PHOENIX

WP2 - Requirements, Use case definition and Architecture Blueprint

Document Version:

D2.3 Technical requirements and human centric architecture specifications

1.0

Project Number:	**Project Acronym:**	**Project Title:**
893079	PHOENIX	Adapt-&-**P**lay **H**olistic c**O**st Effective and user-frie**N**dly **I**nnovations with high replicability to upgrade smartness of e**X**isting buildings with legacy equipment

Contractual Delivery Date:	**Actual Delivery Date:**	**Deliverable Type* - Security**:**
30/09/2021	30/09/2021	R - PU

* Type: P - Prototype, R - Report, D - Demonstrator, O - Other

** Security Class: PU- Public, PP - Restricted to other programme participants (including the Commission), RE - Restricted to a group defined by the consortium (including the Commission), CO - Confidential, only for members of the consortium (including the Commission)

Responsible and Editor/Author:	Organization:	Contributing WP:
Tsatsakis Konstantinos	Suite5	WP2

Authors (organizations):

Tsatsakis Konstantinos, Gkilpathi Evgenia (Suite5), Eirini Christoulaki (Kama), Antonio Skarmeta, Alfonso Ramallo (UMU), Rafael Marín Pérez, Alfredo Quesada Sánchez (ODINS), Aristotelis Ntafalias, Kyriakos Kentzoglanakis, Panagiotis Papadopoulos (VERD) , Josiane Xavier Parreira, Stefan Bischof (SAGOE), Dimitra Georgakaki (Ubitech), Fergal Purcell, Eoin O'Connor (ARDEN)

Abstract:

The scope of the document as defined in the DoA is to specify the PHOENIX Reference Architecture as well as the technical specifications. Based on use cases and business requirements definition, we define: (a) the Conceptual Architecture as an overview of the system architecture unfolding the modules of the PHOENIX and presenting the different sub-elements (b) PHOENIX Modules Functional and Technical Specifications to provide the detailed functional and technical view of Individual Components, the interfaces among them as well as interconnections with external interfaces, (c) The PHOENIX Platform development and deployment specifications analysis. The analysis performed in this document will pave the way for the development of these services in the following parts of the work in the project.

Keywords: Software architecture, ICT services, PHOENIX modules, PHOENIX development, PHOENIX deployment

Revision History

The following table describes the main changes done in the document since created.

Revision	Date	Description	Organization
0.1	02/02/2021	Intro to the task activities and linkage with other tasks of the project	Suite5, UMU, ODINS, MIWENERGIA, KaMa, VERD, UBITECH, ARDEN, SAGOE, LTU
0.2	08/03/2021	Intro and ToC	Suite5
0.3	12/04/2021	Feedback on the ToC and the allocation of the work	UMU, ODINS, MIWENERGIA, KaMa, VERD, UBITECH, ARDEN, SAGOE, LTU
0.4	24/04/2021	First version of the document	Suite5
0.5	12/05/2021	Feedback on the 1st version of the document	UMU, ODINS, UBITECH, SAGOE
0.6	30/06/2021	Second version of the document	Suite5
0.7	10/07/2021	Feedback on the 2nd version of the document (focus on deployment and demonstration actions)	UMU, ODINS, MIWENERGIA, KaMa, VERD, UBITECH, ARDEN, SAGOE, LTU
0.8	31/08/2021	Final Version for review	Suite5
0.9	10/09/2021	Internal revision	MERITCH, SKEBIT
1.0	30/09/2021	Final Version for submission	UMU

Executive Summary

The scope of the document as stated in the DoA is to specify the PHOENIX Reference Architecture as well as the technical specifications. The aim of the respective task (T2.3) is to deliver a complete understanding on the PHOENIX Reference Architecture, its building blocks, elements, interdependencies between components and related limitations to the development procedure.

Following the definition of the PHOENIX use cases and requirements at the very early phase of the work, the overview of the architecture of PHOENIX project is specified, describing the key modules that consist of the overall PHOENIX platform along with their functionalities. More specifically, the software view of the PHOENIX solution is reported, namely:

- Conceptual Architecture Documentation: an overview of the system architecture unfolding the modules of the PHOENIX and presenting the different sub-elements, the interfaces among them as well as interconnections with external interfaces
- PHOENIX Modules Functional and Technical Specifications: the scope of this part of the architecture is twofold:
    - to provide the functional view of Individual Components, mentioning to the comprehensive description of the functionalities, non-functional characterisations as well as communicational specifications
    - to provide the list of technical requirements that needs to be addressed at the development by taking into account the functional requirements as defined at the early phase of the project
- PHOENIX Platform dynamic view, reporting the different sequence flows among the different components in order to address the use cases of the project
- The PHOENIX Platform development and deployment providing details about the development process for the different modules as well as information about the software deployment activities to be performed in the project.

The outcome of this work is the definition of the PHOENIX architecture and its comprehensive specifications. The analysis performed in this document will pave the way for the development of these services in the following activities (WP3, WP4, WP5 and WP6) of the project.

Disclaimer

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 893079, but this document only reflects the consortium's view. The European Commission is not responsible for any use that may be made of the information it contains.

List of Figures

List of Tables

1 Introduction

1.1 Purpose of the Document

The scope of the document is to specify the PHOENIX Reference Architecture as well as its technical specifications. The aim of the work as defined in the DoA [1] is to provide a complete understanding on the PHOENIX Reference Architecture, by defining its building blocks, elements, interdependencies between components and related limitations to the development procedure.

Considering use cases and business requirements definition in D2.1, the work in this document focuses on the characterisation of the architecture of PHOENIX by describing its key modules along with their functionalities. For that reason, the following steps of the work are considered:

- Conceptual Architecture Documentation: an overview of the system architecture unfolding the modules of the PHOENIX and presenting the different sub-elements, the interfaces among them as well as interconnections with external interfaces

- PHOENIX Modules Functional and Technical Specifications: the scope of this part of the architecture is twofold:

    - to provide the detailed functional view of Individual Components, mentioning to the comprehensive description of the functionalities, non-functional characterisations as well as communicational specifications

    - to define in detail, the list of technical requirements that needs to be addressed at the development of the different system components

- The PHOENIX Platform development and deployment specifications analysis will be extracted complementary to the definition of system modules.

