

## D6.3 PIXEL data acquisition, information hub and data representation v1

<b>Deliverable No.</b>	D6.3	<b>Due Date</b>	31-Aug-2019
<b>Type</b>	Report	<b>Dissemination Level</b>	Public
<b>Version</b>	1.0	<b>Status</b>	Final
<b>Description</b>	<p>Deliverables D6.3 and D6.4 are the main asset of software documentation for this project, including data sources, collecting mechanisms, technologies, protocols, the operational analytics engine, operational tools and the visualization and notification module.</p> <p>This release includes the report about software mechanisms for data acquisition and description of the performance of a service in the port area. It also includes a technological state of art, a short description of the technology developed and the software itself.</p> <p>The second version will contain the design of integrated information system, theoretical representation of data and the requirements description. Generally, <b>the first version contains software prototypes while</b> the second will provide the final products. Associated tasks: T6.2, T6.3, T6.4, T6.5 and T6.6.</p>		
<b>Work Package</b>	WP6		

## Authors

Name	Partner	e-mail
Benjamin Molina	P01 UPV	benmomo@upvnet.upv.es
Eneko Olivares	P01 UPV	enolgor@teleco.upv.es
Miguel A. Llorente Carmona	P02 PRODEVELOP	mllorente@prodevelop.es
Julián Martínez	P02 PRODEVELOP	jmartinez@prodevelop.es
José A. Clemente	P02 PRODEVELOP	jclemente@prodevelop.es
Flavio Fuat	P03 XLAB	flavio.fuat@xlab.si
Matjaž Sprogar	P03 XLAB	matjaz.sprogar@xlab.si
Damjan Murn	P03 XLAB	damjan.murn@xlab.si
Gilda De Marco	P04 INSIEL	gilda.demarco@insiel.it
Paolo Casoto	P04 INSIEL	paolo.casoto@gmail.com
Marc Despland	P06 ORANGE	marc.despland@orange.com
Fabien Battello	P06 ORANGE	fabien.battello@orange.com

## History

Date	Version	Change
31-May-2019	0.1	ToC and task assignments
01-June-2019	0.2	First draft for sections 3, 4, 5.
15-June-2019	0.3	Relation with PIXEL objectives and use cases, requirements (sect. 1)
28-June-2019	0.4	Section 6 first draft. Section 7 – list of software modules.
7-July-2019	0.5	Sections 1-5 for internal review. Section 6 aligned with D8.1.
25-July-2019	0.6	All sections finalised.
10-Aug-2019	0.7	First version of software code provided.
16-Sept-2019	0.8	Software code provided after Ljubljana Workshop alignment. Ready for internal review.
30-Sept-2019	1.0	Official release

## Key Data

<b>Keywords</b>	ICT framework, data acquisition, cybersecurity, operational tools, visualisation, dashboards, port operations, software
<b>Lead Editor</b>	Flavio Fuart, P03 XLAB
<b>Internal Reviewer(s)</b>	P12 – ASPM P08 – MEDRI Innovation Review, P03 XLAB, Joao Pita Costa

## Abstract

PIXEL Enabling ICT Infrastructure framework is one of the key outcomes of PIXEL activities. The main goal is to compose a complete data-centric port solution, allowing data-level interoperability of different systems, including legacy industrial and port operations systems.

This framework provides sound technological foundations for efficient and cost-effective execution of models, simulations and predictions that are part of the PIXEL environmental impacts assessment model, to be used by ports of the future for efficient management and tackling environmental issues. The framework will also integrate supporting ICT tools for the calculation of the Port Environmental Index (PEI), as a key parameter to improve operations in the ports of the future.

The most important asset of this deliverable is the provision of software prototypes for the data acquisition layer, information hub, operational tools, dashboard & notifications and security & privacy modules. **PIXEL Data acquisition** provides software mechanisms to enable appropriate data acquisition in different port areas, from logistic agents and public data sources. Data acquisition is based on FIWARE, a curated framework of open source platform components for smart solutions, financed through several EU research programmes. **PIXEL Information Hub** is the primary information source in all port related activities and is being designed to strengthen the capacity and accuracy of port of the future logistics processes and to maintain a high level of service and offer a system which will be in line with the needs and expectations of users. **PIXEL Operational tools** enable model-based simulations and analysis of data gathered and fused in the PIXEL Information Hub to provide more flexible operations and create decision-making tools, resulting from other PIXEL activities. **PIXEL Integrated Dashboard and Notifications** provide the visual environment to show in a single dashboard the different KPIs, including PEI, user interfaces for the operational tools and the configuration and management tools needed to control other PIXEL framework components. The dashboard has been designed to support generalization to other ports or terminals with similar needs. **PIXEL Security and Privacy** is a transversal activity that provides end-to-end security for the PIXEL platform by deploying basic cybersecurity mechanisms for all other ICT components.

In addition to software prototypes this deliverable provides a report about software mechanisms for data acquisition, description of methods for performance measurements of services in the port area, technological state of art and a brief description of the technology developed. While this version of the deliverable provides software prototypes, the second version will provide the final products.

## Statement of originality

This document contains material, which is the copyright of certain PIXEL consortium parties, and may not be reproduced or copied without permission. This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

The information contained in this document is the proprietary confidential information of the PIXEL consortium (including the Commission Services) and may not be disclosed except in accordance with the consortium agreement.

The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the project consortium as a whole nor a certain party of the consortium warrant that the information contained in this document is capable of use, nor that use of the information is free from risk and accepts no liability for loss or damage suffered by any person using this information.

The information in this document is subject to change without notice.

The content of this report reflects only the authors' view. The Innovation and Networks Executive Agency (INEA) is not responsible for any use that may be made of the information it contains.

# Table of contents

1.	About this document.....	9
1.1.	Deliverable context.....	9
1.2.	The rationale behind the structure .....	11
1.3.	Version-specific notes .....	11
2.	Introduction .....	12
2.1.	Relation with PIXEL objectives and use cases .....	12
2.1.1.	Use case of GPMB .....	13
2.1.2.	Use case of Port of Monfalcone .....	14
2.1.3.	Use case of Port of Piraeus .....	15
2.1.4.	Use case of Port of Thessaloniki .....	16
2.2.	Relation to requirements.....	17
3.	Technological state of the art.....	19
3.1.	PIXEL Data Acquisition.....	19
3.2.	PIXEL Information Hub.....	20
3.3.	PIXEL Operational Tools.....	20
3.4.	PIXEL Integrated Dashboard and Notification .....	21
3.5.	PIXEL Security and Privacy .....	22
4.	PIXEL platform prototype.....	23
4.1.	PIXEL software components.....	23
4.1.1.	PIXEL Data Acquisition.....	23
4.1.2.	PIXEL Information Hub.....	23
4.1.3.	PIXEL Operational Tools .....	24
4.1.4.	PIXEL Integrated Dashboard and Notification .....	26
4.1.5.	PIXEL Security and Privacy.....	27
4.2.	PIXEL development environment.....	29
4.2.1.	Git repository.....	29
4.2.2.	FIWARE Lab.....	30
5.	Software mechanisms for data acquisition .....	32
6.	Service performance in the port area.....	35
6.1.	PIXEL Data Acquisition.....	35
6.2.	PIXEL Information Hub.....	38
6.3.	PIXEL Operational Tools.....	40
6.4.	PIXEL Integrated Dashboard and Notification .....	43
6.5.	PIXEL Security and Privacy .....	45
7.	Software release and documentation .....	48

## List of tables

Table 1. List of input data for GPMB.....	13
Table 2. List of input data for the Port of Monfalcone.....	14
Table 3. List of input data for the Port of Piraeus .....	15
Table 4. List of input data for the Port of Thessaloniki.....	16
Table 5. Relation to PIXEL requirements .....	17
Table 6. Technological SotA: Data Acquisition Core .....	19
Table 7. Technological SotA: PIXEL Information Hub.....	20
Table 8. Technological SotA: PIXEL Operational Tools.....	20
Table 9. Technological SotA: PIXEL Integrated Dashboard and Notification .....	21
Table 10. Technological SotA: PIXEL Security and Privacy .....	22
Table 11. NGSI agents already available .....	32
Table 12. NGSI agents – identified needs and development status.....	33
Table 13. PIXEL Data Models .....	34
Table 14. KPIs measurement approach: PIXEL Data Acquisition.....	35
Table 15. KPIs measurement approach: PIXEL Information Hub.....	38
Table 16. KPIs measurement approach: PIXEL Operational Tools.....	40
Table 17. KPIs measurement approach: PIXEL Integrated Dashboard and Notification .....	43
Table 18. KPIs measurement approach: PIXEL Security and Privacy .....	45
Table 19. Software release overview: PIXEL Data Acquisition .....	48
Table 20. Software release overview: PIXEL Information Hub .....	49
Table 21. Software release overview: PIXEL Operational Tools .....	49
Table 22. Software release overview: PIXEL Integrated Dashboard and Notification.....	49
Table 23. Software release overview: PIXEL Security and Privacy .....	50

## List of figures

Figure 1. PIXEL energy model dashboard .....	25
Figure 2. PIXEL Visualisation Example .....	27
Figure 3. PIXEL Gogs repository screenshot.....	30
Figure 4. PIXEL virtual environment overview .....	31
Figure 5. PIXEL virtual server instances overview.....	31

## List of acronyms

Acronym	Explanation
AIS	Automatic Identification System
API	Application Programming Interface
ARPA	Agenzia regionale per la protezione ambientale
CEP	Complex Event Processing
DAL	Data Acquisition Layer
DOA	Description of Action
FAIR	Facility for Antiproton and Ion Research
GIS	Geographic information system
GPMB	Grand port maritime de Bordeaux
GUI	Graphic User Interface
ICT	Information and communications technology
IH	Information Hub
IT	Information Technology
KPI	Key Performance Indicator
LTS	Long Term Storage
NGSI	Name of the FIWARE API
NIFI	Apache NiFi is a software project designed to automate the flow of data between software systems
NMEA	National Marine Electronics Association
OS	Operating System
PCS	Port Community System
PEI	Port Environmental Index
WP	Work Package
PMS	Port Management System
PMIS	Port Management Information System
REST	Representational state transfer
SILI	Sistema Informativo Logistico Integrato
SotA	State-of-the-Art
UI	User Interface
URL	Universal Resource Locator
XACML	eXtensible Access Control Markup Language



# 1. About this document

This deliverable describes and provides the initial work that has been done in the technical tasks of WP6, Enabling ICT (Information and communications technology) infrastructure framework. The deliverable consists of:

- The reporting part (this document), where developed software mechanisms have been described, methods for performance measurement in the port area have been defined, a state-of-the-art research and status of developments in M16 have been described.
- The source code for software components (software prototypes) and documentation for building and deploying the components.

This document is the first of two iterations and contains software prototypes while the second will provide the final products.

The deliverable is intended to be a starting point for WP7 (Pilot trials integration, deployment and evaluation). Software prototypes and documentation resulting from D6.1 will be used as input for planning the integration with existing system and creating a proof-of-concept (T7.1). Similarly, D6.3 in combination with D6.1/D6.2 (forthcoming) will be used for the standardisation of the interface between WP4, WP5 and WP6. The section about the performance in the port area (Section 6) is an important input for the technical assessment performed in WP8 (Task 8.2 - Technical impact assessment).

This deliverable is primarily intended as a main reference for software developers and other IT (Information Technology) professionals interested in using the PIXEL ICT framework in the PIXEL context. Nevertheless, provided prototypes could also be used by early adopters for proof-of-concept deployments. The next version, D6.4, will provide a final product intended for a wider audience of ICT professionals.

