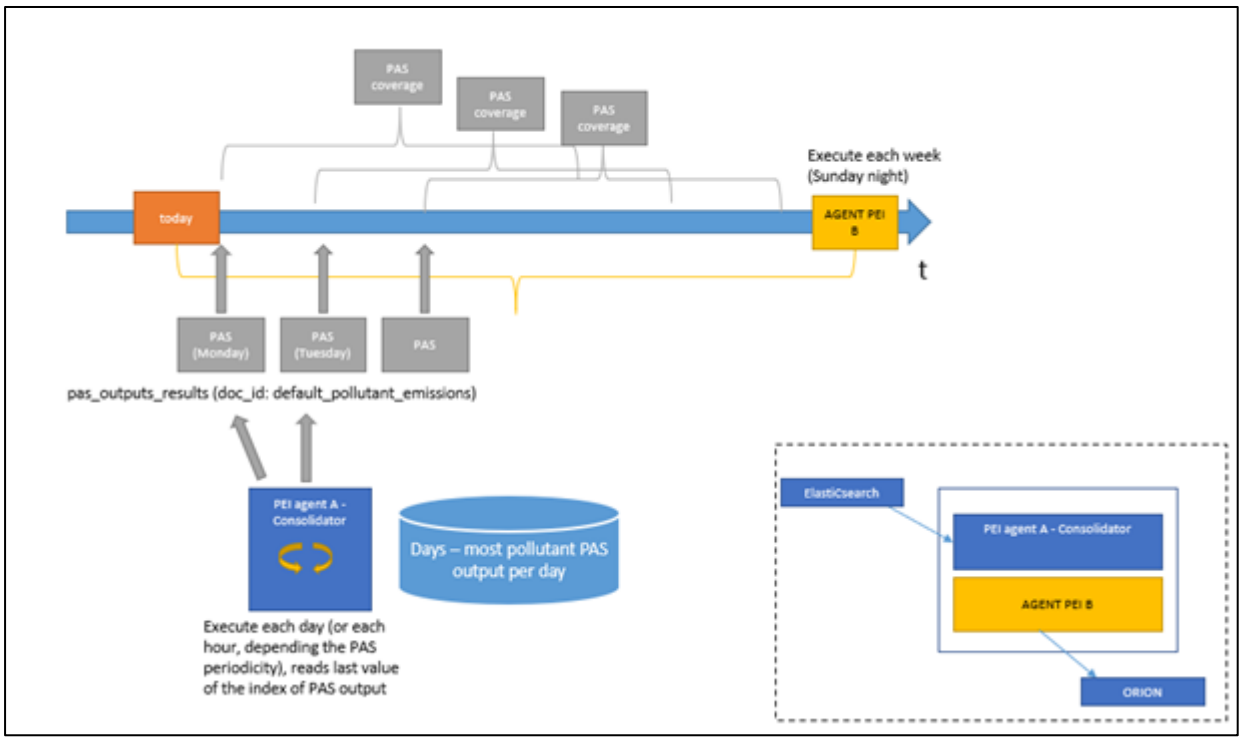


Figure 33: PAS-PEI Process description

- To implement the above scheme, the agent was composed of two sections:
  - **First section:** Analyses the value of each pollutant from the last PAS output. The records are expected to be timestamped on future dates (drawing from the nature of the PAS model, which aims at simulating port terminal activities forward), then some storage and consolidation must be performed. Here, the agent stores in a JSON file DataExtracted.json) the forecasted values of each pollutant per day according to the last PAS simulation. On the following iterations, this file is consulted (day by day) in case the new predictions should update the pollution estimated for a certain date, overriding the entry in case the value is greater than the currently registered value.
  - **Second section:** For each week (actually, on Sundays - 22h30), it compiles the data of the past week (from Monday to Sunday) existing on DataExtracted.json, sums it up, normalises the resulting value to the total tonnage operated by the port during that week and obtains the value of all pollutants' eKPIs (units = kg/ton).
  - Finally, the agent creates the NGSI-compliant entity and forwards it to the DAL (ORION Context Broker) to update each entity.

The agent was developed with the purpose to be port-independent, can be applied as long as PAS and PEI exist in the PIXEL instance of the port and proper (enough) data is stored in the expected indices of the Information Hub. Technically, that generalization is obtained by the creation of a file that contains some constants which can be changed to establish the necessary values of the data source and the necessary values of the storage where data will be saved after its processing, without the necessity to modify any part of the agent.

Same as for the rest of NGSI agents in PIXEL, for the agent to be used, the Docker image must be built (that will encapsulate all dependencies of the agent) using Docker file that includes all the necessities that the process needs to make the required calculations obtain the results.

By measuring these values and carrying out the aforementioned calculations, it is intended to control and reduce pollution as much as possible by observing the results obtained from the emission of polluting elements generated during operations carried out in ports.

6. Lessons Learnt

6.1. Energy Management Trial - GPMB

6.1.1. Technical

The use of GIT and docker-compose makes it much easier for the operability of the platform. That way, it is possible to quickly update the platform several times without big issues.

To enable this, it was also important to separate the local configuration file from the GIT repository in order to prevent accidents. The secret folder was once rewritten by accident. It contains all the technical passwords used on the platform. Thanks to having backups of the data, the problem was solved by restoring the system from the copy.

The 2 servers that constitute the pilot, allow SSH connections for administration purposes. Since they are hosted by an external provider, this access is open on the Internet. The pilot platform faced one SSH attack using a botnet that crashed the server (too much connection attempt). It was necessary to restart the server and change the listening port of the daemon SSH. As the connection is only possible with a RSA key, the attacker cannot enter the platform. For this reason, it is always better to disable external SSH access and configure a third VM as a bastion. But it was not possible with this provider.

One of the main technical issues that arose during the integration of the PIXEL platform in GPMB was related to the availability of the data. Indeed, the data exists but it could be difficult to understand how to retrieve it. Another difficulty was related with the deployment of a real end-to-end scenario for the energy consumption simulation.

6.1.2. End User (Ports)

Thanks to the work done in WP4, PIXEL is able to provide for each stopover the vertical handling equipment mobilized. Indeed, the 5 specific logistics chains initially described in WP3 (D3.3) are now fully configured in PIXEL to determine as soon as the stopover is announced, depending on the nature and quantity of goods, each equipment required, as well as the duration of use and power used.