The work will yield the PHOENIX architecture and comprehensive specifications for the different ICT services delivered in the project. The analysis performed in this section will pave the way for the development of these services in the following parts of the work.

1.2 Relevance with other tasks

It is evident that the definition of the PHOENIX architecture is tightly linked with the definition of business needs and objectives of the project. Therefore, the starting point for the work is the

definition of the high-level objectives of the project, as specified in the DoA, further considering the PHOENIX business details (use cases and business requirements) as reported in D2.1 [2]. In addition, the preliminary market analysis which is further reflected in the business case definition in D2.2 is also considered at the design of the final solution. Moreover, the review of the social requirements (social enablers and barriers as also specified in D2.2 [3]) of the PHOENIX project are also examined for the design of the final solution.

On the other hand, the definition of the PHOENIX architecture will pave the way for the development of the ICT services in WP3, WP4, WP5 and WP6. The description of the functionalities along with the documentation of the technical requirements will be the key point for the development activities to be performed in the project.

1.3 Structure of the Document

In the first section of the document, an introduction to the work is provided. Then the structure of the work follows:

- In Section 2, the methodological framework for the work is defined
- In Section 3, we present the conceptual architecture of the PHOENIX solution
- In Section 4-7, the different viewpoints of the PHOENIX solution are presented, namely the logical, process, development and deployment view.

In the last section, the summary results of the work are presented.

2 PHOENIX Architecture Methodology

Before starting the presentation of the overall PHOENIX Architecture, we specify the methodology to be considered for the design of the PHOENIX solution. A standards-based methodology has been selected for the definition of PHOENIX Architecture framework. More specifically, we have adopted the principles as specified in ISO/IEC/IEEE/42010 standard through the provision of the different viewpoints to describe and represent large and complex software-intensive systems.

As there are in the literature different modelling approaches that adopt the ISO standard principles, we are selecting for PHOENIX the '4+1' view model of architecture as one the most popular approaches. This modelling approach is based on 4 different views (Logical, Implementation, Process and Deployment) that are combined together to produce the extra view of the system (Scenarios) [4]. The generic representation of the '4+1' model architecture model is presented in the following figure (Figure 1).

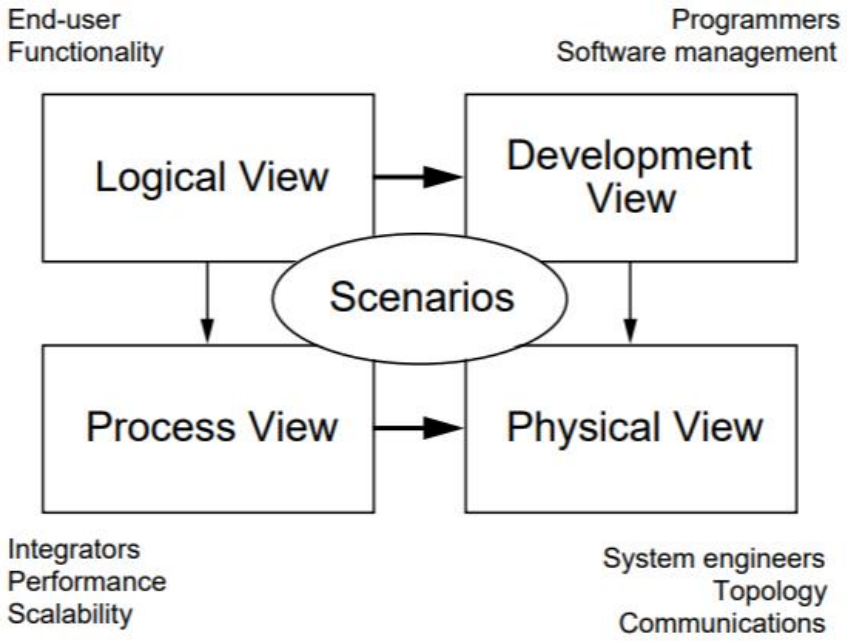

Figure 1 The '4+1' view model of architecture [4]

The different architectural views that consist of this architecture approach, to be considered also in PHOENIX project, are described below:

- The logical (structural) view is the static view that describes the functionalities provided and the system elements. Modules, abstractions, separation of concerns and responsibilities per element are the key points covered by this view. As part of this work, the technical requirements for each of the components will be defined on the basis of the business

requirements as derived in the early phase of the project. This view is presented in Section 3.

- The process view is responsible of presenting the dynamic flows of the architecture. It addresses the way the different modules communicate to each other, putting the focus on the sequence flows of the systems. This view of the system is presented in detail in Chapter 4.
- The implementation (development) view provides the viewpoint of a software engineer, as it defines the usage of existing development tools and programming languages (including potential legal and access policies), technical approaches and dependencies. This information is documented in Section 5.
- The deployment (physical) view is the part where software modules are mapped to hardware. It describes the real and physical deployment of the final solution. This part is covered in Section 6.

Finally, the use case (scenario) view is the plus one view of the provided methodology and has already been addressed in previous deliverables. The details about the system use cases have been provided in D2.1 while the specifications for the PHOENIX business case are provided in D2.2.

Before presenting the different viewpoints of the PHOENIX architecture, the description of the PHOENIX conceptual architecture as defined also in the DoA is provided in the next section.

3 PHOENIX Conceptual Architecture

As stated above and in the DoA, the scope of the project is to provide a solution that will guarantee i) that the overall implementation will be carried out to ensure full compatibility with legacy formats, interfaces and operating systems currently adopted in the building sector, and ii) that the project will consider both easy-to-upgrade and promising ICT technologies that has a high chance to be deployed in the future. This will guarantee the adaptability of PHOENIX ICT solutions to future scenarios, which will ensure the use of PHOENIX innovations beyond the end of the project. By taking into account these high-level principles for the PHOENIX solution, a first version of the conceptual architecture was defined in the DoA, further fine-tuned in this section to reflect the business needs and priorities as expressed in D2.1 and D2.2.

Overall, the conceptual architecture of the PHOENIX solution is divided into different layers, presented in the following figure (Figure 2).