## 1.1. Deliverable context

Keywords	Lead Editor
Objectives	<p><i><u>Objective 1:</u> Enable the IoT-based connection of port resources, transport agents and city sensor networks.</i></p> <p>This deliverable provided the first software prototype for IoT enablement and interconnection of different data providers listed in the objective. The prototype already supports data collection from generic sources. In addition to generic mechanisms, WP6 also provided a proof-of-concept implementation with public AIS (Automatic Identification System) data, AIS data and port calls data provided by PMS (Port Management System) / PCS (Port Community System) of GPMB (Grand port maritime de Bordeaux). The mechanisms for full integration will be provided in D6.2/D6.4, while the integration itself will be finalised in WP7.</p>
	<p><i><u>Objective 2:</u> Achieve an automatic aggregation, homogenization and semantic annotation of multi-source heterogeneous data from different internal and external actors.</i></p> <p>D6.3 provides the prototype implementation of the reference model provided in D6.1. It also further elaborates the data model at a more technical level. In this deliverable the PIXEL information hub provides a generic data aggregation mechanism, homogenization is performed both at Data Acquisition Layer level as well as in the information hub and the semantic annotation is done at data acquisition level.</p>
	<p><i><u>Objective 3:</u> Develop an operational management dashboard to enable a quicker, more accurate and in-depth knowledge of port operations.</i></p>

Keywords	Lead Editor
	<p>D6.3 provides the first dashboard prototype resulting from T6.4 (see 4.1.4. PIXEL Integrated Dashboard and Notification).</p> <p><i>Objective 4: Model and simulate port operations processes for automated optimisation.</i></p> <p>D6.3 provides the first operational tools prototype resulting from T6.5 (see section 4.1.3. PIXEL Operational Tools). These tools give high-level technological support for the configuration and execution of predictive algorithms models developed in WP4.</p> <p><i>Objective 5: Develop predictive algorithms.</i></p> <p>Similar as for Objective 4, D6.3 provides the first operational tools prototype resulting from T6.5 (see section 4.1.3. PIXEL Operational Tools). Operational tools prototype provides an initial generic framework for the setup and execution of predictive algorithms.</p>
Exploitable results	D6.3 provides a prototype version of the following exploitable assets: PIXEL Data Acquisition; PIXEL Information Hub; PIXEL Operational Tools; PIXEL Integrated Dashboard and Notification; PIXEL Security and Privacy.
Work plan	This deliverable is the result of work performed from M7 to M16 on tasks T6.2 – PIXEL Data Acquisition, T6.3 - PIXEL Information Hub, Task 6.4 - PIXEL Operational Tools, T6.5 - PIXEL Integrated Dashboard and Notification, Task 6.6 – PIXEL Security and Privacy.
Milestones	This is the first deliverable for verification of the MS7 ICT solution developed (M26). The second deliverable in this series, D6.4 will be released in M26.
Deliverables	This deliverable follows the system architecture defined in D6.1 PIXEL Information system architecture and design v1. Requirements, scenarios and use cases defined in D3.2, D3.3 and D3.4 have been used to identify relevant ICT-related tasks. This deliverable is the first in the series of two, with the final version D6.4 scheduled for M26.
Risks	<p><b>WT5#6 Technical activities are not completed on time, are not aligned with the main objective, are not accurate or present a lack of consistency.</b> This deliverable shows that technical activities related to T6.2 – T6.6 have been executed in a timely fashion in accordance to the architecture proposed in D6.1.</p> <p><b>WT5#8 Requirements fail to align with ICT systems.</b> D6.3 provides a relation between PIXEL use cases and requirements and the technology developed in WP6. To minimise this risk, list of requirements and their relation to WP6 components is being consulted in each development phase.</p> <p><b>WT5#14 Due to harshly divergences between formats of output/input data of ICT systems to integrate, the development can be delayed or paralyzed, and some extra effort will be needed to carry out the project.</b> Particular attention is being devoted to the analysis and definition of data models in WP6. While the generic principles have been provided in D6.1, this deliverable provides a more detailed list of data entities identified in PIXEL.</p> <p><b>WT5#15 IoT components have security vulnerabilities.</b> This assessment is part of T6.6. While in D6.3 generic authentication mechanisms are being provided, subsequent deliverables in WP6 (D6.2 and D6.4), will provide an assessment of IoT security vulnerabilities in PIXEL use-cases.</p>

Keywords	Lead Editor
	<p><b>WT5#16 PIXEL Information Hub results to be bloated as different stakeholders are not able to find usable data within it from other agents involved in port activities.</b></p> <p>Tackling this risk is tightly related to the definition of a unified data model based on FIWARE entity types and data structures resulting from WP4. Section 5.3 Data Models provides the status of developments related to this topic.</p>

## 1.2. The rationale behind the structure

This report describes the work performed in T6.2 – T6.6 of PIXEL. Except the introduction and conclusion, each section provides specific content defined in the DoA deliverable description. Topics that are covered in each PIXEL component are further split in sub-sections for each of those components:

1. PIXEL Data Acquisition
2. PIXEL Information Hub
3. PIXEL Operational Tools
4. PIXEL Integrated Dashboard and Notification
5. PIXEL Security and Privacy

This report consists of the following sections:

1. **About this document:** Deliverable context in relation to the PIXEL DoA, work packages, tasks and other deliverables.
2. **Introduction:** Relation with PIXEL objectives, use cases and requirements.
3. **Technological state of art:** Technological choices in relation to D6.1 and the state of their implementation.
4. **PIXEL platform prototype:** Provides a **short description of the technology developed** and status of development of each component.
5. **Software mechanisms for data acquisition:** Report about data acquisition activities related to the implementation of the Data Acquisition Layer and selected FIWARE data models.
6. **Service performance in the port area:** Description of performance tests for each PIXEL component.
7. **Software release and documentation:** Description of the software release that is part of this deliverable. The section provides the description of files provided for each PIXEL component.
8. **Conclusions:** Closing remarks.

## 1.3. Version-specific notes

This is the first in a series of two deliverables. This report and software release provide the results of WP6 performed until M16 of the project. The final version of this deliverable, due in M26 of the project will provide a full report about applied methods, achieved results and the final PIXEL software release.

## 2. Introduction

Developments of the PIXEL ICT Infrastructure Framework are driven by real needs of the ports involved in the project. Those needs and requirements are the key results of WP3. In order to keep development in line with overall PIXEL objectives, each technical deliverable provides an introductory section where the relation with objectives, use cases and requirements is defined. The following sections in this chapter provide that overview for software modules developed in WP6.

### 2.1. Relation with PIXEL objectives and use cases

All work described in D6.3 and performed as part of tasks T6.2-T6.6 aims to fulfil objectives 1, 2, 3, 4 and 5, which are listed in section 1.1 of this document.

Even if a more in-depth analysis has been performed in the previous task (T6.1- architecture), this deliverable presents an initial software draft of the different main architectural modules. The development of DAL (*Data Acquisition Layer*) has the potential to **connect** (Obj1) different data sources to a common broker in a standardized way. Here the concept of data source is wide and applies not only to sensors in ports but also to open data and some existing port applications (e.g. vessel calls) providing data. The DAL represents the first level of data integration and support, by means of NGSI agents, **automatic aggregation** (Obj2) of connected data sources.

The second level of data integration is represented by the IH (*Information Hub*), where heterogeneous port data is brought to a common space (hub) and exposed in a **harmonized** (Obj2) way. To achieve this objective, IH exposes an API (Application Programming Interface) for discovering, querying and storing data. Such port data has also been converted into more common and useful **semantic data formats** (Obj2) that will later be integrated in higher level applications: some examples will be provided in PIXEL with the Dashboard and Operational Tools (Obj4, Obj5), but in fact the objective is general for future applications developed by ports and/or port agents.

The IH (and in fact also the DAL) is designed not only to treat data but also to provide an **interoperable framework for actors, such as port agents and city networks** (Obj1) to exchange data and optimize resources and common policies.

To **appropriately manage** (Obj3) and exploit all available data in the IH, a dashboard has been developed, which provides an operational UI (User Interface). Such a dashboard can represent data in different ways according to various port profiles (e.g. environmental manager, gate manager, etc.) and their access rights. Moreover, it encompasses additional features such as notifications, and Operational Tools, which can use the data to manage models and predictive algorithms with the aim of providing **better insights of port operations** (Obj3, Obj4, and Obj5).

The following **use cases** provide some examples of input data that needs to be processed in the PIXEL platform to build useful models and provide relevant insights. Inputs have been identified based on the needs expressed in D3.4 (Use cases and scenarios). Such inputs must be acquired from multiple data sources (port data sources, open data sources, sensors, ...), stored and represented properly in the IH for further processing and visualization.

### 2.1.1. Use case of GPMB

GPMB is mostly concerned about energy management. This implies the usage of models for forecasting short-term energy consumption and predictive algorithms to plan a green energy transition. In order to be able to work with such models and predictions, an initial list of input data has been identified for integration through the DAL in the IH.

*Table 1. List of input data for GPMB*

<b>Data acquisition</b>
Vessel calls Weather information Type of cargo Cargo operation
<b>Data representation</b>
FIWARE Data models Custom FIWARE Data models Extended FIWARE Data models
<b>Data visualization</b>
Single-values Time-series graph Table Text (Recommendation)

## 2.1.2. Use case of Port of Monfalcone

The Port of Monfalcone use case relates to monitoring regional road network to predict short-term traffic volume and to detect and forecast traffic peaks. To be able to work with such models and predictions an initial list of input data has been identified for integration through the DAL in the IH.

*Table 2. List of input data for the Port of Monfalcone*

Data acquisition
Planned arrivals and departures of vessels (in a time interval)
Planned arrivals and departures of trucks (in a time interval)
Real time parking occupancy
Real time situation on port's gates
Weather situation on port premises
Road traffic situation (including temporary traffic changes)
Complexity of loading and unloading activities to estimate the time trucks will be stuck inside port area
Historical data related to arrival and departure of vessels
Truck traffic data coming from SILI (Sistema Informativo Logistico Integrato)
Air quality data openly available from ARPA (Agenzia regionale per la protezione ambientale)
Vessel data coming from PMIS
Machine used for each type of cargo
Type of cargo
Data representation
FIWARE Data models
Custom FIWARE Data models
Extended FIWARE Data models
Data visualization
Single-values
Time-series graph
Table
Georeferenced map
List

### 2.1.3. Use case of Port of Piraeus

The use case of Port of Piraeus is concerned with measuring the pollution (air and noise pollution) that is caused by traffic towards the port. A goal is to improve the access to the port, which can be leveraged by better understanding of traffic volumes, which can be obtained by a predictive algorithm. To be able to work with such models and predictions an initial list of input data has been identified for integration through the DAL in the IH.

*Table 3. List of input data for the Port of Piraeus*

<b>Data acquisition</b>
Weather information (including wind) Information on air pollution sources (types of pollutants, emission rates, etc.) Noise information Vessel calls Traffic outside the port (nearby)
<b>Data representation</b>
FIWARE Data models Custom FIWARE Data models Extended FIWARE Data models
<b>Data visualization</b>
Single-values Time-series graph Table Permanent state graph

## 2.1.4. Use case of Port of Thessaloniki

The use case of Thessaloniki is similar to PPA and is mainly interested in measuring pollution (air and noise pollution). Knowledge about traffic prediction will be obtained by a predictive algorithm. To be able to work with such models and predictions an initial list of input data has been identified for integration through the DAL in the IH.

*Table 4. List of input data for the Port of Thessaloniki*

<b>Data acquisition</b>
Entry/exit of trucks at gates (16, 10A) Vessel calls Type and quantity of cargo Traffic outside the port (nearby) Information on air pollution sources (types of pollutants, emission rates, etc.) Weather information (including wind) Noise information Fuel/electricity consumption Spatial data (location of sensors, etc.)
<b>Data representation</b>
FIWARE Data models Custom FIWARE Data models Extended FIWARE Data models
<b>Data visualization</b>
Single-values Time-series graph Table Permanent state graph



## 2.2. Relation to requirements

Requirements, gathered in WP3, that are related to the ICT framework are one of the main drivers of development of the PIXEL platform. In this deliverable, we provide a full list of requirements directly or indirectly related to the implementation of PIXEL software components (see Table 5). Keeping an up-to-date list of requirements and their relation to WP6 ensure that PIXEL has a user-centric focus on real needs of port stakeholders.