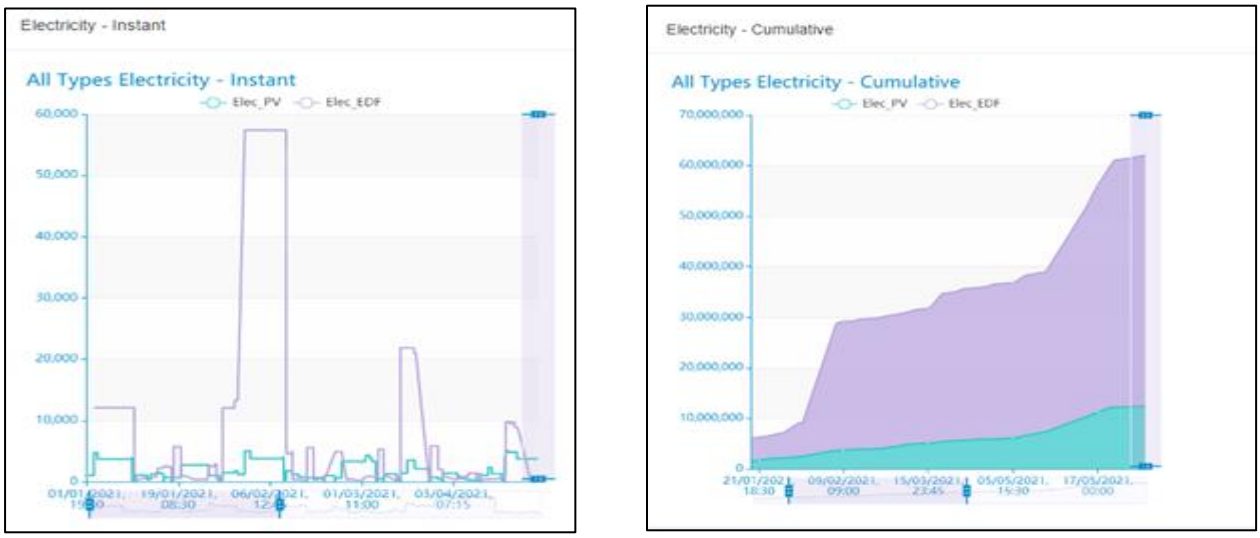


Figure 34: electricity demand and photovoltaic production (Left ; instant – Right ; cumulative)

The forecast of energy consumption, in particular electricity, enables the exploration of self-consumption of green energy. Indeed, PIXEL now describes the energy demand and the local photovoltaic production (Elec PV in the graph above). It now makes it possible to study the necessary dimensioning of electricity production and storage to meet the energy demand.

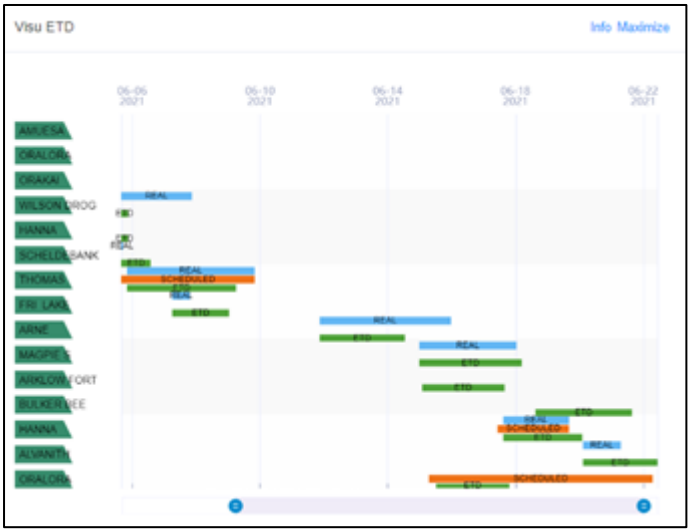


Figure 35: forecast of vessel calls

These results are important to elaborate the next strategy of GPMB to reduce the energy demand through management practices and technologies;

- Technologies improvement
  - Calculate ROI to produce and store green electricity.
  - The kind of technology needed for storage depends on ROI objectives.
  - Prepare works needed for installation of PV local production ; the installation studies on the roofs of the hangars of the Bassens terminal has just started, for a start of green electricity production in 2022.
- New practices
  - Follow PIXEL indicators to maximize energy conservation and operational productivity.
  - Identify the most energy-intensive logistics chains to prioritize the actions to be taken to reduce consumption by renewing equipment, or even rethink certain parts of the logistics chain.

GPMB is now interested in having a detailed balance (hour by hour) of terminal demand and PV production electricity to estimate the quantity of electricity produced, stored, used by terminal, and the overall balance of electricity purchased and sold to and from the network. These elements make it possible to optimize the energy efficiency of self-consumption and to anticipate future traffic trends.

6.2. Intermodal Transport Trial - Monfalcone

6.2.1. Technical

The PIXEL infrastructure represents, from the point of view of a modeller, a rich and powerful platform. More specifically each model can be fed with data acquired from different data sources (by means of NSGI agents and / or manually provided data) and, at the same, from the output of other models provided by the different partners (e.g.: data provided by Traffic Prediction Algorithm developed by XLAB, vessel calls data shared between PAS and Intermodal Transport Model).

The ability to integrate in a unique environment such a large set of tools represent one of the key features of the PIXEL infrastructure, allowing IoT devices to be proficiently used as data source to optimize ports' related operations.

At the same time, while providing undoubtedly several advantages, such a complex infrastructure can lead to several issues which can affect the development of new models in terms of both time and cost. Data provided by NSGI agents need, in fact, to be represented with a strict cohesion with FIWARE Data Models. Such strong tiding may lead to minor issues during data processing (missing fields when data is collected from the DAL and sent to the IH) and, moreover, a significant overhead in terms of data required to represent information (e.g.: by using JSON documents with a complex schema to represent simple data coming from an IoT sensor such as the HOPU station or a Smart Camera).

The adoption of existing FIWARE Data Models represents another potential limitation of the platform, in fact, each schema used to store data needs to be defined and known in advance to support the data processing from sources to IH.

Another issue which could impact on the development of new models is represented by the difficulties in reading data from the IH. Queries need to be expressed in a specific dialect (the Elastic Search Query language) different from the relational DBMS standard tools (such as SQL syntax). Moreover, data extracted from IH need to be properly deserialized to be used by the model. Such approach increases the dependencies (in terms of data structure and schema) between the different modules of the infrastructure and the way models interact with each other.

Operation Tools represent a powerful tool, allowing each model developer to work independently from the context, by using its own Docker container (and context) to define the content of the model. Each model developer can work on the models in an autonomous way, being free to choose the preferred language and technological stack.

The dashboard capabilities exposed by the PIXEL platform are complete, effective, and easy to use and understand by the users, providing a practical tool to analyse results from each model and plan future execution of each model.

6.2.2. End User

The PIXEL project succeeded in developing a comprehensive platform for small and medium ports which, generally, are economic ecosystems that are rarely supported by any operational IT systems, almost never based on IOT data.

Taking in consideration these premises, the platform and models developed thanks to PIXEL appear as a great novelty for the Port of Monfalcone. The Port Community is used to work with institutional systems developed to carry out formal practices in the fields of customs and maritime declarations but never had the chance to take advantage of an operative system capable to assist it in operative activities or for general purpose scopes.