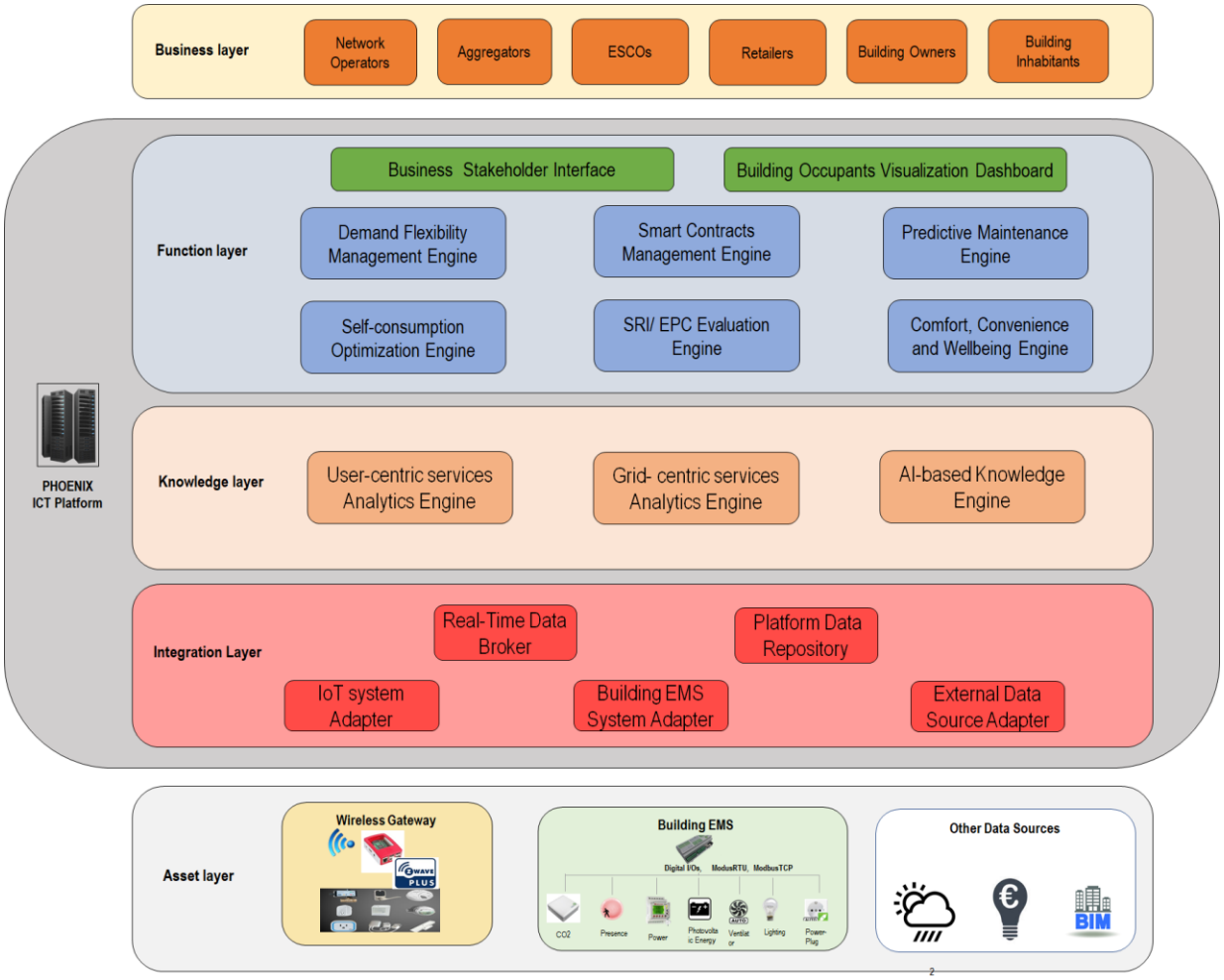

Figure 2 PHOENIX Conceptual Architecture

The different layers that consist of the overall solution are detailed:

- **Business layer** represents the point of views and the interactions with the end-users (e.g., building users/managers) and business stakeholders (e.g., ESCO, Aggregators, etc). This layer provides the technical and business aspects of the large-scale pilots along with the associated requirements, with a specific focus on the "reference" nature of the envisaged architecture. This layer will also enable to create also holistic links among different business models.
- **Function layer** includes multiple smart cost-effective services offered to the end users to optimise energy saving, occupants' satisfaction, overall performance of the buildings and grid related operations. More specifically, PHOENIX will implement decision support dashboards and grid communication interfaces to allow a smart building operation according to the needs of end-users and business actors, respectively. PHOENIX will provide energy related and non-energy related services for the building occupants along with communication interfaces and services for enabling advanced grid operations such as demand response, smart billing, load shifting, etc. On top of the different business services as briefly presented, PHOENIX will provide adaptable dashboards for collecting the user preferences and behaviour in the use of the building equipment and energy resource.
- **Knowledge layer** consist of modular tools that create knowledge through data processing and analytic techniques on the way to upgrade the smartness of the buildings. PHOENIX will explore knowledge base techniques and annotations to build a background Knowledge Graph (KG) that will facilitate the extraction of building related information (e.g. SRI related aspects) along with relevant information about preferences related to user behaviours. In addition, PHOENIX will implement artificial intelligence data algorithms (e.g., machine learning and deep learning) to enable self-learning capacities and automatic decisions to improve energy savings and the overall performance of buildings. Moreover, the data algorithms will generate valuable knowledge to feed the upper layer of smart services. In particular, this layer will provide the necessary knowledge and predictions to support the smart services for building users and grid interactions.
- **Integration layer** provides the mechanisms for the remote control and data monitoring of different building equipment, systems and external data sources (i.e., weather predictions) with heterogeneous protocols and technologies. This counts for (a) IoT Gateways and Smart Controllers to turn on/off and measure the consumption of non-digital devices as well as to translate legacy protocols (e.g. Modbus, Zigbee, etc) of digital equipment to

standardised Internet protocol in order to monitor and control their operations, (b) building management systems integrators to handle the same functionality in commercial buildings where BMS systems are available (c) external data source wrappers to incorporate energy e.g. energy tariffs) and weather related data from national or international organizations. This layer will be also responsible for the homogenisation of the different communication protocols and also storage to different data formats to enable the easy link between the physical assets and the layer of knowledge creation as presented above.