*Table 5. Relation to PIXEL requirements*

Requirement	Addressed in software modules
<b>Common functional requirements</b>	
historical data [36]	PIXEL platform
Interaction with models [41]	PIXEL platform
Anomaly and event list [44]	PIXEL platform
Anomaly and event detection [45]	PIXEL platform
Homogenize Data [61]	PIXEL platform
Catalogue of models [62]	PIXEL platform
Detection of anomalies [63]	PIXEL platform
Feedback [64]	PIXEL platform
Centralized user administration system [65]	PIXEL platform
Configurable Dashboard [66]	PIXEL platform
UI Notification System [67]	PIXEL platform
Port Operational KPI list [70]	IDN <sup>1</sup> , OT <sup>2</sup>
Operational Interface [71]	PIXEL platform
Analyse historical data [81]	IH <sup>3</sup> , OT
Support for manually provided data [86]	IH, DAL <sup>4</sup>
Discovery service for data [104]	PIXEL platform
Visualization of data [105]	IDN, OT
<b>Port of Bordeaux – Energy Management Use Case</b>	
Support electricity consumption sensors [9]	DAL
Access to traffic data [10]	IH
Monitor expected port calls [11]	PA <sup>5</sup> (maritime traffic)
Collect sensor data through Port Community System (VIGIEsip) [12]	IH, DAL
Support Air Quality Sensors [14]	IH, DAL
Modelling and analysis of energy consumption during ship handling procedures [15]	MO <sup>6</sup> (energy demand), PA (energy)
Support wind speed sensors [16]	IH, DAL
Support weather sensor/service [17]	IH, DAL
Support old sensors (gauge stations network) [18]	DAL
Optimization of photovoltaic energy production and consumption [19]	PA (energy)
Monitoring l'Ostrea dredge environmental impact [20]	PEI, IH
Monitor energy consumption of the port authority [22]	DAL, PA (energy)
Expose data to VIGIEsip system [82]	IH

<sup>1</sup> IDN: Integrated Dashboard and Notifications

<sup>2</sup> OT: Operational Tools

<sup>3</sup> IH: Information Hub

<sup>4</sup> DAL: Data Acquisition Layer

<sup>5</sup> PA: Predictive Algorithms resulting from WP4

<sup>6</sup> MO: Models resulting from WP4

Requirement	Addressed in software modules
<b>Port of Monfalcone – SDAG – Intermodal Transport Use Case</b>	
Integration with the SILI Information System [23]	IH, DAL
Integration with the PMIS Information System [24]	IH, DAL
Integration with ASPM <sup>7</sup> video monitoring system [25]	IH, DAL
Traffic peak and congestion monitoring at the port facility [26]	MO (transport)
Integration with the SDAG Access Control System [27]	IH, DAL
Integration with data provided by sensors, cameras and feeds by third parties [28]	DAL
Cooperation with railway authorities [29]	IH, DAL
Provide a common access for management and monitoring of ADR <sup>8</sup> [30]	MO (transport)
Truck re-routing alerting system for operators [31]	IDN&N, OT
Truck re-routing alerting system for final users [32]	IDN&N, OT
Truck re-routing booking system [33]	IDN&N, OT
Port congestion forecasting [34]	PA (road traffic)
Port congestion simulation [35]	MO (transport)
Port - SDAG highway congestion forecasting [37]	MO (transport), PA (road traffic)
<b>Port of Thessaloniki – Port City Integration Use Case</b>	
Support noise sensors and data [50]	IH, DAL
Support real-time fuel consumption sensors [51]	IH, DAL
Support real-time gate surveillance sensors [52]	IH, DAL
Support wind and weather data provided by third party [53]	IH, DAL
Support air quality data provided by third party [54]	IH, DAL
Support traffic data provided by third party [55]	IH, DAL
Estimate noise pollution impact of handling cargo [57]	MO (pollution)
Estimate air pollution impact of bulk cargo operations [58]	MO (pollution)
Visualize the traffic status [106]	IDN, OT
Visualize air pollution [107]	IDN, OT
<b>Port of Piraeus – Port City Integration Use Case</b>	
Support air quality sensors [73]*	DAL
Support water quality data [75]*	DAL
Integration with the PMIS SPARC N4 [76]	IH, DAL
Estimate air pollution impact of cruise and passengers ships related activities [78]	MO (pollution)
Measure real-time air pollution impact of cruise and passengers ships related activities [79]	MO (pollution)
Support noise sensors and data [87]	DAL
Support pollution and traffic data provided by third party [88]	IH, DAL
Measure real-time noise pollution impact of cargo ships related activities [89]*	MO (pollution)
Estimate noise pollution impact of cargo ships related activities [90]	MO (pollution)
Port - City road congestion forecasting [91]	PA (road traffic)

\* Proper handling of requirements [73], [75] and [89] is still under discussion in WP4 and WP5. There is a possibility that an alternative approach will be used.

<sup>7</sup> ASPM: Azienda Speciale per il Porto di Monfalcone

<sup>8</sup> ADR: European Agreement concerning the International Carriage of Dangerous Goods by Road

### 3. Technological state of the art

An in-depth state-of-the-art (SotA) analysis for each PIXEL component has been reported in D6.1. The objective of it was to provide a list of technology candidates for the fulfilment of specific requirements. In D6.3, we take this analysis further by selecting the best candidates and reporting about their implementation status at M16.

#### 3.1. PIXEL Data Acquisition

A state-of-the-art analysis and technological choices for the implementation of the PIXEL Data Acquisition Core layer have been extensively elaborated in D6.1. In this section, we provide a summary of the analysis provided in section 7.1 of D6.1. in addition, the status of implementation for this first software release.

*Table 6. Technological SotA: Data Acquisition Core*

Approach/component	D6.1 choice
Context Broker	FIWARE Orion Context Broker
<b>Status:</b> The FIWARE Orion Context provides the mechanism and API to connect data sources to the IH.	
Persistent Data Hub	FIWARE Cygnus
<b>Status:</b> FIWARE Cygnus provides the mechanism to queue the event fired by The Orion Context Broker to the IH and Short-Term Historic solution. Depending on the integration approach with the IH, FIWARE Cygnus may be deployed, or a similar feature embedded in the IH will be used.	
Short Term History	FIWARE Comet or Quantum Leap
<b>Status:</b> The Short Term History provides the mechanisms to have statistics on the most recent data imported in the Context Broker. This component is going to be deployed in cases where this functionality is needed.	
External Data Sources Connectors	NGSI Agent
<b>Status:</b> An NGSI Agent is a piece of software that will be used to import external data sources into the PIXEL data hub through the DAL. Which NGSI Agents are to be used for this task depends on the list of the data sources and communication protocols identified for each use case. <i>A full list of NGSI agents, data sources and communication protocols is to be provided in D6.2.</i>	
NGSI Agents prototyping solution	none
<p><i>This is a new need identified during the execution of T6.2 and will be fully elaborated in D6.2 as it concerns a generic approach to prototyping.</i></p> <p><b>Status:</b> Implementing the NGSI agents is not a difficult task. Each agent is a simple software that reads data from the data source converts it in the right format and pushes it into the DAL. The main problem is that not all the data sources are currently available and the definition of some common data formats, which would match all our needs, is not easily defined.</p> <p>Most data sources are not fully designed and exposed through a strong API. To allow the other parts of the project to make use of the data, NGSI Agents must be developed or modified as soon as possible. To test the end-to-end solution as quickly as possible, we propose a system that would allow fast prototyping of NGSI agents. To speed up the integration process, it is essential that the data import process can be validated quickly, and the data is made available to the other parts of the system. We found that several technologies based on Mashup interfaces could achieve this goal. Once the process is verified, it will be possible to develop quickly a robust and secure agent for each data source. Currently two solutions are under scrutiny: Node RED and NIFI.</p>	

## 3.2. PIXEL Information Hub

A state-of-the-art analysis as well as technological choices for the implementation of the PIXEL Information Hub have been extensively elaborated in D6.1. In this section, we provide a summary of the analysis provided in section 7.2 of D6.1. in addition, the status of implementation for this first software release.

*Table 7. Technological SotA: PIXEL Information Hub*

Approach/component	D6.1 choice
IH architecture approaches	Message Broker architectural pattern
<b>Status:</b> This architectural approach, combined with the selection of Apache Kafka as its implementation, has shown excellent results during initial developments and testing.	
Data/message broker	Apache Kafka
<b>Status:</b> Apache Kafka is the implementation of the Message Broker architectural pattern. It has shown good performance during initial tests.	
Distributed coordination system	Zookeeper
<b>Status:</b> Zookeeper is required when using Apache Kafka. During initial developments, we confirmed it is a reliable and stable component. To avoid additional complexity, it will be used as a central storage for IH configuration.	
Storage	Elasticsearch
<b>Status:</b> Elasticsearch has been selected as a common data store for the IH, Operational Tools, Dashboard and Notifications. In order to facilitate integration among components and avoid compatibility issues, version 7.2 will be used for all components.	

## 3.3. PIXEL Operational Tools

A state-of-the-art analysis as well as technological choices for the implementation of the PIXEL Operational Tools have been elaborated in D6.1. In this section, we provide a summary of the analysis provided in section 7.3 of D6.1. in addition, the status of implementation for this first software release.

*Table 8. Technological SotA: PIXEL Operational Tools*

Approach/component	D6.1 choice
Operational architecture approaches	Custom development
<b>Status:</b> The Operational Tools are intended to encapsulate various types of models and predictive algorithms, integrate them in a dashboard and use the IH, all developed within the PIXEL project. It is difficult to build it upon existing software tools, except for the CEP (Complex Event Processing). Custom scripts and interfaces will be developed. Models and predictive algorithms will also be converted into services prior to publication in the Operational Tools, but it will depend on the way (language) they are supplied from WP4.	
CEP	Siddhi (WSO2 Complex Event Processor)
<b>Status:</b> The Grafana and Kibana plugins, used in the dashboard, are also able to provide basic CEP functionalities. Considering that, ports have not shown any complex rule, and in order not to confuse	

Approach/component	D6.1 choice
users with different tools, effort will be placed on Grafana and Kibana to provide the needed CEP functionality. If necessary, Siddhi (WSO2) will be incorporated in the final release (D6.4)	
Operational Tools UI	Vuetify
<b>Status:</b> The user interface of the Operational Tools (management, models and predictive algorithms) must be integrated as part of the PIXEL Dashboard, based on VUE and Vuetify. There is a UI for each model/PA and, additionally, a UI for publication of models and PAs.	
Storage	Mongo, Elasticsearch
<b>Status:</b> Elasticsearch has been selected as a common data store for the IH, Operational Tools, Dashboard and Notifications. To facilitate integration among components and avoid compatibility issues, version 7.2 will be used for all components. Some development (linked with the PIXEL data model) has been developed with Mongo in its initial version.	

### 3.4. PIXEL Integrated Dashboard and Notification

This module has been analysed in deliverable D6.1 as part of the initial architecture description. Part of this analysis was a state-of-the-art research of involved technologies. The following table shows the main components and the technological choices done so far.

*Table 9. Technological SotA: PIXEL Integrated Dashboard and Notification*

Approach/component	D6.1 choice
Dashboard tool	Kibana and Grafana
<b>Status:</b> In the initial phase of the project, two tools have been selected to provide off-the-shelf visualizations and to power the PIXEL custom dashboards. They are complementary as Kibana is more focused on data exploration of time series and Grafana is better with static data.	
Notifications tool	Grafana / Elastalert
<b>Status:</b> According to the selected dashboard tools, there are currently two notification engines running. These are unified in the web portal, which makes this differentiation merely a technical aspect, transparent for the user. Kibana brings its own built-in notification system whereas Elastalert is an external plugin for Elasticsearch.	
GIS (Geographic information system)	Custom development - leaflet.js, vue.js/vuetify, Kibana and Grafana
<b>Status:</b> The library leaflet has been selected for the base map representation, while the real time representation of data is implemented by integrating with Kibana/Grafana visualizations. The GIS component needs to be very flexible and adapted to the project needs, so the technical decision has been to do it from scratch by using libraries and integrations with other components.	
Storage	Elasticsearch
<b>Status:</b> Elasticsearch has been selected as a common data store for the IH, Operational Tools, Dashboard and Notifications. In order to facilitate integration among components and avoid compatibility issues, version 7.2 will be used for all components.	
Web portal	Custom development - vue.js/vuetify

Approach/component	D6.1 choice
<b>Status:</b> The web portal is the entry point and main interface for most PIXEL users. Customization development is made based on the web framework vue.js and vuetify. These frameworks combine a good learning curve with a great flexibility and support from the community, helping in the development of project-fitted features.	

### 3.5. PIXEL Security and Privacy

A state-of-the-art analysis as well as technological choices for the implementation of PIXEL DAL have been extensively elaborated in D6.1. In this section, we provide a summary of the analysis provided in section 7.5 of D6.1. in addition, the status of implementation for this first software release.

*Table 10. Technological SotA: PIXEL Security and Privacy*

Approach/component	D6.1 choice
Identity Manager	FIWARE Keyrock
<b>Status:</b> FIWARE Keyrock is fully deployed on the integration platform.	
API Rules Management	FIWARE AuthZForce
<b>Status:</b> FIWARE AuthZForce allows implementing URL (Universal Resource Locator) based rules on role associated to OAuth2 identity. The component is not yet deployed as the needs not clearly defined.	
Security Proxy	FIWARE Wilma
<b>Status:</b> FIWARE Wilma is deployed with a nginx proxy in order to offer an OAuth2 based security gateway for the API access.	



## 4. PIXEL platform prototype

The PIXEL project aims to provide a software solution for using IoT devices and IT data sources to an affordably and effectively exploit data in the ports of the future. These two characteristics, affordability and effectiveness, are key influences when thinking about technological choices and the development of the project. Open sourced with friendly licensing projects are selected to build up the main blocks while specific code is developed to implement the strategic features of the platform. The following sections describe the technologies developed so far.