In these fields the models and tools developed for the "Intermodal Transport Trial" are a great opportunity to maximize the value of all the information already available online, such as the ones related to the port's sailing list, or the new ones that were implemented to feed the platform, coming from external sensors, or manually provided through the dedicated forms provided by PIXEL system.

Nowadays, the Port is still in the evaluating phase of the platform, but it already received some interesting internal and external feedback in relation to the platform capabilities and further development potential. This

feedback refers to both the capabilities of the model developed in the "Intermodal Transport Trial" as well as with reference to other models developed in Piraeus, Thessaloniki and Bordeaux ports that can be extended in the Port of Monfalcone.

Furthermore, SDAG considers the platform development also a useful tool to handle the freight traffic in an interoperable and shared way. PIXEL platform is the first system that gathers data coming from different regional logistic stakeholders and supply chain actors, at the moment to fulfil the requirements of the Port of Monfalcone. However, considering the potentialities of the platform and the capacity of integrating in a single environment a large set of different tools coming from different sources, it can be possibly used to support regional logistic strategy, both at an operational level both to support public authorities and decision-makers to detect the most efficient and effective logistic models to be implemented on the territory. For this specific functionality also, predictive algorithms are considered very useful.

The platform itself seems a "user friendly" infrastructure that, after a first set up to be made by IT department technicians, can serve well the managerial and operative staff which can benefit from an "Overview" page where all the data and models processed by the platform can be easily readable by the end user.

Another valuable output is that "external" users can benefit from the models run in the platform and data processed externally, such as the one related to the Intermodal Transport Trial, that is available in the institutional website of the Special Agency for the Port of Monfalcone

- http://www.porto.monfalcone.gorizia.it/eng/permesso_acc.asp;
- http://www.porto.monfalcone.gorizia.it/permesso_acc.asp

Figure 36: Intermodal Transport Trial online banner

The capability of the platform to collect, merge and process data coming from different sources, coming both by the port ecosystem as well as from other subjects (general services providers, road, and environment monitoring authorities, etc.), can extend the PIXEL scopes to a wide range of future possibilities: linking ports, inland ports and transport infrastructures is a key concept in modern logistic. Intermodality is based, more than over an adequate transport infrastructure, in the capability to process information coming from different subjects (almost already available locally) and model them to obtain valuable data to perform efficiently, greener and with lower costs. With the Intermodal Transport Trial, the Port of Monfalcone started an IT approach to these concepts and set the baseline for future development of the platform over other logistic partners of the Port.

6.3. Port-City Integration Trial - THPA

6.3.1. Technical

PIXEL as a platform offers an online interface which is modular and highly customizable. With the use of the NGSI agents, different sources can be integrated in the platform, to be used by the models.

One limitation faced during the insertion of data for the PAS model, was that the PAS forms allow only for one instance of each machinery to be used in one operation of a supply chain. There are two ways to lift this impediment:

1. By creating virtual machines which will be a bundle of n identical machines. Then, the bundle of the machines will be inserted in an operation. The throughput, energy consumption and power of the virtual machine will be the sum of the constituting machines.
2. By creating identical operations that will be running in parallel within a supply chain. This solution requires that each operation will end after the amount of cargo that corresponds to its percentage in relation with other identical operations has been processed.

With ThPA, the solution no. 2 has been selected.

Regarding the amount of cargo processed which is required for an operation to end in general, it is noted that this always corresponds to the total amount of cargo handled and not to the amount of cargo fed to this operation by the previous operation.

6.3.2. End User (Ports)

PIXEL managed to deliver a platform that can collect and processing data originating from various sources (IoT sensors, manual inputs, historical data) and through powerful analytics, to provide a valuable tool for ThPA S.A., but also for every small and medium sized port. The knowledge and modeling of the supply chains and relative port activities provided the basis to identify and visualize the energy sources/consumption, local emissions of pollutants but also to estimate the flow of cargoes entering or leaving the port. Finally, data such as seasonality and weather were considered to improve the platform's accuracy.

During the installation phase, some problems were encountered; due to the choice of a site for the installation of the light/noise sensor, special works needed to take place, with people from different departments collaborating (IT staff, welders, lifting machinery operators), which caused a significant delay in the initial plan of the hardware installation. After the hardware installation, the network connectivity of the sensor turned out to be much more complex than initially evaluated. Finally, the decision was made to find an alternative site for the sensor, with a different network setup, thus pushing the 'go live' date further back. One final issue that became apparent at a later stage, was the availability of data. It turned out that people responsible for data entry (to the application that feeds the agents) may occasionally delay registering vessel calls, thus creating problems for the agents that rely on them and impairing the platform's overall performance.

After the installation of the PIXEL platform in the Port of Thessaloniki, the evaluation phase commenced. In the case of Thessaloniki, the scenarios were mostly concentrated in traffic conditions and possible congestion in the surrounding area, as well as the environmental impact of relative port operations. For this purpose, a Prediction algorithm was implemented, which allows the estimation of traffic at both ThPA gates, as well as noise and air pollution models.

The platform itself is considered as user-friendly, since it provides a comprehensible array of models, visualizations and outcomes, while it provides the user with various choices. In the following two pictures the interface is presented for the selection of sources and type of visualization.

Figure 37: Source and type of model visualization

Finally, a valuable aspect of the platform is the scheduling of models' executions; that way the user may have the "in-time" information that is required in everyday tasks. However, the overall value of the PIXEL product is that it already has the proper architecture that can easily support more IoT sensors, as well as the possibility to use the models that were implemented for the rest of the ports in the consortium.

The PIXEL platform is considered as rather valuable in the case of ThPA S.A., since with the estimation for a specific upcoming timeframe, in terms of traffic the gate managers can visualize in various ways the available results and establish different thresholds for each gate, while the Environmental Department, to assure compliance with ThPA SA Environmental policy, from all stakeholders in its premises.

6.4. Port-City Integration Trial - PPA

6.4.1. Technical

PIXEL as a platform offers an online interface which is modular and highly customizable. With the use of the NGSI agents, different sources can be integrated in the platform, to be used by the models.

One limitation faced during the insertion of data for the PAS model, was that the PAS forms allow only for one instance of each machinery to be used in one operation of a supply chain. There are two ways to lift this impediment:

1- By creating virtual machines which will be a bundle of n identical machines. Then, the bundle of the machines will be inserted in an operation. The throughputs, energy consumption and power of the virtual machine will be the sum of the constituting machines.
2- By creating identical operations that will be running in parallel within a supply chain. This solution requires that each operation will end after the amount of cargo that corresponds to its percentage in relation with other identical operations has been processed.

With PPA, the solution no. 1 has been selected.

Regarding the amount of cargo processed which is required for an operation to end in general, it is noted that this always corresponds to the total amount of cargo handled and not to the amount of cargo fed to this operation by the previous operation.