- **Asset layer** consists of heterogeneous legacy equipment and systems already deployed in the buildings that must be integrated and managed intelligently. Namely:
    - Short-life appliances (e.g., fridges, ovens, washing machines, microwaves, etc).
    - Long-life appliances (e.g., HVAC, boilers, radiators, DHW (domestic hot water) devices, ventilation, lighting, etc.)
    - Energy building equipment (e.g., renewable sources, energy storage, e-vehicle recharging points, energy demanding points, smart meters, smart actuators, etc).
    - Management systems (e.g., EMSs) already deployed in existing buildings to monitor and control legacy equipment.

  Devices that exist in building can be categorised in four families: Envelope Adaptation systems (e.g., windows automatic opening and shading), Technical Building Systems (HVAC, DHW, built-in lighting, building automation and control, on-site electricity generation), Indoor and Outdoor Information Systems (i.e., sensors) and Entertainment Systems (e.g., television, radio, etc.). Also, equipment can be grouped in two categories according to their communication capacities. First, digital equipment (e.g., HVAC, smart appliances) has wired or wireless communication for monitoring and control based on technologies and protocols such as KNX, Modbus, BACnet, Zigbee, Zwave, etc. Non-digital devices (e.g., typical fridges, ovens, microwaves) have not the capacity for data communications thus these devices only can be turned on/off and measured their consumptions.

It is evident from the previous analysis, that there are three main technological layers (Integration, Knowledge, Function) that define the ICT software details of the PHOENIX solution with the rest layers (Asset & Business Layer) to complement the overall picture for the demonstration of the PHOENIX solution.

We have to point out that there is an extra layer, the **Protection Layer,** which provides the techniques and protocols to ensure the security, privacy and trust of the data exchange in all layers of the PHOENIX solution. Due to the interoperable and connected nature of PHOENIX architecture, security, privacy and trust mechanisms are required to protect the data exchange among physical assets, integration agents, knowledge processing techniques, smart services and user interfaces in all abovementioned layers. These security/privacy mechanisms, to be developed based on FIWARE Security Enablers to allow devices/systems authentication, privacy preserving & data protection, services access control and user management will be described in details in a dedicated task of the PHOENIX project (Task 4.1 Security and Privacy feature).

By presenting in brief, the different layers that consist of the PHOENIX overall solution, the details of the different components that consist of the PHOENIX platform are provided in the following section.

4 PHOENIX Logical View

By taking into account the high-level objectives of the PHOENIX solution and the definition of conceptual architecture in previous section, the details description of the different modules that consist of the PHOENIX framework is provided in this section. More specifically, the scope of this section as stated also in the introductory methodology is twofold:

- To clearly define the features to be supported by the different modules that consist of the final solution along with interface details.
- To document the list of the technical requirements that need to be fulfilled in order to meet the business objectives as defined in the project. The analysis takes into account the list of end user, business and social requirements as extracted in D2.1 and D2.2 in order to extract the technical requirements for the different software components.

The following table (Table 1) presents the list of software components that consist of the PHOENIX solution (as depicted also in Figure 2).

ID	Software Component	PHOENIX Architecture Layer
1	PHOENIX External Data Source Adapter	Integration Layer
2	PHOENIX Building System Adapter	
3	PHOENIX IoT System Adapter	
4	PHOENIX Real-Time Data Broker	
5	PHOENIX Platform Data Repository	
6	AI-based Knowledge Engine	Knowledge Layer
7	User - centric Services Analytics Engine	
8	Grid - centric Services Analytics Engine	
9	Comfort, Convenience and Wellbeing Engine	Function Layer
10	Predictive Maintenance Engine	
11	SRI/ EPC Evaluation Engine	
12	Building Occupants Visualization Dashboard	
13	Smart Contracts Management Engine	
14	Demand Flexibility Management Engine	
15	Self-consumption Optimization Engine	

16	Business Stakeholder Interface Engine	

Table 1 PHOENIX Software Components

The functional details of these components are presented in the following sections.

4.1 PHOENIX External Data Source Adapter

Overview

Buildings are complex entities which are highly dependent on external factors such as the weather, the social events or the energy prices. It is for this reason that an integrated solution such as PHOENIX will need to gather data not only from within the building envelope, but also from third party sources that will provide context to the events occurring within the building. The External data source adapter is a conceptualisation of a module that is in charge of making possible the connection of the PHOENIX platform with the third parties. Although the actual connection is done with a series of software parts, it is possible to consider them as a single module as they all have the same functional parts. As there has been considerable progress on this implementation, in Annex I we present the developments on external data sources integration on PHOENIX so a real example can be seen to illustrate the section.

List of Features

The detailed list of features supported by the component is presented in the following:

- **3rd party data source configuration:** the scope of this feature is to ensure connectivity with a 3rd party data source in order to be able to retrieve the data required.
- **Continuous update of data variables:** the flow of information between the third party and the broker will be done with the adapter. This will be processed by the agent and converted into an update in the broker.

Component Functional Elements

By presenting the key functionalities, of the External Data Source Adapter, details about the different functional modules are presented in the following schema (Figure 3).

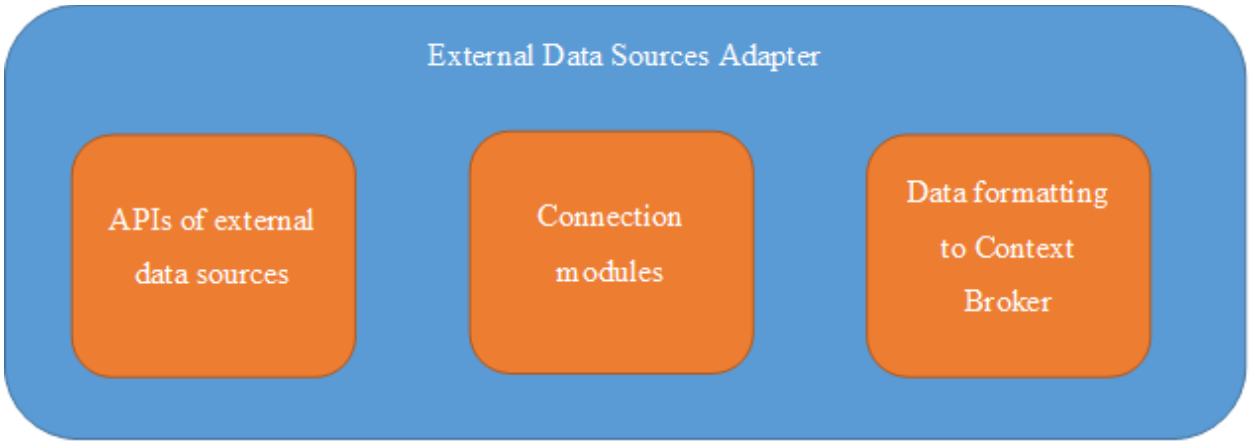

Figure 3 External data sources adapter Overview

More specifically:

- **APIs of external data sources:** The adapter for the external data sources will have a hardcoded interconnection with the APIs of the external data sources, so relevant information of the sources is extracted with a periodicity that is optimal for the PHOENIX platform.
- **Connection modules:** This component will consist of node RED and python codes (or any other low-level programming language) that are hosting on secure servers to retrieve the information from the APIs of the third parties.
- **Data formatting for context broker:** Within this module there will exist the definition of the entities on NGSI-LD format, and the necessary stages to transform the data coming in a variety of formats from the third parties' APIs to the correct NGSI-LD format JSON that respond to the entities' definition.