### 4.1. PIXEL software components

Development of these components has been done following the architectural criteria and technical analysis reported in D6.1. That document shows how the different PIXEL modules relate together and provide features to fulfil PIXEL requirements and achieve global as well as particular objectives. The following subsections report about the status of development of the different modules.

#### 4.1.1. PIXEL Data Acquisition

As described in the D6.1 the DAL relies on the *FIWARE* based architecture. The main components come from the **FIWARE catalogue**<sup>9</sup> and they have been specifically deployed and configured for PIXEL needs.

NGSI Agents for the Data Acquisition have been developed specifically for PIXEL needs. They are used to import data from available port data sources to the FIWARE Context Broker, before pushing them into the IH.

- Two NGSI agents have been already developed:
  - *WindData* for Thessaloniki, where data is pushed using a custom data model;
  - *VesselCall* activity for Bordeaux;
- Two NGSI agents have been prototyped using the NIFI platform:
  - *WindData* for Thessaloniki where data is pushed using WeatherObserved;
  - *VesselCall* for Thessaloniki where data is pushed using a custom data model;
- A prototype of an AIS receiver, using a Raspberry PI, which is able to push data to the PIXEL DAL, has been developed and will be tested in Bordeaux.
- The NIFI solution has been packaged using docker in order to deploy it on the PIXEL infrastructure.

#### 4.1.2. PIXEL Information Hub

D6.1, Section 4.1 Global architecture describes the global functional architecture of PIXEL, with relations among different PIXEL components. In this architecture the PIXEL Information Hub acts as a central integration component for data sharing among PIXEL components, as it implements the Information domain of the Reference Architecture (see D6.1 Figure 10: Equivalence from RAMI and IIRA architecture to PIXEL RA).

As stated in D6.1, the main architectural approach for the PIXEL Information Hub is based on the robust experience gained by XLAB during the design and implementation of a similar technical solution, FAIR (Facility for Antiproton and Ion Research) Archiving System, for the FAIR particle accelerator based in Darmstadt, Germany. XLAB carried out a technical and legal feasibility assessment in order to identify if any of the components built for the FAIR Archiving System project could be adapted for use in PIXEL.

The conclusion is that the core of the Archiving System partially fits functional and non-functional requirements of PIXEL, although there are parts of the system that have to be adapted to specifics of port environments. Furthermore, data collector/data proxy components have to be developed to support PIXEL data sources.

---

<sup>9</sup> <https://www.fiware.org/developers/catalogue/>

The following functionalities have been developed for the PIXEL project:

- *Docker scripts*: a set of Docker Compose scripts have been created for the packaging and deployment of the IH and all the dependencies (Kafka, Zookeeper, Elasticsearch, Kibana, FAIR Archiving system).
- Following the notice from Oracle about changes in licensing of Java products, the system has been adapted to the use of *OpenJDK*. The reason is that *OpenJDK* offers the same set of functionalities while retaining the PIXEL commitment to an open-source approach.
- The system has been upgraded to use Elasticsearch 7.2. This upgrade required some code changes, as the new version is not fully backwards compatible. The upgrade also represents a first step towards the integration of different PIXEL software components, because the IH, Dashboard, Notifications and Operational Tools will all directly access the common PIXEL data store.
- LTS (Long Term Storage) update functionality has been added. The Archiving System core did not have functionality for the update of existing records. This functionality is needed in several PIXEL scenarios (for example updating vessel departure time of a port call), so it has been added to the core system.
- A specific PIXEL DataCollector component is under development. It already provides functionality for AIS data collection as well as basic integration capabilities with DAL. Integration with DAL was a crucial step to start testing end-to-end data transfer in PIXEL. The integration is being further expanded to allow automatic detection of new data sources, conversion of data types and some additional PIXEL-specific functionalities.
- FLOAT and GEOLOCATION datatypes have been added to the Archiving System Core. Geolocation datatype allows easier computation of geo-related queries, as well as usage of ready-made visualisation tools.

### 4.1.3. PIXEL Operational Tools

OTs (Operational Tools) will allow operators to use high-level models and predictive algorithms (PAs) directly from within the dashboard. In this stage, the development focus has been on the development of a generic OTs engine that would allow executing different tools resulting from WP4 and data collected through activities in WP7. In the next phase, by the release of integrated PIXEL components (WP6/D6.4), finalised integration activities (WP7) and final version of models and PAs (WP4), the full potential of OTs will be exploited.

For this first OTs release, we have taken the following approach:

- A basic OT model is used as a proof-of-concept to describe the different steps.
- Static synthetic input data (JSON files) have been used in order to facilitate the development and repeatability of unit tests.
- A basic UI prototype derived from the basic OT model has already been developed in order to provide a template for future dashboard developments.

For the final release, to be published in D6.4, OTs will provide the following functionalities:

- In the second phase, after successful module testing and with finalised integration activities, data collected through the IH will be used for OTs operation. One important step to complete before the final integration is the finalisation of data models, which are still under development.
- The final OTs version will use models and predictive algorithms resulting from WP4.
- The Dashboard section the OTs will fully support specific models and PAs resulting from WP4.

The Operational Tools distinguishes three main phases: (i) publication of a model or PA, (ii) execution (or scheduling) of the model or PA, and (iii) visualization of the results. They are briefly commented below.

#### Publication of a model or PA

This step assumes that the model (or PA) has already been implemented in a particular language. Through the process of publication PIXEL, models (and PAs) will potentially be accessible to all internal components as a



backend service. The definition of a service within the PIXEL data model also allows establishing access permission to different profiles (linked with security). Each model/PA has two mandatory fields:

- *Inputs* (JSON files): represents data obtained from the IH. For this release, as commented we will use static JSON files (that the user may even edit/change);
- *Outputs* (JSON files): represents data stored in the IH that will be later represented in the dashboard (Kibana, Grafana);

The publication API has been implemented in Java, particularly with the JAX-RS API, that facilitates the inclusion of annotations and development of Swagger interfaces (useful for D6.5). The initial database employed was Mongo but there is an ongoing discussion to shift (part of) it into Elasticsearch for smoother integration with the IH and dashboard. This will be reflected in the next version.

### Execution of a model or PA

Models and predictive algorithms can be executed in three different ways:

- *Scheduled as a service (e.g. every hour, every day)*. The input is obtained automatically (e.g. current weather, daily vessel calls) and the output is stored in a database that users can request without a need for several unnecessary executions. Following the basic model regarding energy, every day the consumption is forecasted with the daily vessel calls and the weather forecast. Scheduled models (services) will probably be the most common ones. A script or periodic process is responsible for getting the list of scheduled services from the data model, obtain the necessary input and invoke the service through its corresponding endpoint, and finally store the output in the database. Initially, it was implemented in Mongo, but it will probably be ported to Elasticsearch.
- *Manually from the user*. This corresponds to a special case where the user explicitly requests an execution to obtain a response, maybe because (i) it requires the latest updated info or (ii) the inputs have been changed by the user to analyse a what-if scenario. In this case, the results are not typically stored in the database, as they will be shown on the dashboard directly to the user.
- *Triggered by a certain event*. It is also possible to execute the model or PA whenever a certain event occurs. For example, if an unexpected vessel is going to berth at the current day, this may trigger a new recalculation of energy consumption forecast, updating the result of a scheduled service.

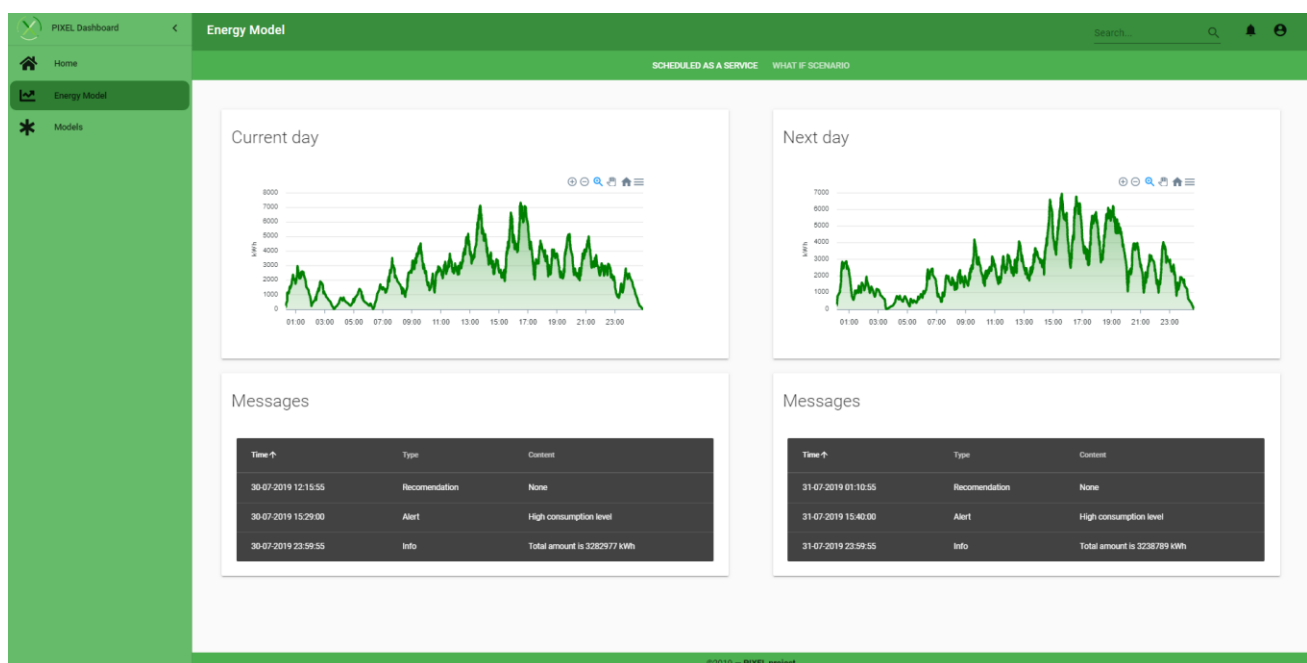


Figure 1. PIXEL energy model dashboard

## Visualization of results

Once the model or PA has been executed, the results, at least the scheduled ones, are stored in a database so that they can be visualized from the dashboard. Obviously, the output of each model/PA should satisfy certain properties to be smoothly integrated into Kibana and Grafana plugins in the dashboard.

Kibana will smoothly access the data if it is stored in Elasticsearch. Grafana is much more flexible and allows for several possibilities; for the basic model, the Grafana SimpleJSON plugin was initially proposed as it provides an independent layer of abstraction from the physical database.

Considering the integration with the dashboard, a basic UI has been developed for the basic model, which is aligned with the real energy model. It has been implemented in Vuetify, in order to allow a smooth integration with the dashboard, which follows the same approach (VUE and Vuetify).

### 4.1.4.PIXEL Integrated Dashboard and Notification

A functional demo version of the component has been developed, in order to present the functionality to the rest of the partners, with special interest in the participant ports, the proposal of UI for PIXEL, including dashboards, PEI representation, notification configuration and GIS. A first demo version was available coinciding with the first advisory board meeting (22nd May 2019). The following features have been developed at the date of publication:

- **General access:** login and access to the application. The login is integrated with the security component, based on the FIWARE security stack.
- **General navigation:** structure of the application, menus and buttons for overall navigability.
- **Summary dashboards:** a summary view of the dashboard for each port, configurable by the port application administrator. These dashboards can have mixed origins (Kibana, Grafana).
- **Initial notification configuration UI:** Alerts and notifications configuration UI, which is integrated with the backend dashboards. Configuration of the notification channels.
- **PEI representation mock-up:** Initial design of PEI shows the different visual indicators and the legend to interpret them.
- **Models and entities representation:** Models and entities integrated into the platform, which are specific for each port. The *models* view shows a repository of the different models available in the OT components, allowing basic operations such as start/stop a simulation. *Entities* is a list of the different data models integrated in the PIXEL instance.
- **GIS view:** Representation of static/dynamic sensors and devices on a map. Detail of the different measures of each piece of equipment, access to visualizations/alerts.
- **Visualization catalogue:** List of the different visualizations created for the port scenario analytics.
- **Data management view:** Administration list for development/maintenance purposes showing different installed instances and the data sources (NGSI Agents).
- **Security view:** Management of users, devices, roles and XACML (eXtensible Access Control Markup Language) access rules.