The data gathering from some sources has been a problem since the beginning of the pilot for several reasons:

- The format of some data sources was not compatible with the NGSI framework used. For instance, the acquisition of data form xlsx has been a challenge, because the data stored in these files was not always well processed and it was needed to use several libraries until a good one was found.
- Some data sources have not been available until the end of the pilot development. In some cases, the data were not available digitally at it was to be created manually by the port, in other cases the port had to subscribe to a paid service to obtain that data.

In the end, it was possible to address all the issues with data acquisition and all data were successfully integrated in the pilot.

On the other hand, during the configuration of the models some problems arises:

- The traffic model had to be adapted in order to run with the data integrated in the platform. In the beginning it was designed to work with specific data, and in the end the data was not exactly the same and it was to be adapted.
- The nose and the air pollution models were not able to be integrated in the platform and they were executed outside the platform.

6.4.2. End User (Ports)

The PIXEL platform is a tool capable of monitoring and controlling various environmental aspects to the PIXEL project. Thanks to the PIXEL platform, PPA may not need to have several isolated SW/HW solutions to control different environmental parameters. In the PPA pilot, the platform is oriented to monitor the impact of the port activities on the city and to anticipate/predict serious environmental impacts, as well as traffic problems for both citizens and tourists.

In relation to the data gathering, several data sources provided by PPA have been available to the technical partners for their integration to the PIXEL platform. Through the platform dashboard, visualisations can be created to display such information in real time. Currently the platform displays information on real-time air pollution and weather conditions.

Regarding the models, two models have been integrated in the Pilot:

- The Noise model, which is based on a commercial sound meter that presently is not integrated in the platform. To overcome this barrier and to be able to have useful information in relation to the noise produced by the port activity, the results of the model are integrated into the platform through intuitive and easy to understand heat maps. The model can potentially be run several times a year, simulating a pessimistic situation (port with 100% activity) and an optimistic situation (port with low activity). In this way, the port could be aware of the potential impact produced in the best and worst case scenarios.
- The model to be used for the air impact calculation is not integrated into the platform for technical reasons explained in D7.2. This model allows to calculate the effect of port activity on air quality using as input meteorological conditions, port topography and port activity. The first two are obtained by the model in real time from external services, while the port activity is currently entered manually. The current integration allows the port to create the simulations needed based on past, current or expected port activity. The input of this information is costly and has to be done manually while in the future it would be recommended that this information is obtained from the vessel calls and other developments used in Pixel for the calculation of the PEI.

The PEI, PAS and Traffic congestion models are partially working in a standalone fashion. By the time of closing this deliverable, the first results are appearing, however those functionalities are not available for PPA through the platform. The current status of the platform may allow PPA to be optimistic on the reliability of the results and the accuracy of the predictions.

The interpretation of these results and the analysis of their technical and business/socioeconomic impact in the port will be a matter of WP8 and t will be reported in the proper deliverables by M41. With regards to the use of the platform, PIXEL shows a clear potential although it lacks usability for the non-technical end users like certain port stakeholders.

7. Conclusions

This document has explained the activities carried out, during the execution of the Task 7.6, in order to apply models in the pilots that were not initially planned. Due to the current pandemic situation that has been hitting the world for more than a year now, the members of the PIXEL project thought of helping ports to minimise the spread of the virus. For this reason, the COVID Model was conceived and developed. This new model can help ports in complying with the anti-COVID-19 measures to ensure the safety of workers by monitoring the number of people in different areas of the port depending on the current and expected port activities.

In addition to the COVID model, the energy model and its various variants that were conceived for Bordeaux port, due to the relation between the PAS model and the energy model, it was possible to partially apply the energy model in all pilots.

The integration of the PIXEL platform components, data sources and models has been more costly than initially estimated, because the design and implementation of the platform should consider the requirement of portability and genericity of the platform and models to be applicable to other ports. This document explains the mechanisms/processing used to realise the integration of the model and the data sources. These integrations are based in the concept of microservices based on docker that independize the models from the programming language utilized for developing the models and data sources acquisition.

Apart from designing a generic platform, there are two models "PAS and PEI" that have been used in the four pilots. As all ports do not have the same needs and data, these models have had to be adapted and made more flexible to suit the different ports. Thanks to these adaptations they are generic enough to be successfully applied in other ports.

The global lessons learnt during the execution of the pilots can be summarized as follows:

- Lack of clear documentation of some components of the platform and models.
- Some usability problems related with the installation of models and data models.
- The installation process of the platform should be made easier, as it is costly and not very intuitive as it assumes a certain knowledge that is not always available.
- The visualisations of the models have had to be redone several times, as they are coupled with the model outputs and when these are changed, the visualisations have to be adapted.
- The data acquisition process has been much more costly than expected for several reasons: unclear data requirements, incorrect data format, frequency of data retrieval and so on.
- Some models could not be fully integrated into the platform and are run in isolation. Nevertheless, an effort has been made to integrate the data produced by these models, so the results of the execution can be displayed on the dashboard as another result.
- The models developed in the project for the different pilots accomplish their purpose of helping the port to improve and monitor its environmental impact.

As a global conclusion. After the pilots have been carried out. The PIXEL platform could be seen as a unique tool capable of monitoring and controlling all environmental aspects of concern to the ports. Given its extensibility, the PIXEL platform is capable of integrating both current and future environmental needs.

During the evaluation phase, that is being carried out in Work Package 8, the ports will have the opportunity to deeply evaluate the models and the platform and complete the lessons learnt.

Annex 1. JSON format (publication)

In order to better understand the JSON format (instead of providing the schema), the basic **pingcount** model is used as an example, available at *https://gitpixel.satrdlab.upv.es/benmomo/ngsi-agents-thpa/src/master/thpa-model-pingcount*. The model is able to count the number of pings of a given set of elements of type Ping. This is a PIXEL data model generated by a basic PingTest NGSI agent (available at *https://gitpixel.satrdlab.upv.es/benmomo/ngsi-agents-thpa/src/master/thpa-ping*)