The external data sources adapter has the following list of technical requirements:

Req. ID	Description
T_EDS_01	The component should have access to the APIs of third-party data providers
T_EDS_02	The component should have access to the NGSI-LD broker
T_EDS_03	The component should allow secure communications between itself and the broker

T_EDS_04	The component will perform all communications with the broker, following the NGSI-LD standard, both for context management and subscriptions
T_EDS_05	The component will ensure real time information exchange among the Third parties and the NGSI-LD broker
T_EDS_06	The component will perform all necessary data transformations required in order to ensure semantic interoperability at the NGSI-LD broker level
T_EDS_07	The component should be flexible enough to be resilient to changes on relevant APIs
T_EDS_08	The component needs to be capable of dealing with the data flow of the APIs and to take into consideration potential lags on requests
T_EDS_09	Outdoor environmental conditions should be available by this component
T_EDS_10	The component needs to be capable of dealing with weather forecast data should be available for further analysis
T_EDS_11	The component needs to be capable of dealing with Energy Price data should be available for further analysis

Table 2 External Data Source Adapter - Technical Requirements

Dependencies/ Inputs/Outputs

The PHOENIX External Data Source Adapter interfaces with the:

- PHOENIX Real-Time Data Broker:
    - Update timely data associated with the contextual conditions related to buildings operation as retrieved from 3rd party entities.

4.2 PHOENIX Building System Adapter

Overview

Several buildings, particularly those of large size, have an integrated Building Management System, or BMS. These building management systems are solutions that collect information from the building, and in some cases, allow actuation over certain devices either planned or manual. Buildings with a system as such have an ahead start on the pathway to becoming a smart building, but for that to be fully used requires an integration in an overruling solution as the one that has been designed on PHOENIX. The integration of BMSs has been taken into consideration for PHOENIX, at the level that a BMS adapter can be seen as a functional part of the solution.

List of Features

The detailed list of features supported by the component is presented in the following:

- **Bidirectional communication with components on the BMS:** the adapter has to be capable of performing both updates from a device on the BMS to the NGSI-LD broker (PHOENIX Real-Time Data Broker), and command execution via the BMS to a given device, triggered from an update in the broker.

- **Updating of components from the BMS to the Context Broker:** The BMS is an active component of the building management. It is for this reason that it will be necessary to have a way of catching up with new additions to the BMS of the building so they can be integrated on the adapter and ultimately on the context broker.

Component Functional Elements

By presenting the key functionalities of the Building System Adapter, details about the different functional modules are presented in the following schema (Figure 4). We have to point out that the Building System Adapter architecture is similar to the External Data Source Adapter architecture as presented above:

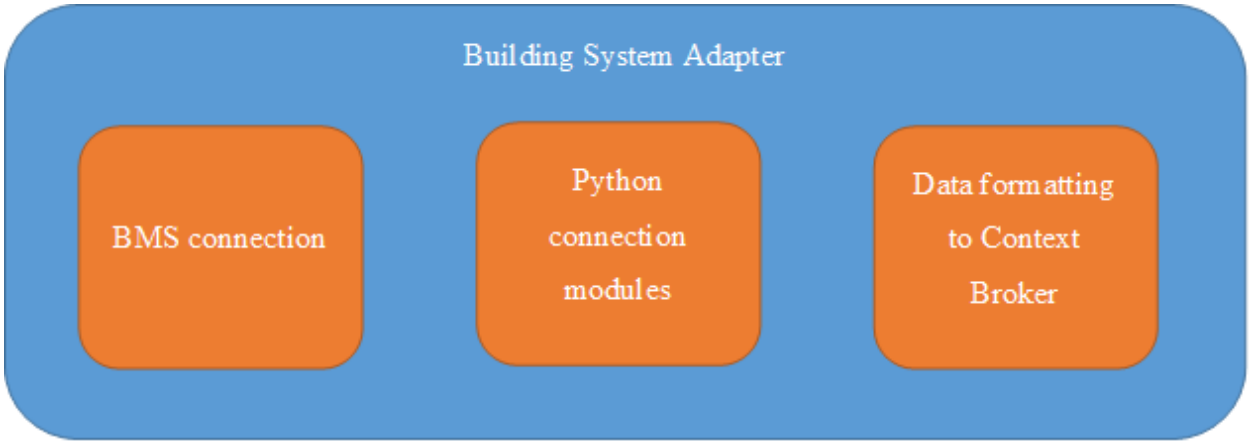

Figure 4 BMS Adapter Overview

More specifically, the different modules that consist of the Building System Adapter are presented:

- **BMS Connection:** Different BMSs have different ways of connecting to them to allow bi-directional communication. The BMS adapter will have the necessary language to be able to connect to the BMS. Therefore, a configuration process is required to ensure connectivity with each BMS under examination.
- **Connection modules:** This component consist of node RED and/or python codes (although Figure 4 shows only python) that are hosting on secure servers to retrieve the information from the BMS on a live manner. It complements the functionality of the BMS connector as presented above.

- **Data formatting to context broker:** Within this module there will exist the definition of the entities on NGSI-LD format, and the necessary stages to transform the data coming in a variety of formats from the BMS to the correct NGSI-LD format JSON that respond to the entities definition. This is the wrapper to ensure connectivity with the Real-Time Data Broker of the project.