Apart from this, the following features are also available in the current development version:

- **Dockerisation:** All the components have been containerized using Docker.
- **Integration with storage in IH:** Dashboards and IH shares the NoSQL database (Elastic) so the data inserted in the IH is fully available in the dashboards and notification tools.
- **Integration with the security back-end:** The application is fully integrated with the identity manager of the FIWARE stack, so all the platforms security aspects can be managed from PIXEL platform.
- **Integration with back-end dashboard tools:** Web application is connected to Kibana and Grafana dashboard tools so they can provide features (visualization, alerts, navigation of charts) to the application.

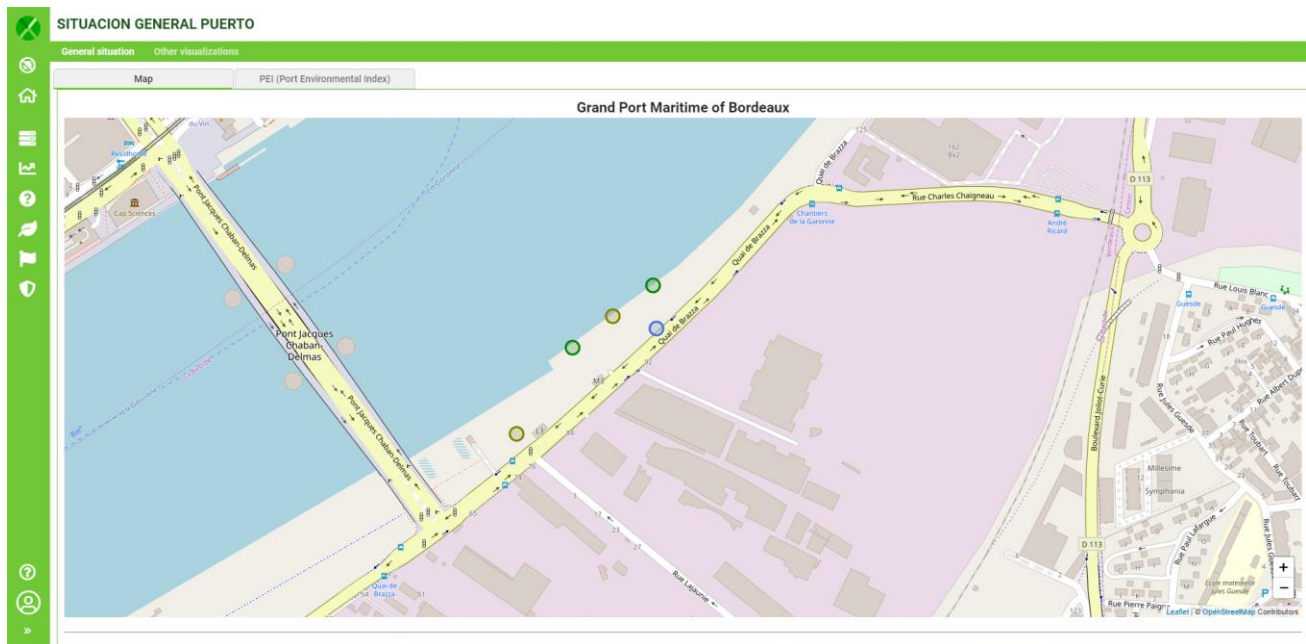


Figure 2. PIXEL Visualisation Example

### 4.1.5. PIXEL Security and Privacy

As described in the D6.1 the security layer relies on a FIWARE based architecture. The main components are from the FIWARE catalogue<sup>10</sup> and they have been specifically deployed and configured for the PIXEL project needs.

FIWARE Keyrock has been deployed in order to provide identity management for the PIXEL platform.

A documentation has been produced to describe how to connect PIXEL components with the FIWARE security solution using the OAuth2 protocol. That documentation covers the security mechanism to connect a data source through the Wilma proxy to the NGSI Agent or the Context Broker and furthermore how the dashboard will be able to use Keyrock to manage users using the FIWARE API or the OAuth2 protocol.

A first draft of UI templates has been studied with the goal to integrate the authentication screens of Keyrock and make them fit the UI design chosen for the dashboard.

The FIWARE AuthZForce has been deployed for test purpose on the Orange development platform to prepare its installation on the integration platform. It will be done shortly and provide a better security solution for Data Acquisition API.

PIXEL exposes API's for internal and external communication. To achieve that goal, we need a security component that can manage the different API Gateway features and adapt them to our needs. An API Gateway solution needs to cover a large set of security features to fit the specific needs of each type of project. The PIXEL API gateway should provide the following capabilities:

- **Identity Control:** The minimum feature we need from the API Gateway is to control the access of each API with an authentication mechanism. For D6.1, we chose the OAuth2 standard protocol to satisfy this need.

<sup>10</sup> <https://www.fiware.org/developers/catalogue/>

- **Access control:** Another important feature expected from an API Gateway is to control, whether a user can call the request he makes. To implement this feature, there are several solutions with different security levels:
  - *EndPoint Access control:* We only check the validity of the *access\_token*, and then the gateway allows the user to access the full API;
  - *VERB+URL Access Control:* The system allows to define the allowed part of the API based on the HTTP verb and the requested URL;
  - *Deep packet introspection:* With this solution the gateway will inspect the content of the request in order to decide if the user is allowed or not to make it;

Choosing the right access level is not only driven by the security aspect, but it also depends on the design of the API you want to protect. If your API has been designed with integrating security patterns, then a simple endpoint access control could be secure enough.

- **API publication:** API publication is a feature provided by the API Gateway to expose a new API to the solution:
  - *Configuration:* To add a new API, we have to change the configuration of the Gateway manually;
  - *Web based interface or API:* The Gateway provides a solution to allow admins to change the configuration settings dynamically;
  - *Self-Publication:* Users can expose their API's using a web-based UI or API, themselves;
- **API Subscription:** API Subscription is the function defining how a user can request access to a given API.
  - *Manual request management:* The admin decides how to handle the request of a new system user.
  - *Auto approbation:* The user can freely request access to some APIs and the system will provide him a portal to manage its credentials.

Several solutions based on FIWARE have been identified:

- **FIWARE Keyrock + Wilma:** This solution offers a simple ready to use OAuth2 access control solution. The API publication is done manually, and we can use Keyrock UI and API to manage subscriptions.
- **FIWARE Keyrock + Wilma + AuthZForce:** By adding AuthZForce to the previous solution, we incorporate a method to check verbs and URLs. The configuration could be done by using Keyrock's API and UI.
- **APIGEE<sup>11</sup>:** Is a product provided by Google (not free, not open source), which provides a complete API management solution with all the features we need.
- **Kong<sup>12</sup>:** "The World's Most Popular Open Source Microservice API Gateway and Platform". A solution that provides most of the feature we can need.
- **APIInf<sup>13</sup>:** Provides a complete solution for API Management, and it is fully integrated with the FIWARE solution<sup>14</sup>;

Based on the above analysis we deployed several solutions as listed below. Further research carried on as part of T6.6. are needed in order to propose the optimal solution to be used for PIXEL.

On the integration platform, we deployed a *Keyrock + Wilma* solution to provide the basic mechanism needed by the DAL in order to protect externally exposed NGSI agent APIs. This offers the basic level of security, where each NGSI agent is exposed by using a different base path. It provides the OAuth2 mechanism each

<sup>11</sup> <https://cloud.google.com/apigee>

<sup>12</sup> <https://konghq.com/kong/>

<sup>13</sup> <https://www.apinf.com/>

<sup>14</sup> <https://www.fiware.org/2017/07/28/apinf-contributes-top-class-api-management-technologies-to-fiware-platform/>

client has to use to access our API and allows that this solution could evolve to integrate new security capabilities in the future.

In the meantime, we have started to test the deployment of *AuthZForce*, which allows extending our configuration and applying strong security rules to ensure sufficient access control before final deployment on the port infrastructure.

We also have started to test the integration of an *APIInf* solution based on FIWARE to evaluate the product integration capabilities.

FIWARE Wilma + nginx has been deployed in order to provide an API Gateway based on OAuth2 protocol; The APIInf solution is currently under test to evaluate its capabilities and integration with FIWARE to determine if it will be possible to use it for the Security Gateway feature.

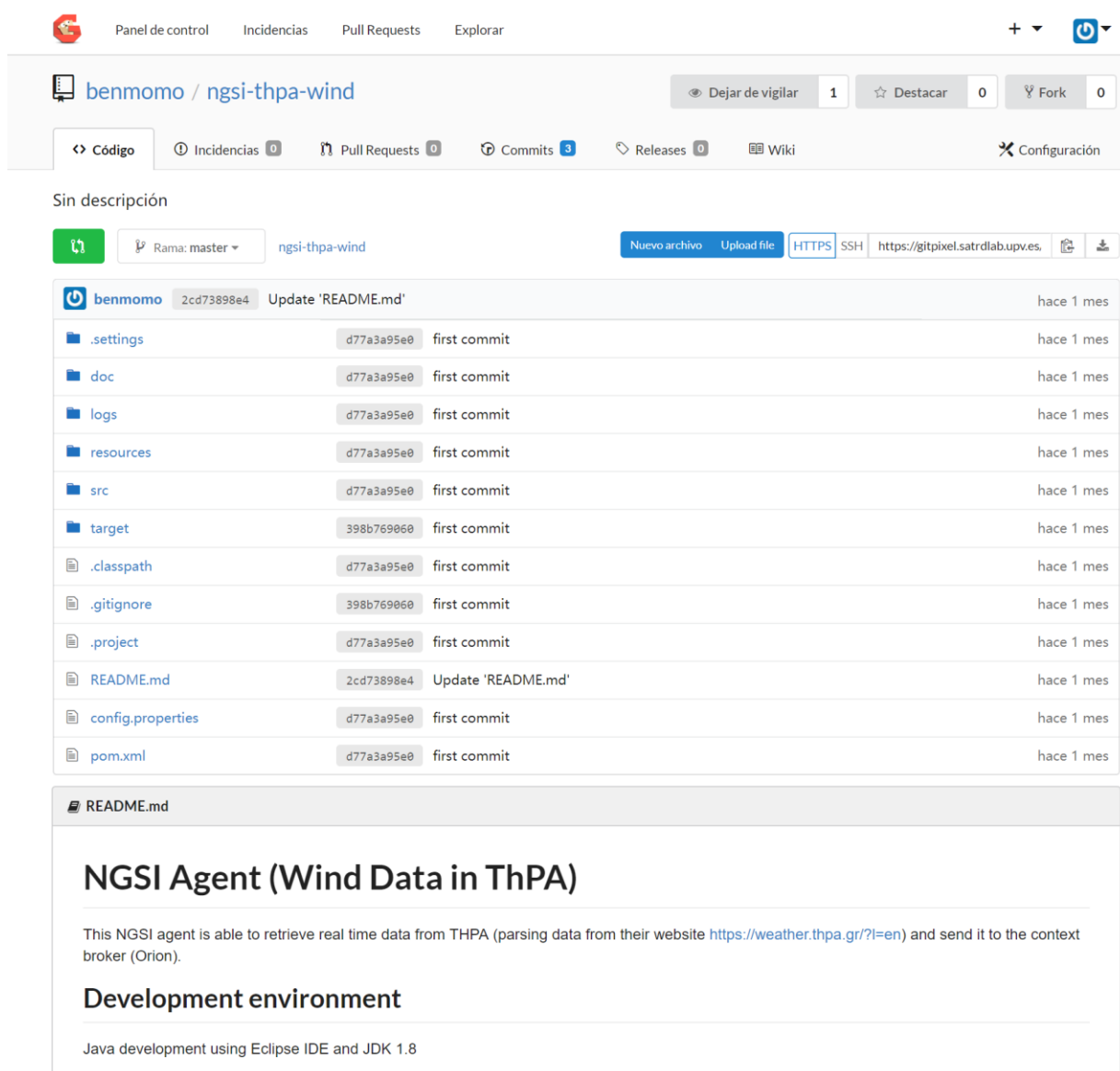
## 4.2. PIXEL development environment

### 4.2.1. Git repository

Git is a distributed version control system commonly used by software developers in order to track changes in source code for different projects. It is mandatory for serious development teams that require easy management of thousands of files. The most known implementation is provided by Github ([www.github.com](http://www.github.com)), but other open implementations can be used at local level, such as Gogs, which is being used in PIXEL.

Gogs (Go Git Service) offers a fully functional git environment, which can be deployed on a company's premises for internal use. It is written in Go language and is open source (MIT license). The software is cross-platform and includes a web interface similar to Github.

Some of the functionalities included are amongst others: management of Git repositories (files, users and access rights), collaborative teamwork in private environments, authentication and secure communication (HTTPS), branch management, etc.