The JSON format is as follows:


```
{
"name":"pingcount",
"version":"0.1",
"description":"PingCount model. It checks for incoming pings from an NGSI Ping Agent and counts them within a timeframe. It serves as documentation example",
"type":"model",
"category":"ping",
"supportSubscription":false,
"supportExecSync":false,
"supportExecAsync":true,
"system":{
  "connectors":[
  {       "type":"ih-api",
          "description": "Needed connection data to reach the Information Hub API in the PIXEL platform. This API is typically used by models/PA to get data",
          "options":[
          { "name":"url", "type":"string", "pattern":null,
          "description":"URL string representing the API to reach the IH. If a value is given in getInfo.json, it represents a default value",
          "required": true,
          "value": "http://172.24.1.17:8080/archivingSystem/extractor/v1"
          },
          {"name":"user", "type":"string", "pattern": null,
          "description":"Credentials (user) if IH requires authentication. Currently not needed (future use)",
          "required": false,
          "value":null
          },
          {"name":"password",  "type":"string", "pattern": null,
          "description":"Credentials (password) if IH requires authentication. Currently not needed (future use)",
          "required": false,
          "value":null
          }
          ]
  },
  {       "type":"es-api",
          "description": "Needed connection data to reach the Elasticsearch API in the PIXEL platform.This API is typically used by models/PA to store (result) data",
          "options":[
          {
          "name":"url", "type":"string",      "pattern": null,
          "description":"URL string representing the API to reach Elasticsearch. If a value is given in getInfo.json, it represents a default value",
          "required": true,
          "value": "http://172.24.1.11:9200"
          },
          { "name":"user", "type":"string","pattern": null,
          "description":"credentials (user) if Elastic requires authentication. Currently not needed (future use)",
          "required": false,
          "value":null
          },
          {"name":"password", "type":"string",  "pattern": null,
          "description":"credentials (password) if Elastic requires authentication. Currently not needed (future use)",
          "required": false,
          "value":null
          } ]
  }]
},
"input":[
  {"name":"pingset",
  "type":"[urn:pixel:DataSource:Ping]",
```

```
      "description":"array of JSON documents of type urn:pixel:DataSource:Ping. Additional info to retrieve the data is given in 'options' array",
      "supportedConnectors": ["ih-api"],
      "metadata":{},
      "required":true,
      "force input": true,
      "options":[
            {"name":"sourceId",   "type":"string", "value": "ping",
            "description":"sourceId (mapped to Index) within the IH where the NGSI generated pings are stored. If a value is given in getInfo.json, it represents a default value",
            "pattern":null,
            "metadata":{},
            "required": true
            },
            {"name":"from", "type":"date-time", "value": "",
            "description":". From timestamp. ISO8601 format. If a value is given in getInfo.json, it represents a default value",
            "pattern":null,
            "metadata":{},
            "required": true
            },
            {
            "name":"to", "type":"date-time", "value": "",
            "description":". To timestamp. ISO8601 format. If a value is given in getInfo.json, it represents a default value",
            "pattern":null,
            "metadata":{},
            "required": true
            }
      ]}

  ],
  "output":[
      { "name":"output",
      "type":"urn:pixel:DataModelResult:PingCount",
      "required":true,
      "description":"output provided by the PingCount execution",
      "supportedConnectors": ["es-api"],
      "metadata:":{},
      "options":[
            { "name": "es_index", "type": "string",
            "description": " Elasticsearch index to store the result  of the execution. Typically, models will store output under index models-output-<name>",
            "pattern": null,
            "required": false,
            "value": "models-output-pingcount"
            }
      ] }
  ],
  "logging": [
      {     "name": "ping-logging", "type": "pixel-logging-format", "supportedConnectors": ["es-api"],
            "description": "activity logging for the pingCount model. id (id_execution, idRef (id_model) are given by the OT at invocation time",
            "required": true,
            "verbose": null,
            "metadata": {},
            "options":[
            {
            "name": "es_index",   "type": "string",
            "description": " Elasticsearch index to store the logs of the execution (start, end, error). All models will store output under same index",
            "pattern": null,
            "required": false,
            "value": "models-logging"
            }
            ]
      }
  ]
}
```


The description of each property is provided below:

- **name (mandatory)**: this is the name of your model (string).
- **version (mandatory)**: this is the version of your model (string).
- **description (optional)**: this is the description of your model (string).

- **type (mandatory)**: this is the type of your model string). Two possible values: *model* or *pa*, the latter one for predictive algorithms. It is just a convention as PIXEL differentiates between models and predictive algorithms.
- **category (mandatory)**: this is the category of your model (string). No specific list of categories has been described in PIXEL, but some have been pre-identified according to the models developed in PIXEL, such as *environmental, traffic, PAS*, etc.
- **supportSubscription (mandatory)**: indicates whether your model supports subscription mode while execution (boolean). This is the case of some predictive algorithms developed in PIXEL.
- **supportExecSync (mandatory)**: indicates whether your model supports synchronous mode while execution (boolean). This is not the case for the models developed in PIXEL, but it could be useful in the future.
- **supportExecASync (mandatory)**: indicates whether your model supports asynchronous mode while execution (boolean). This is the typical case for the models developed in PIXEL. There is no waiting time at invocation and the outputs are stored in the IH as the models end executing.
- **system (mandatory)**: this is an element intended to encompass general information of the model. Currently it includes one single element (array) called *connectors*, which will be described later.
- **input (mandatory)**: this is an array intended to encompass all needed inputs. The input array will be described later. In the example there is only one single input element, but another model might need more inputs.
- **output (mandatory)**: this is an array intended to encompass all needed outputs. The output array will be described later. Usually, there will be only one output element, which can be mapped as a JSON document stored in a database (IH in PIXEL)
- **logging (mandatory)**: this is an array intended to encompass all needed logging mechanisms. The logging array will be described later. Typically, there will be only one logging element, which generates all logging info as JSON documents to be stored in a database (IH in PIXEL)

Now let us deep into the different elements describing the details.

The **connectors** element defines the ability of a model to connect to different services (endpoints) to get data. It can be open data APIs, databases, etc. In PIXEL, there are basically two connectors that need to be supported natively: (i) the Information Hub *ih-api* connector and (ii) the Elasticsearch *es-api* connector. A model will typically get information from the IH via the *ih-api* and store the data in the IG via de *es-pi*. The way they are defined follow a common format, as you can appreciate in the JSON example. The different fields are:

- **type (mandatory)**: this is the supported type (string). Typically *ih-api* or *es-api*
- **description (optional)**: this is the description of this element (string).
- **options (mandatory)**: this is an array of elements containing the different items to access the service endpoint. The elements given in the example refer to the URL endpoint and credentials (if needed). Let us focus only on the first element, as the format is the same for the others:
- **name (mandatory)**: name of the element.
- **type (mandatory)**: type of the element. For URLs it is a string.
- **pattern (optional)**: Allows to specify if the string has to follow a specific pattern. It will allow to pre-check if the string is URL-compliant before launching a model. This field is only useful for the Dashboard.
- **description (optional)**: description of the element.
- **required (mandatory)**: indicates if this element is mandatory to build the connector (boolean). This means, in case it is not required, this element might not appear in the JSON instance. This field is useful for the Dashboard, so that it will block (or not) the user unless this element is given before letting him/her execute the model.
- **value (optional)**: default value of the element. This field is useful to the Dashboard in order to simplify the insertion of data for the user before launching a model. For the PIXEL platform default endpoints for the *ih-api* and the *es-api* are given in the example.