The BMS Adapter has the following list of technical requirements:

Req. ID	Description
T_BMS_01	The component should have access to BMS via IP based connection
T_BMS_02	The component should have access to the NGSI-LD broker
T_BMS_03	The component should allow secure communications between itself and the broker
T_BMS_04	The component will perform all communications with the broker, following the NGSI-LD standard, both for context management and subscriptions
T_BMS_05	The component will ensure information exchange among the BMS and the NGSI-LD broker
T_BMS_06	The component will ensure real time information exchange of the different types of measurements examined in the project (energy metering/demand/generation, storage, device operational status, indoor environmental and health conditions, occupancy related information)
T_BMS_07	The component will perform all necessary data transformations required in order to ensure semantic interoperability at the NGSI-LD broker level
T_BMS_08	The component should be flexible enough to be resilient to changes on relevant BMS
T_BMS_09	The component needs to be capable of dealing with the data flow of the BMS and to take into consideration potential lags on requests
T_BMS_10	The component will need to produce actuation commands on the correct format to be able to actuate over devices on the BMS

Table 3 PHOENIX BMS Adapter - Technical Requirements

Dependencies/ Inputs/Outputs

The PHOENIX BMS Adapter interfaces with the:

- PHOENIX Real-Time Data Broker:
    - Update time data associated with the contextual conditions in building environment and the operational status of the different controllable devices.

- Trigger control actions associated with the decisions as defined by the automation module of the component via the BMS.

4.3 PHOENIX IoT System Adapter

Overview

The role of this component is to bridge the gap between smart home devices (Asset Layer) and the core data brokering (PHOENIX Real-Time Data Broker) of the PHOENIX architecture. In order to do that, this component essentially translates messages received via MQTT from devices, and translates them to NGSI-LD updates on the entities associated to those devices. It also allows the way back, in order to be able to perform some kind of actuation from the NGSI-LD realm to the MQTT device.

List of Features

The detailed list of features supported by the component is presented in the following:

- **Bidirectional communication with smart home devices/building GWs:** the software agent is capable of performing both updates from a smart IoT device to the NGSI-LD broker (PHOENIX Real-Time Data Broker), and command execution in the device, triggered from an update in the broker.
- **Continuous update and passive (lazy) attributes:** the usual flow of information between device and broker will be triggered by an update from the device, in the form of a MQTT notification. This will be processed by the agent and converted into an update in the broker. In order to alleviate the pressure on data update, some attributes can be defined as lazy, meaning that information will not be constantly pushed to the broker, but queried to the device upon request. This way, when the broker receives a query for some lazy attribute, this in turn will query the agent and it will request the information from the device.
- **Device provisioning via REST API:** devices can be registered in the IoT agent via a REST API, which allows individual as well as group registrations.

Component Functional Elements

By presenting the key functionalities, of the IoT system adapter, details about the different functional modules that set the IoT system adapter are presented in the following schema (Figure 5).

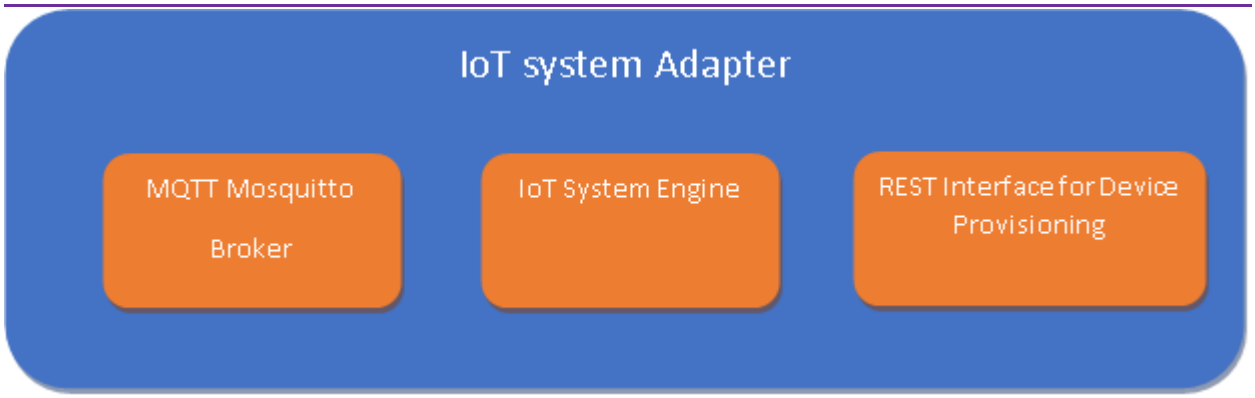

Figure 5 IoT system adapter Overview

More specifically:

- **MQTT Mosquitto Broker:** The role of this module is to act as the wrapper of the IoT adapter to guarantee data exchange with external components (i.e. IoT devices, external data sources) via MQTT protocol.
- **IoT System Engine:** The role of the IoT system engine is to translate the MQTT format of the IoT devices to the REST API of the NGSI-LD broker in order to guarantee bidirectional communications.
- **REST Interface for Device Provisioning:** This internal REST interface to enable the device registration as part of the auto configuration of the different physical devices in building premises.

The IoT system component has the following list of technical requirements:

Req. ID	Description
T_ISA.01	The component should have access to the MQTT broker provided by a smart device/GW component
T_ISA.02	The component should have read access to the MQTT topics to which the devices publish
T_ISA.03	The component should have “write access” to the MQTT topics to which the devices read configuration and commands
T_ISA.04	The component should have access to the NGSI-LD broker
T_ISA.05	The component should be accessible via REST for the use of the provisioning API
T_ISA.06	The component should be accessible via REST for the notifications API of the NGSI-LD broker

T_ISA.07	The component should allow secure communications between the agent and the broker
T_ISA.08	The component will perform all communications with the broker, following the NGSI-LD standard, both for context management and subscriptions
T_ISA.09	The component will ensure information exchange among the device/GW and the NGSI-LD broker in a frequent manner
T_ISA.10	The component will ensure real time information exchange of the different types of measurements examined in the project (energy metering/demand/generation, storage, device operational status, indoor environmental and health conditions, occupancy related information)
T_ISA.11	The component will ensure connectivity with different types of device/GW that support MQTT based communication
T_ISA.12	The component will perform all necessary data transformations required in order to ensure semantic interoperability at the NGSI-LD broker level
T_ISA.13	The component will include an internal REST interface for enabling the device registration.
T_ISA.14	The bidirectional communication of this component with 3rd party systems (i.e. IoT devices) will require the use of security mechanisms for the protection of data exchanged.