The screenshot shows a Gogs repository interface. At the top, there's a navigation bar with links: Panel de control, Incidencias, Pull Requests, and Explorar. The repository name 'ngsi-thpa-wind' is displayed, along with statistics: 1 watcher, 0 stars, and 0 forks. Below this, there are tabs for Código, Incidencias (0), Pull Requests (0), Commits (3), Releases (0), Wiki, and Configuración. The main content area shows a list of files and their commit history. The files listed are: .settings, doc, logs, resources, src, target, .classpath, .gitignore, .project, README.md, config.properties, and pom.xml. The README.md file is highlighted, showing its content. The README content includes the title 'NGSI Agent (Wind Data in ThPA)', a description of the agent's functionality, and a section for the 'Development environment' which mentions Java development using Eclipse IDE and JDK 1.8.

File	Commit Hash	Commit Message	Time
.settings	d77a3a95e0	first commit	hace 1 mes
doc	d77a3a95e0	first commit	hace 1 mes
logs	d77a3a95e0	first commit	hace 1 mes
resources	d77a3a95e0	first commit	hace 1 mes
src	d77a3a95e0	first commit	hace 1 mes
target	398b769060	first commit	hace 1 mes
.classpath	d77a3a95e0	first commit	hace 1 mes
.gitignore	398b769060	first commit	hace 1 mes
.project	d77a3a95e0	first commit	hace 1 mes
README.md	2cd73898e4	Update 'README.md'	hace 1 mes
config.properties	d77a3a95e0	first commit	hace 1 mes
pom.xml	d77a3a95e0	first commit	hace 1 mes

**README.md**

## NGSI Agent (Wind Data in ThPA)

This NGSI agent is able to retrieve real time data from THPA (parsing data from their website <https://weather.thpa.gr/?l=en>) and send it to the context broker (Orion).

### Development environment

Java development using Eclipse IDE and JDK 1.8

Figure 3. PIXEL Gogs repository screenshot

## 4.2.2.FIWARE Lab

FIWARE Lab is the non-commercial sandbox environment of the FIWARE Community. It offers the capability to innovate and experiment with the FIWARE Technologies free of charge. Entrepreneurs and individuals can test FIWARE technologies as well as their applications within the FIWARE Lab, with the possibility to exploit Open Data published by cities and other organizations.

FIWARE Lab is deployed over a geographically distributed network of federated FIWARE Lab nodes. Each FIWARE Lab node maps to one, or a network of, data-centres on top of which an OpenStack instance has been deployed, federated and configured as a FIWARE Lab node (Cloud region) operated by a specific organization.

Orange with Images & Réseaux manages one of those nodes (Lannion5) and uses it to provide an integration platform for the PIXEL project. We have allocated enough resources for the deployment of the full PIXEL solution. This platform allows partners to test the integration of their components together.



## Limit Summary

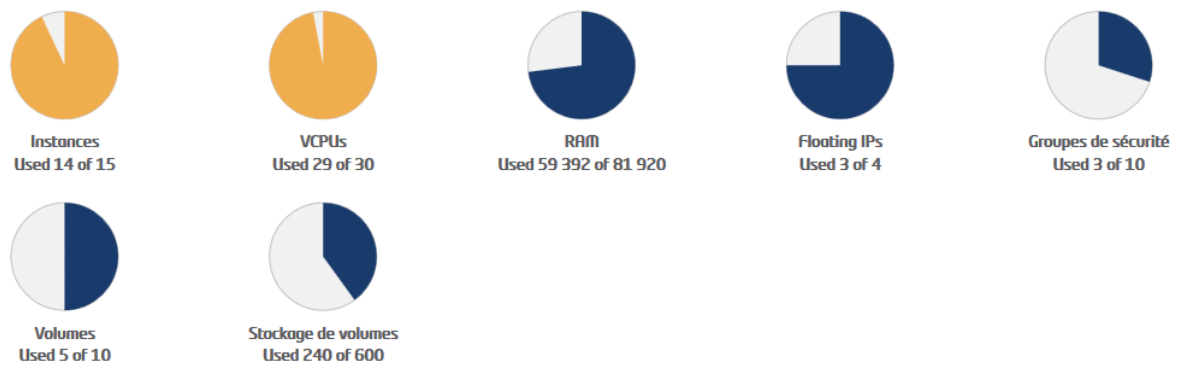


Figure 4. PIXEL virtual environment overview

The various components of the DAL, the IH, operational tools, integrated dashboard and security already available are currently deployed on this platform.

## Usage

[Download CSV Summary](#) [Download Juju Environment File](#)

Instance Name	VCPUs	Disk	RAM	Time since created
Dashboards	1	20GB	2GB	4 months, 3 weeks
ELASTIC	8	160GB	16GB	4 months, 3 weeks
Proxy	1	20GB	2GB	4 months, 2 weeks
bastion	1	20GB	2GB	3 months, 2 weeks
Keyrock	1	20GB	2GB	3 months, 2 weeks
PEPPProxy	1	20GB	2GB	3 months, 2 weeks
ContextBroker	2	40GB	4GB	3 months, 2 weeks
NGSIAgents	1	20GB	2GB	3 months, 1 week
Cygnus	1	20GB	2GB	3 months, 1 week
dbmodel	2	40GB	4GB	2 months, 3 weeks
OT	4	80GB	8GB	2 months, 3 weeks
proxy_thpa	1	20GB	2GB	2 months, 2 weeks
Information Hub	4	80GB	8GB	2 months
Applications	1	20GB	2GB	1 month, 3 weeks

Displaying 14 items of False

Figure 5. PIXEL virtual server instances overview

## 5. Software mechanisms for data acquisition

The DAL uses a simple architecture pattern. We developed a simple software driven NGSI agent, to retrieve the available data from different data sources and convert them in a standard data format, before sending them to the IH.

There are several on-going activities related to data sources integration that are being jointly analysed through several PIXEL work packages:

- **Identification of necessary data needed by WP4 and WP5.** There is an on-going analysis of those sources and best data formats to use in order to store them.
- **Implementation of data access mechanisms.** Some ports like GPMB, already have an API solution exposing parts of their data, while other ports have their data in a proprietary IS where first we need to define a way to access them.
- **Common data model identification.** To import and store the data, we need to identify a common Data Model to use. The main difficulty is that each port uses a different format and timeline to provide the same kind of information like Vessel Call. Some ports provide real time information, planned activity, or report about passed activity. When the majority of data sources are identified, a suitable Data Model is going to be defined.
- **The data quality itself is also another issue, which needs to be addressed.** Some of the required attributes are not yet used for computation, as many ports currently have no need to provide them accurately. We have to analyse this situation to identify how to clean, validate and make the provided data useful.

To be able to import data quickly, we have deployed a NIFI component to prototype the data import process, which allows end to end testing of the full solution. Table 11 provides a list of available NGSI agents and their status. They are all deployed, either as native NGSI agents or as NIFI prototypes.

*Table 11. NGSI agents already available*

Ports	Data Source	Status	Description
Thessaloniki	WindData	Deployed	An NGSI Agent developed by UPV to import Thessaloniki WindData in a custom data format
		NIFI Deployed	An NGSI Agent prototype with NIFI platform to import WindData using FIWARE WeatherObserved data model
	Vessel Reports	NIFI Deployed	An NGSI Agent prototype with NIFI platform to import Vessel reports provide by Thessaloniki using PIXEL VesselCall data model
Bordeaux	Vessel Call	Deployed	An NGSI Agent developed to import Vessel Call activity. This agent listen for connection of a script execute on GPMB side.



Table 12 provides the status of on-going developments related to data acquisition. It lists the development status of identified data sources. Furthermore, an amended list of data sources will be provided in D6.4.

*Table 12. NGSI agents – identified needs and development status*

Port	Data Source	Status	Description
Bordeaux	Fall Forms	Data Model to be defined	Import the Fall Forms information from the Vigiesip API
	Weather	Source to be defined	
	Energy Consumption	Data Model to be defined	Reports are available, still have to find the proper way to import them
	Air Quality Data	Waiting sensor	
	Market price of kWh (electricity)	To Be Defined	Waiting to define source and format
Monfalcone	Vessel Call	To Be Defined	Work in progress on Data Model, waiting for data Endpoint
	Vehicle Activity	To Be Defined	Need to clarify data source endpoint
	Weather	Source to be defined	
	Air Quality Data	Ready to develop	Import data from Open Data source
Piraeus	Air Quality Data	Source to be defined	IoT sensors available, we have to validate how to retrieve their data.
	Noise	Source to be defined	IoT sensors available, we have to validate how to retrieve their data.
	Vehicle Activity	Data Model to be defined	Data available, we are able to prototype NGSI Agent quickly
Thessaloniki	Air Quality Data	Source to be defined	Endpoint to be validated
	Noise	Source to be defined	Endpoint to be validated
	Vehicle Activity	Data Model to be defined	Endpoint to be validated
	Vessel Call	Prototyped	Endpoint to be validated

From the analysis of the identified data sources from each port, we have made the first step to define the main data model for the PIXEL solution. The data model will be based, to the maximum extent possible, on the FIWARE Data Model<sup>15</sup> if one is already available or otherwise a new one created and possibly shared with the community. The initial list of models is provided in Table 13.

*Table 13. PIXEL Data Models*

Data Model	Source	Description
WeatherObserved	FIWARE <sup>16</sup>	An observation of weather conditions at a certain place and time.
AirQualityObserved	FIWARE <sup>17</sup>	An observation of air quality conditions at a certain place and time
WaterQualityObserved	FIWARE <sup>18</sup>	Water Quality data model is intended to represent water quality parameters at a certain water mass (river, lake, sea, etc.) section
NoiseLevelObserved	FIWARE <sup>19</sup>	It represents an observation of those acoustic parameters that estimate noise pressure levels at a certain place and time.
GateAccessObserved	PIXEL	An observation of the type of vehicle going through a gate
VesselCall	PIXEL	Planned an observed activity about a Vessel in a port (FAL <sup>20</sup> Forms)
TrafficFlowObserved	FIWARE <sup>21</sup>	An observation of traffic flow conditions at a certain place and time.
EnergyConsumptionObserved	PIXEL	An observation of energy consumption at a certain place (equipment, tools, building ...) and time.

Those Data Models will be more detailed in future PIXEL deliverables.

<sup>15</sup> <https://fiware-datamodels.readthedocs.io/en/latest/index.html>

<sup>16</sup> <https://fiware-datamodels.readthedocs.io/en/latest/Weather/WeatherObserved/doc/spec/index.html>

<sup>17</sup> <https://fiware-datamodels.readthedocs.io/en/latest/Environment/AirQualityObserved/doc/spec/index.html>

<sup>18</sup> <https://fiware-datamodels.readthedocs.io/en/latest/Environment/WaterQualityObserved/doc/spec/index.html>

<sup>19</sup> <https://fiware-datamodels.readthedocs.io/en/latest/Environment/NoiseLevelObserved/doc/spec/index.html>

<sup>20</sup> Convention on Facilitation of International Maritime Traffic

<sup>21</sup> <https://fiware-datamodels.readthedocs.io/en/latest/Transportation/TrafficFlowObserved/doc/spec/index.html>

## 6. Service performance in the port area

This section describes the approach to handle performance evaluation within PIXEL in terms of establishing and measuring KPIs, while at the same time creating a technical foundation to measure the more technical KPIs (like performance efficiency). The de-facto measurements of KPIs is targeted in WP7 and WP8.

Performance is an ambiguous term, which depends on the various actors involved (networking department, project managers, developers, business analysts, etc.) as well as the target aspect under study (operational performance, functional performance, process performance, business performance). Therefore, it is important to (i) clearly define what is relevant in terms of performance for the PIXEL project and (ii) align the efforts across the different work packages. Such a task was already performed and drafted in D8.1 from two different perspectives:

- Technical perspective (technical assessment)
- Business perspective (business assessment)

Since WP6 is fundamentally a technical Work Package, this section focuses on defining the measurement approach (ie. how exactly will the measurement be performed) of the technical KPIs in D8.1 for each of the software components developed in WP6. This document lists global KPIs (extracted from the BigDataOcean Validation Framework) which are assigned to different software components developed within tasks of WP4 and WP6.

### 6.1. PIXEL Data Acquisition

In Table 14 below, KPI's assigned to T6.2 (Data Acquisition) are described together with the methodology to be followed. Most of the software components of the DAL come from the FIWARE catalogue. Those components have complete quality tests coverage.<sup>22</sup>

*Table 14. KPIs measurement approach: PIXEL Data Acquisition*

KPI	Measurement approach
<b>Functional suitability</b>	
Straightforward task accomplishment	A process to add a new data sources will be analysed to verify that the process does not include unnecessary steps. Boolean response (Yes/No)
The portion of completed requirements	“Should have” and “Must have” requirements from deliverable D3.2 will be taken as input in order to extract all requirements specifically targeting T6.2.
<b>Performance efficiency</b>	
Maximum number of connected data sources	An NGSI Agent is deployed for each connected data source. The number of deployed agents provides the number of connected data sources. Tests will be performed to determine the maximum number.
Maximum database size	The database size will be calculated based on the maximum number and type of connected data sources.