The **input** array includes all needed inputs needed by the model. The way they are defined follow a common format, as you can appreciate in the JSON example. The different fields are:

- **name (mandatory)**: name of the input (string).
- **type (mandatory)**: this is the data source (data format) of this input (string). PIXEL specifies a set of different data models trying to follow FIWARE. Formats are important because otherwise models will have to convert them.
- **description (optional)**: this is the description of this element (string).
- **supportedConnectors (mandatory)**: this is the connector that will use the model to get the input element (string). In PIXEL typically it will be the *ih-api* connector. This means that the input is located in the IH and the model will need to contact the IH API to obtain this particular piece of information.
- **metadata (optional)**: optional element to include extra information specific for the model.
- **required (mandatory)**: indicates if this input is mandatory to execute the model (boolean). This means, in case it is not required, this input might not appear in the JSON instance. This field is useful for the Dashboard, so that it will block (or not) the user unless this input is given before letting him/her execute the model.
- **forceinput (mandatory)**: indicates if this input can be forced by the user to execute the model (boolean). This means, in case it is forced, that the input will appear in a different array in the JSON instance. An example will be given in the chapter of *Execution of a model (instance.json)*. The ability of forcing inputs is interesting for what-if scenarios, where the user may work with fictitious data to get results, without the need to have this data stored previously in the IH. This field is important for the Dashboard, so that it can allow the user to generate the input (i) as an input element within the input array or (ii) as a force input element. For the model, when it starts execution and checks for inputs, the *force input* array will have priority. This will also be explained in the chapter of *Execution of a model (instance.json)*
- **options (mandatory)**: this is an array of elements containing the different items to access the element by using the service endpoint. The elements given in the example refer to the *sourceId* and *lower* and *upper* temporal limits. This means that the model will obtain the input by using the *ih-api* connector (and thus its *options* array) and also the options array of the input element. With the *ih-api* connector the model only knows the API, but to reach specific data it needs to build the specific request based on the additional parameters. In the example, *sourceId* maps to an index of the database, and the temporal limits allow to get all data between that time interval. Let us focus only on the format of the first element, as the format is the same for the others:
- **name (mandatory)**: name of the element (string).
- **type (mandatory)**: type of the element (string).
- **value (optional)**: default value of the element. This field is useful to the Dashboard in order to simplify the insertion of data for the user before launching a model. For the PIXEL platform with inputs coming from NGSI agents, values are typically "*arh-lts-"*.
- **description (optional)**: description of the element.
- **pattern (optional)**: Allows to specify if the string has to follow a specific pattern. This field is only useful for the Dashboard.
- **metadata (optional)**: optional element to include extra information specific for the model.
- **required (mandatory)**: indicates if this element is mandatory to build the input (boolean). This means, in case it is not required, this element might not appear in the JSON instance. This field is useful for the Dashboard, so that it will block (or not) the user unless this element is given before letting him/her execute the model.

The **output** array follows a similar format as the input array, as you can appreciate in the JSON example. The different fields are:

- **name (mandatory)**: name of the output (string). Typically, if you are having just one element in the array, you can name it *output.*
- **type (mandatory)**: this is the data source (data format) of this output (string). FIWARE data models are more related to NGSI agents and inputs, for outputs there is no strong need to follow such model (although possible).
- **required (mandatory)**: indicates if this output is mandatory for the model (boolean). For one single element in the output array this will typically be *true.*
- **description (optional)**: this is the description of this element (string).

- **supportedConnectors (mandatory)**: this is the connector that will use the model to store the output element (string). In PIXEL typically it will be the *es-api* connector. This means that the output will be stored in the IH and the model will need to contact the Elasticsearch API to save this particular piece of information.
- **metadata (optional)**: optional element to include extra information specific for the model.
- **options (mandatory)**: this is an array of elements containing the different items to allow the model to store the output in a particular place so that it can be later be accessible to other components (e.g., visualization component) Let us focus only on the format of the first (and only) element:
- **name (mandatory)**: name of the element (string).
- **type (mandatory)**: type of the element (string).
- **description (optional)**: description of the element.
- **pattern (optional)**: Allows to specify if the string has to follow a specific pattern. This field is only useful for the Dashboard.
- **required (mandatory)**: indicates if this element is mandatory to build the output (boolean). For one single element, it is *true* unless the model does not need to store any output.
- **value (optional)**: default value of the element. This field is useful to the Dashboard in order to simplify the insertion of data for the user before launching a model. For the PIXEL platform values are typically *models-output-*. Therefore, any component looking for results from this model can easily locate all results under this index.

The **logging** array follows a similar format as the input and output array, as you can appreciate in the JSON example. The different fields are:

- **name (mandatory)**: name of the logging (string). Typically, if you are having just one element in the array, you can name it <*model-name-logging.*
- **type (mandatory)**: this is the data source (data format) of this logging (string). In PIXEL, it has already been defined as a specific format containing *iso_8601_timestamp, id_execution, id_model, type and message* . Models should log at least start and end time.
- **supportedConnectors (mandatory)**: this is the connector that will use the model to store the logging element (string). In PIXEL typically it will be the *es-api* connector. This means that the logging will be stored in the IH and the model will need to contact the Elasticsearch API to save this particular piece of information.
- **description (optional)**: this is the description of this element (string).
- **required (mandatory)**: indicates if this logging is mandatory for the model (boolean). For one single element in the logging array, this will typically be *true*.
- **verbose (optional)**: verbosity level for the model (string). A model might provide different levels for logging (e.g., DEBUG, INFO, WARNING, ERROR, etc.)
- **metadata (optional)**: optional element to include extra information specific for the model.
- **options (mandatory)**: this is an array of elements containing the different items to allow the model to store the logging in a particular place so that it can be later be accessible to other components (e.g., visualization component) Let us focus only on the format of the first (and only) element:
- **name (mandatory)**: name of the element (string).
- **type (mandatory)**: type of the element (string).
- **description (optional)**: description of the element.
- **pattern (optional)**: Allows to specify if the string has to follow a specific pattern. This field is only useful for the Dashboard.
- **required (mandatory)**: indicates if this element is mandatory to build the output (boolean). For one single element, it is *true* unless the model does not need to store any logging.
- **value (optional)**: default value of the element. This field is useful to the Dashboard in order to simplify the insertion of data for the user before launching a model. For the PIXEL platform values are typically *models-logging>*. All models will then log data under the same index, but they can be filtered by *id_model, id_execution*, etc.

The logged info is basically described in the Figure below:

In case of error, a typical logging table should be like this

timestamp (ISO 8601)	id_model (UUID string)	id_execution (UUID string)	type (string)	message (string)
2021-01-20T18:24:40+02:00	JJUSG676531	JJH6757423	start	start
2021-01-20T18:24:41+02:00	JJUSG676531	JJH6757423	error	Error 1 message
2021-01-20T18:24:42+02:00	JJUSG676531	JJH6757423	error	Error 2 message
2021-01-20T18:24:40+02:00	JJUSG676531	JJH6757423	end	error

In case of success, a typical logging table should be like this

timestamp (ISO 8601)	id_model (UUID string)	id_execution (UUID string)	type (string)	message (string)
2021-01-20T18:24:40+02:00	JJUSG676531	JJH6757423	start	start
2021-01-20T18:24:41+02:00	JJUSG676531	JJH6757423	info	Info1 message
2021-01-20T18:24:42+02:00	JJUSG676531	JJH6757423	info	Info2 message
2021-01-20T18:24:40+02:00	JJUSG676531	JJH6757423	end	Success

Figure 38: Log Information

The timestamp is given in ISO 8601 format since the current version v0.2 (in the previous version -v0.1 it was in UNIX time).