Table 4 PHOENIX IoT system Adapter - Technical Requirements

Dependencies/ Inputs/Outputs

The PHOENIX IoT system Adapter interfaces with the:

- PHOENIX Real-Time Data Broker to:
  - Update time data associated with the contextual conditions in building environment and the operational status of the different controllable devices
  - Trigger control actions associated with the decisions as defined by the automation module of the component

4.4 PHOENIX Real-Time Data Broker

Overview

The role of this component is to serve as the central context state holder and distribution entity of the PHOENIX platform. Its mission is to receive continuous updates about the building's state and to make sure that updated information reaches the intended parties. It does so by providing a REST API capable of providing query, creation and update functionality for context entities that represent the building information, as well as a subscription mechanism by which interested parties will receive notifications according to their data requirements and needs.

The PHOENIX Real-Time Data Broker, follows the NGSI-LD standard, which defines both the previously mentioned REST-API, as well as the data model to be used for communications and context representation, based in the JSON-LD data format. This format allows us to represent also the semantics, further enhancing the context representation of the information and allowing other components to utilize that information in order to e.g., build knowledge graphs or perform the appropriate inferences.

List of Features

The detailed list of features supported by the component is presented in the following list:

- **Standard based query interface:** a broker offers a REST API by which context can be created, updated, deleted and queried. This API follows the NGSI-LD standard.
- **Standard based model and data handling:** the data model utilized by the broker follows the NGSI-LD standard, which defines both the data representation via JSON-LD as well as the data model and ontologies utilized to represent context data.
- **Context subscription:** the broker also allows context-subscriptions with a wide range of possibilities of filtering; allowing subscribers to receive updates on specific entities, based on type, identifiers and attribute values and ranges. It also allows regular expressions and wildcards as well as literals, to perform the matching of strings.

Component Functional Elements

By presenting the key functionalities, of the Real-Time Data Broker, details about the different functional modules are presented in Figure 6.

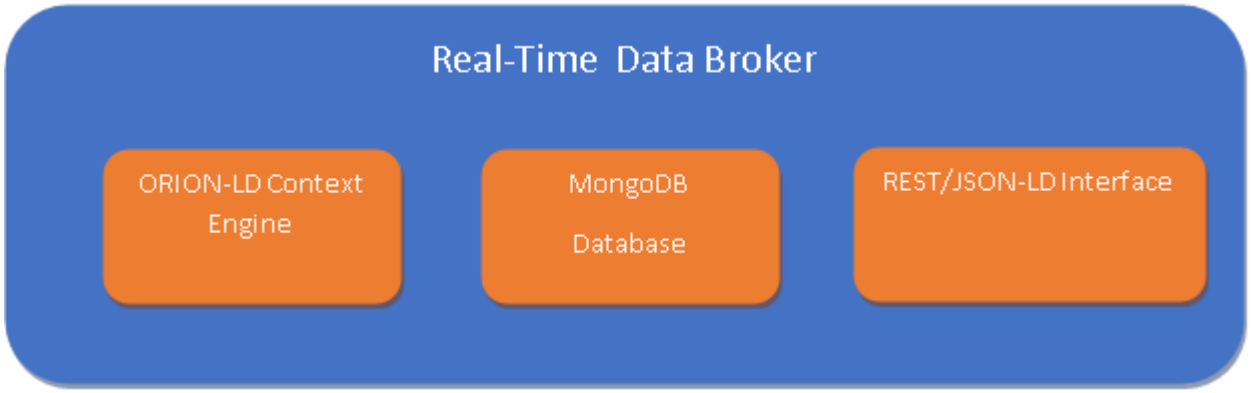

Figure 6 Real-Time Data Broker Overview

More specifically:

- **ORION-LD Context Engine:** Its role is to manage the data publications and subscriptions of the Real-Time Data Broker to allow real-time data exchange with internal components

of PHOENIX platform (e.g. IoT systems, data analytic tools, application services) as well as all internal operations like creation, update, deletion and query of real-time data entities.

- **MongoDB Database:** The role of this database is to store all real-time data published in the Data Broker as well as the configuration information related to the created entities and subscriptions.
- **REST/JSON-LD Interface:** This interface based on NGSI-LD standard enables the bidirectional communication of the Real-Time Data Broker with internal components of PHOENIX platform (e.g., IoT systems, data analytic tools, application services) using a publish/subscribe schema.

Below, the list of requirements for this component is presented:

Req. ID	Description
T_RTB.01	The component should handle context information following the NGSI-LD standard model, including semantic annotations
T_RTB.01	The component should be capable of offering a subscription mechanism allowing different patterns and filtering mechanisms for subscribers to get notifications on specific context information updates (seamless integration of 3rd party data entities to the data streams)
T_RTB.02	The component should offer a REST API both for context management and subscriptions (seamless integration of 3rd party data entities to the data streams)
T_RTB.03	The component should be capable of offering a subscription mechanism following the NGSI-LD standard (state of the art information and communication)
T_RTB.04	The component should offer a REST API following the NGSI-LD standard (state of the art information and communication)
T_RTB.05	The component will ensure information handling and exchange of the different types of measurements examined in the project (energy metering/demand/generation, storage, device operational status, indoor/outdoor environmental and health conditions, occupancy related information, energy prices, building static parameters, building demographics etc..)
T_RTB.06	The component will ensure data management from different types of device/GWs, BMSs, external sources through the interconnection with the adapters defined in the project

T_RTB.07	The component information management framework should be modular enough to integrate additional semantics as defined during the project period
T_RTB.08	The component should be modular enough to enable the provision of additional methods for data retrieval if required during the project period
T_RTB.09	The overall development should be scalable enough in order to ensure integration of multiple data sources (GWs, BMSs, external data sources)
T_RTB.10	The component will enable data discovery and provisioning/registration to support the integration with other PHOENIX component and 3rd party systems through the abovementioned System Adapters.

Table 5 PHOENIX Real-Time Data Broker - Technical Requirements

Dependencies/ Inputs/Outputs

The PHOENIX Real-Time Data Broker acts as a middleware to allow real-time data exchange with other components of PHOENIX platform (e.g. IoT system adapters, data analytic tools, application services). More specifically:

- Inputs from PHOENIX External Data Source Adapter, PHOENIX Building System Adapter and PHOENIX IoT system Adapter pass through the PHOENIX Real-Time Data Broker and further stored in the PHOENIX Platform Data Repository or made available to any other analytic tools and business application services of the project subscribing to the different data streams.