<sup>22</sup> <https://www.fiware.org/developers/qa-labeling-of-fiware-components/testing-fiware-ges/>

KPI	Measurement approach
Average latency	Requests to the PIXEL Data Acquisition will be launched with <b>JMeter</b> and Average response time will be measured. The frequency and number of simultaneous requests will be gradually increased in order to determine DA performance under low/medium/high workload.
Throughput	Requests to the PIXEL Data Acquisition will be launched with <b>JMeter</b> . The frequency, size and number of simultaneous requests will be gradually increased in order to determine IH performance under low/medium/high workload.
Mean CPU Utilisation	The same approach as for average latency is valid. In order to get the mean CPU usage, the <b>PerfMon JMeter plugin</b> <sup>23</sup> will be used for the same JMeter tests. <b>PerfMon</b> is a JMeter package for server monitoring. It can monitor CPU, Memory, Swap, Disks I/O and Networks I/O.
Mean memory usage	The same approach as for average latency is valid. In order to get the mean memory usage, the <b>PerfMon</b> library will be used for the same JMeter tests.
Maximum memory usage	The same approach as for average latency is valid. In order to get the maximum memory usage, the <b>PerfMon JMeter plugin</b> will be used for the same JMeter tests.
Maximum processing power used	The same approach as for average latency is valid. In order to get the maximum CPU usage, the <b>PerfMon JMeter plugin</b> will be used for the same JMeter tests.
<b>Compatibility</b>	
% of APIs coverage	A list of all identified systems in ports exporting or consuming data will be compiled from user requirements and activities in other WPs. Then, this list will be compared against systems that have been successfully integrated in PIXEL (WP7).
Ability to acquire data from different data formats	Similar to the approach for % of APIs coverage, two lists will be compiled and compared: all identified data formats vs. supported data formats.
Ability to support different IoT platforms	Similar to the approach for % of APIs coverage, two lists will be compiled and compared: all identified IoT platforms vs. supported platforms.
Ability to export different data formats	Similar to the approach for % of APIs coverage, two lists will be compiled and compared: all identified data formats vs. supported data formats.
<b>Reliability</b>	

<sup>23</sup> <https://jmeter-plugins.org/wiki/PerfMon/>

KPI	Measurement approach
Simultaneous requests	The same approach as for average latency is valid and <b>JMeter</b> will be used. Here JMeter probes will be defined to increase the number of concurrent requests progressively until the load arrives at a certain threshold.
% Monthly availability	The FIWARE Context Broker provide Metrics API <sup>24</sup> and Statistics API <sup>25</sup> we can rely on for this purpose: <i>uptime_in_secs</i> , Orion uptime in seconds.
Success rate	The FIWARE Context Broker provide Metrics API and Statistics API we can rely on for this purpose: <b>incomingTransactionErrors</b> : number of incoming transactions resulting in error.
<b>Maintainability</b>	
% of modularity	Will be measured by reporting all the independent components that are part of the data acquisition module and comparing them to the number of all components in the data acquisition module. Individual operation means that a component can offer a complete function with meaningful information in the context of PIXEL.
% of reusable assets	Will be measured by reporting all the reusable components that are part of the data acquisition module and comparing them to the number of all components in the data acquisition module. A reusable component is considered any that can be applied in a different context of PIXEL with no modifications of the source code.
% of update	Will be measured by reporting the level of success in software updates on the data acquisition module. It compares successfully completed updates versus all executed updates.
Level of analysability	Will be measured by reporting the ratio between the number of items inside the data acquisition module for which logging is implemented compared to the number of items for which the specifications require logging.
<b>Portability</b>	
Mean number of errors per hardware or OS (Operating System) change/ upgrade	Will be measured by analysing the system & application error logs.
Mean number of errors per software change/ update	Will be measured by analysing the system & application error logs.
Mean number of errors per software install	Will be measured by analysing the system & application error logs.

<sup>24</sup> [https://fiware-orion.readthedocs.io/en/master/admin/metrics\\_api/index.html](https://fiware-orion.readthedocs.io/en/master/admin/metrics_api/index.html)

<sup>25</sup> <https://fiware-orion.readthedocs.io/en/master/admin/statistics/index.html>

KPI	Measurement approach
Mean number of errors per software uninstall	Will be measured by analysing the system & application error logs.

## 6.2. PIXEL Information Hub

In Table 15 below, KPI's assigned to T6.3 (PIXEL Information Hub) are described together with the methodology to be followed.

**Performance efficiency** and **reliability** (partially) will be measured using Apache JMeter and a set of helper libraries. The Apache JMeter™ application is an open-source software designed to load test functional behaviour and measure performance. In order to assess the performance in the port area, measurements will be performed with a predefined set of realistic input data relevant to port operations. In the beginning, all measurements will be performed under laboratory conditions, and on the infrastructure, which will be defined in WP7 (cloud environment v.s. on-premises installation and other parameters).

*Table 15. KPIs measurement approach: PIXEL Information Hub*

KPI	Measurement approach
<b>Functional suitability</b>	
Straightforward task accomplishment	A process to add a new data sources and the process to provide data (data extractor) will be analysed to verify that the process does not include unnecessary steps. Boolean response (Yes/No)
The portion of completed requirements	“Should have” and “Must have” requirements from deliverable D3.2 will be taken as input in order to extract all requirements specifically targeting T6.3.
<b>Performance efficiency</b>	
Maximum number of connected data sources	The number of data sources is the active number of sources connected through the DAL. The maximum number will be determined by gradually activating available sources.
Maximum database size	The database size will be calculated based on the maximum number and type of connected data sources.
Average latency	Requests to the PIXEL Information Hub will be launched with <b>JMeter</b> and Average response time will be measured. The frequency and number of simultaneous requests will be gradually increased in order to determine IH performance under low/medium/high workload.
Throughput	Requests to the PIXEL Information Hub will be launched with <b>JMeter</b> . The frequency, size and number of simultaneous requests will be gradually increased in order to determine IH performance under low/medium/high workload.

KPI	Measurement approach
Mean CPU Utilisation	The same approach as for average latency is valid. In order to get the mean CPU usage, the <b>PerfMon JMeter plugin</b> <sup>26</sup> will be used for the same JMeter tests. <b>PerfMon</b> is a JMeter package for server monitoring. It can monitor CPU, Memory, Swap, Disks I/O and Networks I/O.
Mean memory usage	The same approach as for average latency is valid. In order to get the mean memory usage, the <b>PerfMon</b> library will be used for the same JMeter tests.
Maximum memory usage	The same approach as for average latency is valid. In order to get the maximum memory usage, the <b>PerfMon JMeter plugin</b> will be used for the same JMeter tests.
Maximum processing power used	The same approach as for average latency is valid. In order to get the maximum CPU usage, the <b>PerfMon JMeter plugin</b> will be used for the same JMeter tests.
<b>Reliability</b>	
Simultaneous requests	The same approach as for average latency is valid and <b>JMeter</b> will be used. Here JMeter probes will be defined to increase the number of concurrent requests progressively until the load arrives at a certain threshold.
% Monthly availability	For health status, an <b>availability probe</b> must be defined with minimal impact on performance. A periodic process will check regularly (e.g. every hour) if the IH (e.g. sending the test input and getting an expected successful response). Statistics will be collected and be available per month.
Success rate	For each access to the PIXEL information hub, the success or failure will be recorded, serving as statistics indicator.
<b>Maintainability</b>	
% of modularity	Will be measured by reporting all the independent components that are part of the information hub module and comparing them to the number of all components in the data acquisition module. Individual operation means that a component can offer a complete function with meaningful information in the context of PIXEL.

<sup>26</sup> <https://jmeter-plugins.org/wiki/PerfMon/>

KPI	Measurement approach
% of reusable assets	Will be measured by reporting all the reusable components that are part of the information hub module and comparing them to the number of all components in the data acquisition module. A reusable component is considered any that can be applied in a different context of PIXEL with no modifications of the source code.
% of update	Will be measured by reporting the level of success in software updates on the information hub module. It compares successfully completed updates versus all executed updates.
Level of analysability	Will be measured by reporting the ratio between the numbers of items inside the information hub for which logging is implemented compared to the number of items for which the specifications require logging.
<b>Portability</b>	
Mean number of errors per hardware or OS change/ upgrade	Will be measured by analysing the system & application error logs.
Mean number of errors per software change/ update	Will be measured by analysing the system & application error logs.
Mean number of errors per software install	Will be measured by analysing the system & application error logs.
Mean number of errors per software uninstall	Will be measured by analysing the system & application error logs.

## 6.3. PIXEL Operational Tools

In Table 16 below, KPI's assigned to T6.4 (Operational Tools) are described together with the methodology to be followed. Some of the KPIs will be measured under 'forced' conditions (e.g. average latency), as otherwise, we would need a long timeframe (months) to get useful and practical values. Other KPIs (e.g. CPU and memory usage) can be monitored in real-time by means of in-built services.

*Table 16. KPIs measurement approach: PIXEL Operational Tools*

KPI	T6.4 (Operational Tools) measurement approach
<b>Functional suitability</b>	
Straightforward task accomplishment	A process to add/define a new model or predictive algorithm will be analysed to verify that the process does not include unnecessary steps. Boolean response (Yes/No)
The portion of completed requirements	Deliverable <b>D3.2</b> will be taken as input in order to extract all <b>requirements</b> specifically targeting T6.4.



KPI	T6.4 (Operational Tools) measurement approach
Average latency	<p>Model and PA dependent. This extends T4.X. With <b>JMeter</b>, requests to multiple services (encapsulating models and PAs) will be launched and average response time will be measured. The number of simultaneous services will be established depending on the configuration of the services (models, PAs): refresh rate for scheduled services, estimated manual execution, estimated event-triggered executions.</p> <p>Furthermore, this number might change and provide different JMX probes:</p> <ul style="list-style-type: none"> <li>- Low: least estimation of models and PAs</li> <li>- Medium: average estimation of models and PAs</li> <li>- High: worst-case estimation of models and PAs</li> </ul>
Mean CPU Utilisation	Model and PA dependent. This extends T4.X. The same approach as for average latency is valid. In order to get the mean CPU usage, <b>PerfMon</b> will be used for the same JMeter tests.
Mean memory usage	Model and PA dependent. This extends T4.X. The same approach as for average latency is valid. In order to get the mean memory usage, <b>PerfMon</b> will be used for the same JMeter tests.
Maximum memory usage	Model and PA dependent. This extends T4.X. The same approach as for average latency is valid. In order to get the maximum memory usage, <b>PerfMon</b> will be used for the same JMeter tests.
Maximum processing power used	Model and PA dependent. This extends T4. X. The same approach as for average latency is valid. In order to get the maximum CPU usage, <b>PerfMon</b> will be used for the same JMeter tests.
<b>Reliability</b>	
Simultaneous requests	Model and PA dependent. This extends T4. X. The same approach as for average latency is valid and <b>JMeter</b> will be used. Here JMX probes will be defined to increase the number of concurrent requests progressively until the load arrives at a certain threshold.
% Monthly availability	<p>Model and PA dependent. For health status, an <b>availability probe</b> must be defined per each model and PA with minimal impact on performance. A test input might be provided by model/PA.</p> <p>A periodic process will check regularly (e.g. every hour) if a model/PA is available (e.g. sending the test input and getting an expected successful response). Statistics will be collected and be available per month.</p> <p>If there is unavailability from a service (model), it will try to recover automatically, otherwise, a notification (to the administrator) will be sent</p>

KPI	T6.4 (Operational Tools) measurement approach
Success rate	For each execution of the service (model), the <b>success or failure will be stored</b> , serving as <b>statistics indicator</b> .
<b>Maintainability</b>	
% of modularity	Will be measured by reporting all the independent components that are part of the operational tools module and comparing them to the number of all components in the data acquisition module. Individual operation means that a component can offer a complete function with meaningful information in the context of PIXEL.
% of reusable assets	Will be measured by reporting all the reusable components that are part of the operational tools module and comparing them to the number of all components in the data acquisition module. A reusable component is considered any that can be applied in a different context of PIXEL with no modifications of the source code.
% of update	Will be measured by reporting the level of success in software updates on the operational tools module. It compares successfully completed updates versus all executed updates.
Level of analysability	Will be measured by reporting the ratio between the numbers of items inside the operational tools for which logging is implemented compared to the number of items for which the specifications require logging.
<b>Portability</b>	
Mean number of errors per hardware or OS change/ upgrade	Will be measured by analysing the system & application error logs
Mean number of errors per software change/ update	Will be measured by analysing the system & application error logs.
Mean number of errors per software install	Will be measured by analysing the system & application error logs.
Mean number of errors per software uninstall	Will be measured by analysing the system & application error logs.