Annex 2. JSON format (execution - normal)

In order to better understand the JSON format (instead of providing the schema), the basic **pingcount** model is used as an example, available at *https://gitpixel.satrdlab.upv.es/benmomo/ngsi-agents-thpa/src/master/thpa-model-pingcount*. The model is able to count the number of pings of a given set of elements of type Ping. This is a PIXEL data model generated by a basic PingTest NGSI agent (available at *https://gitpixel.satrdlab.upv.es/benmomo/ngsi-agents-thpa/src/master/thpa-ping*)

The JSON format is as follows:


```
{
  "idRef": "601088c715f76f0007542876",
  "id": "999d771ae2cabc05ec59a999",
  "name": "pingcount-execution1",
  "description": "pingcount execution 1",
  "mode": "ExecAsync",
  "input": [{
    "name": "pingset",
    "category": "ih-api",
    "type": "[urn:pixel:DataSource:Ping]",
    "description": "array of JSON documents of type urn:pixel:DataSource:Ping",
    "metadata": {},
    "options": [{
          "name": "url",
          "type": "string",
          "description": "URL string representing the API to reach the IH",
          "value": "http://172.24.1.17:8080/archivingSystem/extractor/v1/data"
          }, {
          "name": "sourceId",
          "type": "string",
          "description": "sourceId, mapped to Index, within the IH where the NGSI generated pings are stored",
          "value": "urn:pixel:DataSource:Ping"
          }, {
          "name": "from",
          "type": "date-time",
          "description": "From timestamp. ISO8601 format",
          "value": "2021-01-20T11:11:11+02:00"
          }, {
          "name": "to",
          "type": "date-time",
          "description": "To timestamp. ISO8601 format",
          "value": "2021-01-20T11:20:11+02:00"
          }
    ]
    }
  ],
  "output": [{
    "name": "output",
    "category": "es-api",
    "type": "urn:pixel:DataModelResult:PingCount",
    "description": "output provided by the PingCount execution",
    "metadata": {},
    "options": [{
          "name": "url",
          "type": "string",
          "description": "URL string representing the API to reach Elasticsearch",
          "value": "http://172.24.1.11:9200"
          }, {
          "name": "es_index",
          "type": "string",
          "description": "Elasticsearch index to store the result  of the execution.",
          "value": "models-output-pingcount"
          }
    ]
    }
  ],
  "logging": [{
    "name": "logging",
    "category": "es-api",
    "type": "pixel-logging-format",
    "description": "activity logging for the pingCount model. id_execution, idRef/id_model is given by the OT at invocation time",
    "verbose" : null,
    "metadata": {},
```

```
    "options": [{
            "name": "url",
            "type": "string",
            "description": "URL string representing the API to reach Elasticsearch",
            "value": "http://172.24.1.11:9200"
            }, {
            "name": "es_index",
            "type": "string",
            "description": "Elasticsearch index to store the logs of the execution. start, end, error",
            "value": "models-logging"
            }
    ]
    }
  ]
}
```


Note that the format is very similar to the model as in fact, it extends from it. The Main specific aspects, differences or remarks are described below:

- **id**: this is the ID of the execution itself. It is given by the OT at execution time and allows the model to log activity in the IH including this id, so that it is uniquely identified.
- **idRef**: this is the ID of the model itself. By the time the model is published in the OT, it gets an ID. This is also useful for logging and outputting info, so that later it is possible to search for all executions (and results) of a given model.
- **mode**: this refers to the way the model is executed (*Subscription*, *ExecSync*, *ExecAsync*). These were given as independent fields in the *getInfo.json* format.

Note also that some fields from *getInfo.json*, such as *version, type, category* are not needed here.

Note that there is no *system* or *connector* element. Instead of that, it is *ported* directly to each input element, as the format of the options element is the same. In our example (*pingcount*), this means that the single input *pingtest* element will include the *url* options element, as it is of type *ih-api*. As the other elements (*user,password*) are not required, they do not have to be present there.

The same reasoning applies for the *output* and *logging* elements. Here they include the *url options* element from the *es-api* connector.

Note also as the final remark that some fields from the *getInfo.json* are not necessary for the instance, such as *pattern* and *required*, as they are interpreted by the Dashboard and have no further relevance.

Annex 3. JSON format (execution – forced inputs)

In the previous section it is provided a normal *instance.json* with inputs and outputs. But it is possible that some inputs might be forced by the user to analyse *what-if scenarios*. In such cases, the data is provided raw within the *instance.json*, and the user has to directly type it. Obviously, this is valid (user friendly) for easy inputs and not long data structures.

Forced inputs are specified at model level (*getInfo.json*) with the boolean parameter *force input* set to *true*. If this is the case and the user includes raw data, then such data appears in a new array called *force input*. This can be observed in the example (*pingcount*), where the only input *pingset* is added.

The JSON format is as follows:


```
{
  "idRef": "5e3d771ae2cabc05ec59a14c",
  "id": "999d771ae2cabc05ec59a999",
  "name": "pingcount-execution1",
  "description": "pingcount execution 1",
  "mode": "ExecAsync",
  "input": [],
  "forceinput": [
    {
        "name": "pingset",
        "value":[
        {
        "id": "Ping",
        "type": "Ping",
        "source": {
                "value": "urn:pixel:DataSource:Ping",
                "type": "Text"
        },
        "from": {
                "value": "059501e23885-172.17.0.2", "type": "Text"
        },
        "when": {
                "value": "2021-01-20T18:21:40+02:00","type": "Text"
        }
        },
        {
        "id": "Ping",
        "type": "Ping",
        "source": {
                "value": "urn:pixel:DataSource:Ping", "type": "Text"
        },
        "from": {
                "value": "059501e23885-172.17.0.2", "type": "Text"
        },
        "when": {
                "value": "2021-01-20T18:22:40+02:00",      "type": "Text"
        }
        },
        {
        "id": "Ping",
        "type": "Ping",
        "source": {
                "value": "urn:pixel:DataSource:Ping","type": "Text"
        },
        "from": {
                "value": "059501e23885-172.17.0.2", "type": "Text"
        },
        "when": {
                "value": "2021-01-20T18:23:40+02:00", "type": "Text"
        }
        },
        {
        "id": "Ping",
        "type": "Ping",
        "source": {
                "value": "urn:pixel:DataSource:Ping", "type": "Text"
        },
        "from": {
```

```
                        "value": "059501e23885-172.17.0.2", "type": "Text"
                },
                "when": {
                        "value": "2021-01-20T18:24:40+02:00", "type": "Text"
                }
                }
                ]
        }

    ],
    "output": [{
        "name": "output",
        "category": "es-api",
        "type": "urn:pixel:DataModelResult:PingCount",
        "description": "output provided by the PingCount execution",
        "metadata": {},
        "options": [{
                "name": "url",
                "type": "string",
                "description": "URL string representing the API to reach Elasticsearch",
                "value": "http://localhost:9200"
                }, {
                "name": "es_index",
                "type": "string",
                "description": "Elasticsearch index to store the result  of the execution.",
                "value": "models-output-pingcount"
                }
        ]
        }
    ],
    "logging": [{
        "name": "logging",
        "category": "es-api",
        "type": "pixel-logging-format",
        "description": "activity logging for the pingCount model. id (id_execution, idRef (id_model) are given by the OT at invocation time",
        "verbose" : null,
        "metadata": {},
        "options": [{
                "name": "url",
                "type": "string",
                "description": "URL string representing the API to reach Elasticsearch",
                "value": "http://localhost:9200"
                }, {
                "name": "es_index",
                "type": "string",
                "description": "Elasticsearch index to store the logs of the execution (start, end, error)",
                "value": "models-logging"
                }
        ]
        }
    ]
}
```