## 6.4. PIXEL Integrated Dashboard and Notification

In Table 17 below, KPI's assigned to T6.5 (Dashboard and notifications) are described together with the methodology to be followed.

*Table 17. KPIs measurement approach: PIXEL Integrated Dashboard and Notification*

KPI	Measurement approach
<b>Functional suitability</b>	
Straightforward task accomplishment	Unnecessary steps will be identified (if any) by technical partners in the consortium. Boolean response (Yes/No). A sample of tasks will be identified (5 tasks) and the number of clicks to accomplish them will be measured. This will be tested in different user profiles to check the KPI in different roles.
The portion of completed requirements	“Should have” and “Must have” requirements from deliverable D3.2 will be taken as input in order to extract all requirements specifically targeting T6.5.
Average latency	Requests to the dashboard environment will be launched with <b>JMeter</b> and Average response time will be measured. The requests will measure the time of loading a visualization or a dashboard via the REST (Representational state transfer) API. The frequency and number of parallel jobs will be gradually increased in order to determine dashboard environment performance under low/medium/high workload.
Mean CPU Utilisation	It will be measured by using browser-monitoring tools and performing a script of typical use.
Mean memory usage	It will be measured by using browser-monitoring tools and performing a script of typical use.
Maximum memory usage	It will be measured by using browser performance monitoring tools and performing a script of demanding use.
Maximum processing power used	It will be measured by using browser performance monitoring tools and performing a script of demanding use.
<b>Operability</b>	
Dashboard availability	Is there an available dashboard with easy navigation? Ease of use will be measured by quantitative aspects (see KPI ‘Straightforward task accomplishment’) [Yes/No/Partially]
Notifications system availability	Is there an available notifications system? [Yes/No/Partially]
GUI (Graphic User Interface) module availability	This KPI will check whether there is a GUI for: (i) dashboards navigation; (ii) notifications management. The results of this measure are [Yes/No/Partially].

KPI	Measurement approach
WCAG <sup>27</sup> 2.0 Conformance Level	The guidelines are described in <a href="https://www.w3.org/TR/WCAG20/">https://www.w3.org/TR/WCAG20/</a> and various <i>features</i> will be evaluated in terms of perceivable, operable, understandable and robust. This is not a single value [None/A/AA/AAA], but a list of pairs <feature-value> As there was not a strong requirement about content accessibility for the UI, and thus it is not anticipated to fulfil all <i>features</i> .
Maximum Concurrent users	It will be measured by using browser performance monitoring tools and performing a script of demanding use.
<b>Reliability</b>	
Simultaneous requests	The same approach as for average latency is valid and <b>JMeter</b> will be used. Here JMeter probes will be defined to increase the number of concurrent requests progressively until the load arrives at a certain threshold.
% Monthly availability	For health status, an availability probe must be defined with minimal impact on performance. A periodic process will check regularly (e.g. every hour) if the dashboard tool and the notification service are available (e.g. sending a request to the dashboard engines, checking the status of the web application and the notifications service)). Statistics will be collected and be available per month.
Success rate	For each execution of the dashboard and notification services, the success or failure will be stored, serving as statistics indicator.
<b>Maintainability</b>	
% of modularity	This will be measured by reporting all the independent components that are part of the dashboards and notifications module. The individual operation means that the component can offer a complete function with meaningful information in the context of PIXEL.
% of reusable assets	This will be measured by reporting all the reusable components that are part of the dashboards and notifications module. A reusable component is considered any that can be applied in a different context of PIXEL with no modifications of the source code.
% of update	This KPI will measure the level of success in software updates. It will be measured by reporting the different updates performed during pilots.
Level of analysability	Measured the quality and precision of measures that can report about the performance when components change.
<b>Portability</b>	

<sup>27</sup> Web Content Accessibility Guidelines

KPI	Measurement approach
Mean number of errors per hardware or OS change/ upgrade	They will be measured by analysing the system error logs and application error logs (e.g. using the command dmesg).
Mean number of errors per software change/ update	They will be measured by analysing the system error logs and application error logs (e.g. using the command dmesg).
Mean number of errors per software install	They will be measured by analysing the system error logs and application error logs (e.g. using the command dmesg).
Mean number of errors per software uninstall	They will be measured by analysing the system error logs and application error logs (e.g. using the command dmesg).

## 6.5. PIXEL Security and Privacy

In Table 18 below, KPI's assigned to T6.6 (Security) are described together with the methodology to be followed. Most of the software components of covering security and privacy come from the FIWARE catalogue. Those components have complete [quality tests coverage](#).

*Table 18. KPIs measurement approach: PIXEL Security and Privacy*

KPI	Measurement approach
<b>Functional suitability</b>	
Straightforward task accomplishment	Processes for authentication and authorisation will be analysed to verify that they do not include unnecessary steps. Boolean response (Yes/No)
The portion of completed requirements	“Should have” and “Must have” requirements from deliverable D3.2 will be taken as input in order to extract all requirements specifically targeting T6.6.
Mean CPU Utilisation	The same approach as for average latency is valid. In order to get the mean CPU usage, the <b>PerfMon JMeter plugin</b> <sup>28</sup> will be used for the same JMeter tests. <b>PerfMon</b> is a JMeter package for server monitoring. It can monitor CPU, Memory, Swap, Disks I/O and Networks I/O.
Mean memory usage	The same approach as for average latency is valid. In order to get the mean memory usage, PerfMon will be used for the same JMeter tests.
Maximum memory usage	The same approach as for average latency is valid. In order to get the maximum memory usage, the <b>PerfMon JMeter plugin</b> will be used for the same JMeter tests.

<sup>28</sup> <https://jmeter-plugins.org/wiki/PerfMon/>

KPI	Measurement approach
Maximum processing power used	The same approach as for average latency is valid. In order to get the maximum CPU usage, the <b>PerfMon JMeter plugin</b> will be used for the same JMeter tests.
<b>Reliability</b>	
Simultaneous requests	The same approach as for average latency is valid and <b>JMeter</b> will be used. Here JMeter probes will be defined to increase the number of concurrent requests progressively until the load arrives at a certain threshold.
% Monthly availability	For health status, an <b>availability probe</b> must be defined with minimal impact on performance. A periodic process will check regularly (e.g. every hour) if security components are available. Statistics will be collected and be available per month.
Success rate	For each access to the PIXEL information hub, the success or failure will be recorded, serving as statistics indicator.
<b>Security</b>	
Incidents of ownership changes and accessing prohibited data	Will be measured by reviewing the API Gateway logs.
Incidents of authentication mechanisms breaches	Will be measured by reviewing the Keyrock logs.
Level of User authenticity	The authentication system will be analysed/assessed to assure that it provides genuine user authenticity.
<b>Maintainability</b>	
% of modularity	Will be measured by reporting all the independent components that are part of the security and privacy module and comparing them to the number of all components in the data acquisition module. Individual operation means that a component can offer a complete function with meaningful information in the context of PIXEL.
% of reusable assets	Will be measured by reporting all the reusable components that are part of the security and privacy module and comparing them to the number of all components in the data acquisition module. A reusable component is considered any that can be applied in a different context of PIXEL with no modifications of the source code.
% of update	Will be measured by reporting the level of success in software updates on the security and privacy module. It compares successfully completed updates versus all executed updates.

KPI	Measurement approach
<b>Portability</b>	
Mean number of errors per hardware or OS change/ upgrade	Will be measured by analysing the system & application error logs.
Mean number of errors per software change/ update	Will be measured by analysing the system & application error logs.
Mean number of errors per software install	Will be measured by analysing the system & application error logs.
Mean number of errors per software uninstall	Will be measured by analysing the system & application error logs.

## 7. Software release and documentation

The different software components in PIXEL are being developed under two main objectives:

- *Open source approach*: most of the developed modules will be released as open source, while the source code is available in an internal Git repository, so that any partner within the Consortium can get an easy access and can potentially contribute (as a tester or developer). In the future, once the final releases of each software module are available and have been tested in the pilots, the aim is to port the source code to a public repository (e.g. GitHub<sup>29</sup>).
- *Containerization*: Though services may be implemented in different languages, one important goal is to encapsulate the different modules into (Docker) containers, so that they facilitate the deployment within the PIXEL platform. In fact, this is a best practice for continuous integration, delivery and deployment. Similar to the previous step, the aim is to port the implemented containers into a public repository (e.g. Docker Hub<sup>30</sup>).

In the chapters below, a list of developed software modules is presented. Access to the different files will be granted through a dedicated link with associated credentials.

Regarding documentation, as for the Grant Agreement, D6.3 and D6.4 are “**the main asset of software documentation of the project**”. This implies documenting all software pieces that have been implemented within WP6, mainly concerning the architecture modules, about general description, installation, configuration and execution. Two approaches have been considered:

- Include the documentation within the Git repository. Therefore, the documentation is attached with the source code, typically with a README.md file, and possibly with a doc directory with related information, if any. This approach will be considered for this D6.3.
- Export the documentation to a public and user-friendly interface. In other open source projects, the use of *Readthedocs*<sup>31</sup> has been used, which offers a web user interface, more accessible to non-developers. PIXEL plans to release the documentation into this platform for D6.4, once all software components have been integrated.

Items from Table 19 to Table 23 provide an overview of software releases for each of the PIXEL components. Each table provides a list of modules with their respective: file name, short description of the module, size, filename for the documentation and the programming language in which the module has been developed.

The current release of PIXEL Data Acquisition is decomposed in four main blocks, as presented in the table below.

*Table 19. Software release overview: PIXEL Data Acquisition*

File name	Description	Documentation	Language
data-acquisition-core.zip	PIXEL DAL Installation files and process	README.md	Binary Installation
nifi-ngsiagents-proto.zip	NiFi NGSI Agents prototype for Thessaloniki WindData and VesselCall	README.md	NiFi XML Flow
gpmb-vcallfwagent.zip	GSI Agents to handle VesselCall request from GPMB	README.md	Python

<sup>29</sup> <https://github.com/>

<sup>30</sup> <https://hub.docker.com/>

<sup>31</sup> <https://readthedocs.org/>



File name	Description	Documentation	Language
nmeafwagent.zip	NGSI test agent to handle AIS information from Open Source NMEA (National Marine Electronics Association)	README.md	Python

The current release of PIXEL Information Hub is decomposed in three main blocks, as presented in the table below.

*Table 20. Software release overview: PIXEL Information Hub*

File name	Description	Documentation	Language
information-hub-docker.zip	Docker deployment for PIXEL Information Hub.	README.md	Docker scripts
pixel-data-collector.zip	Data collector component for PIXEL Information Hub.	README.MD	Java

The current release of PIXEL Operational Tools is decomposed in three main blocks, as presented in the table below.

*Table 21. Software release overview: PIXEL Operational Tools*

File name	Description	Documentation	Language
otpixel-master.zip	Publication service (for models and PAs)	otpixel.MD; Swagger for API documentation	Java
ot-energy-basicModel-master.zip	Basic model for energy consumption forecasts	ot-energy-basicModel.MD	Javascript (NodeJS)
ot-planner-master.zip	Engine for executing scheduled services (models and PAs)	ot-planner.MD	Java
ot-ui-basicModel-master.zip	UI interface for the basic model (vuetified)	ot-ui-basicModel.MD	Javascript (Vuetify)

The current release of PIXEL Integrated Dashboard and Notification is decomposed in five main blocks, as presented in the table below.

*Table 22. Software release overview: PIXEL Integrated Dashboard and Notification*

File name	Description	Documentation	Language
nginx-proxy.zip	Repository for Nginx configuration	README.md	Configuration files
docker-elastic.zip	Docker-compose for elastic machine	README.md	Docker scripts
docker-dashboards.zip	Docker-compose for dashboard machine	README.md	Docker scripts

File name	Description	Documentation	Language
openvpn-proxy.zip	OpenVPN configuration files for development	README.md	Configuration files
pixel-platform.zip	PIXEL platform main project	README.md	Java

The current release of PIXEL Security and Privacy consists of one module, as presented in the table below.

*Table 23. Software release overview: PIXEL Security and Privacy*

File name	Description	Documentation	Language
security-core.zip	PIXEL Security Keyrock and Wilma Installation files and process	README.md	Binary Installation