Note the difference in this example with the previous section:

- **input**: this element appears as empty, and it might even not appear, if all inputs are provided in the *force input* array. But if a model has several inputs, some of them may appear here whereas others may appear in the *force input* array. From the point of view of the logic of the model, when it starts checking for inputs, it should *FIRST* check them in the *force input* array, in case it exists and *THEN*, if not, check in the *input* array. This means that *force input* has priority over *input.*
- **forceinput**: this represents the raw data introduced directly here, as the value field of the *pingset* element. This should theoretically be the same input (maybe with some transformations) as the model will obtain by invoking the IH with the *url, sourceId, from and to* option parameters.

The arrays *output* and *logging* remain the same.

Annex 4. JSON format (scheduled executions)

In the previous section, it is provided two *instance.json* examples (with and without *forceinputs*). For *scheduledInstances* this is irrelevant, as the main difference lies in the inclusion of the *scheduleInfo* element. Typically, it will use more the *input* than the *force input*. An example with *input* for the *pingcount* model is provided:

The JSON format is as follows:


```
{
  "idRef": "601088c715f76f0007542876",
  "name": "pingcount-scheduledExecution1",
  "description": "pingcount execution 1",
  "mode": "ExecAsync",
  "user": null,
  "active":true,
  "scheduleInfo" : {
     "start" : "2021-01-20T11:11:11+02:00",
     "unit" : "minute",
     "value" : 5
     },
  "input": [{
     "name": "pingset",
     "category": "ih-api",
     "type": "[urn:pixel:DataSource:Ping]",
     "description": "array of JSON documents of type urn:pixel:DataSource:Ping",
     "metadata": {},
     "options": [{
              "name": "url",
              "type": "string",
              "description": "URL string representing the API to reach the IH",
              "value": "http://172.24.1.17:8080/archivingSystem/extractor/v1/data"
              }, {
              "name": "sourceId",  "type": "string",
              "description": "sourceId, mapped to Index, within the IH where the NGSI generated pings are stored",
              "value": "urn:pixel:DataSource:Ping"
              }, {
              "name": "from",  "type": "date-time",
              "description": "From timestamp. ISO8601 format",
              "value": "${DATE_MINUTE_INIT}"
              }, {
              "name": "to", "type": "date-time",
              "description": "To timestamp. ISO8601 format",
              "value": "${DATE_MINUTE_LAST}"
              }
     ]
     }
  ],
  "output": [{
     "name": "output",
     "category": "es-api",
     "type": "urn:pixel:DataModelResult:PingCount",
     "description": "output provided by the PingCount execution",
     "metadata": {},
     "options": [{
              "name": "url", "type": "string",
              "description": "URL string representing the API to reach Elasticsearch",
              "value": "http://172.24.1.11:9200"
              }, {
              "name": "es_index", "type": "string",
              "description": "Elasticsearch index to store the result  of the execution.",
              "value": "models-output-pingcount"
              }]
     } ],
  "logging": [{
     "name": "logging",
     "category": "es-api",
     "type": "pixel-logging-format",
     "description": "activity logging for the pingCount model. id_execution, idRef/id_model is given by the OT at invocation time",
     "verbose" : null,
     "metadata": {},
     "options": [{
              "name": "url",
              "type": "string",
```

```
            "description": "URL string representing the API to reach Elasticsearch",
            "value": "http://172.24.1.11:9200"
            }, {
            "name": "es_index",
            "type": "string",
            "description": "Elasticsearch index to store the logs of the execution. start, end, error",
            "value": "models-logging"
            }]
    } ]
}
```


You can easily see that there is practically no difference compared to an instance, except for: - **scheduleInfo**: this has already been commented before

- **active**: this field is no longer used since version v0.2 (you can leave it to true, or just remove it).
- **timing functions**: you should have noticed that timing parameters in the input element, such as *from* and *to*, have a special value (*${DATE_MINUTE_INIT}*, *${DATE_MINUTE_INIT}*). This makes absolutely sense as otherwise, the *pingcount* model would always make the same operation with the same input. By means of these special functions, data will always refer to the current minute, every 5 minutes (according to s*cheduleInfo* in the example). Thus:
- every 5 minutes the *pingcount* model will be launched.
- input data will be taken from the last minutes.
- calculation will be performed and the result will be stored in the IH.

The timing functions are pre-processed by the OT, so the *pingcount* model will get an ISO 8601 date every time it is launched. This functionality is particularly useful to make periodic calculations every hour, day, week, month. The set of timing functions currently supported by the OT (v0.2 version) are:

Table 4: Time functions

Format	Description(Unix format - millis)	Potential Use
${DATE_current}	Current date	Models started by triggers
${DATE_MINUTE_INIT}	Date of the first second of the current minute	test,RT data
${DATE_MINUTE_LAST}	Date of the last second of the current minute	test,RT data
${DATE_HOUR_INIT}	Date of the first second of the current hour	traffic,weather
${DATE_HOUR_LAST}	Date of the last second of the current minute	traffic,weather
${DATE_DAY_INIT}	Date of the first second of the current day	PAS
${DATE_DAY_LAST}	Date of the last second of the current day	PAS
${DATE_WEEK_INIT}	Date of the first second of the current week	PEI
${DATE_WEEK_LAST}	Date of the last second of the current week	PEI
${DATE_MONTH_INIT}	Date of the first second of the current month	PEI
${DATE_MONTH_LAST}	Date of the last second of the current month	PEI

Note: Additional timing functions could be added (by demand) on further versions of the OT