- Inputs from all PHOENIX application services and analytic tools of the project could pass through the PHOENIX Real-Time Data Broker and further stored in the PHOENIX Platform Data Repository or made available to the PHOENIX Building System Adapter and PHOENIX IoT system Adapter for execution (this applies mainly for control actions triggered to the different devices/GWs installed in building premises).

It is evident that the PHOENIX Real-Time Data Broker is acting as a central entity of the PHOENIX platform to ensure seamless communication and information exchange among the different system components. More details about the integration of the different applications to the PHOENIX Real-Time Data Broker are reported in the following sections.

4.5 PHOENIX Platform Data Repository

Overview

The role of this component is to act as the mid-term data storage for temporal series of data. Its mission is to record context information changes (for example building sensor readings) as they happen. It does so by leveraging the subscription mechanism of the PHOENIX Real-Time Data Broker, which will result in a stream of notifications of data entity changes from the broker, which the PHOENIX Platform Data Repository will store in its internal database.

List of Features

The detailed list of features supported by the component is presented in the following list:

- **Query interface mechanism:** The Platform Data Repository offers a REST API by which historical data can be retrieved.
- **Historical data storage and handling:** The Platform Data Repository stores historical information on its internal database.
- **Standard based notification REST end-point:** The Platform Data Repository offers an end-point based mechanism to receive NGSI-LD notifications coming from the broker.

By presenting the core features of the Platform Data Repository, the different modules considered in order to serve the aforementioned functionality are presented.

Component Functional Elements

The PHOENIX Platform Data Repository consists of three functional modules presented in Figure 7.

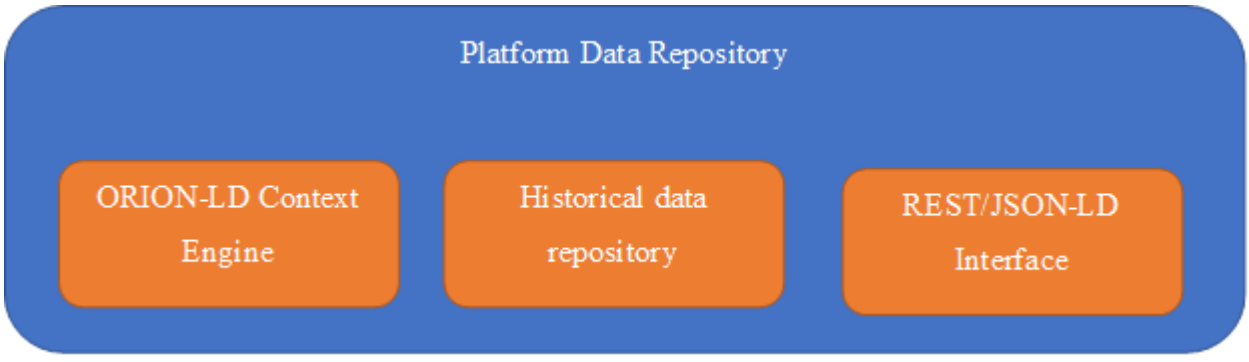

Figure 7 PHOENIX Platform Data Repository Overview

More specifically, the different software modules that consist of the PHOENIX Platform Data Repository are:

- **ORION-LD Context Engine Integration:** Its role is to manage the data publications and subscriptions of the Real-Time Data Broker to allow real-time data exchange with internal components of PHOENIX platform (e.g. IoT systems, data analytic tools, application services) as well as all internal operations like creation, update, deletion and query of real-time data entities.
- **Historical data repository:** The role of this database is to store all real-time data published in the Data Broker as well as the configuration information related to the created entities and subscriptions.
- **REST/JSON Interface:** This interface based on REST/JSON protocols enables the data access and queries from other internal components of PHOENIX platform (e.g., data analytic tools, application services) using a query/response schema.

Following, the list of technical requirements is presented.

Req. ID	Description
T_PDR.01	The component should implement the NGSI-LD standard notification REST end-point
T_PDR.02	The component should record and store the data reported by the Real-Time Data Broker
T_PDR.03	The component should record and store all modifications notified via the notifications end-point
T_PDR.04	The component will ensure information storage of the different types of measurements examined in the project (energy metering/demand/generation, storage, device operational status, indoor/outdoor environmental and health conditions, occupancy related information, energy prices, building static parameters)
T_PDR.05	The component will ensure information storage of the different information entities defined at the application layer of PHOENIX project (and handled by the Real-Time Data Broker)
T_PDR.06	The component should allow access to the historical information recorded for the notified entities, through a REST API (seamless integration of 3rd party data entities to the data management framework)
T_PDR.07	The component should allow access to the historical information recorded following the PHOENIX common information model (state of the art information management)
T_PDR.08	PHOENIX Platform Data Repository shall store the raw data with the appropriate data and metadata annotations
T_PDR.09	A cloud-based data repository shall stand as the data management layer in PHOENIX
T_PDR.10	A data discovery and provisioning service will be supported by the component to support integration of 3rd party application

T_PDR.11	The overall development should be scalable enough in order to ensure storage of information coming from multiple data sources (GWs, BMSs, external data sources)

Table 6 PHOENIX Platform Data Repository - Technical Requirements

Dependencies/ Inputs/Outputs

The PHOENIX Platform Data Repository tightly depends on the PHOENIX Real-Time Data Broker, which sends notifications triggered by context modifications, acting as the only input to the component. Therefore, the PHOENIX Platform Data Repository stores the data that are routed via the PHOENIX Real-Time Data Broker.

On the other hand, the historical data as stored in the PHOENIX Platform Data Repository are further accessible by all other components of the PHOENIX Platform. More details about the different needs are provided on the description of the different application and business components that follows.

4.6 AI-based Knowledge Engine

Overview

The three components of the Knowledge layer are the User-centric services Analytics Engine, the Grid-centric services Analytics Engine and the AI-based Knowledge Engine (as shown in Figure 2). The AI-based Knowledge Engine which is presented in this section will be enabled by the Knowledge Graph (KG), where properties and relationships among different entities (sensors, appliances, users) will be described. The Knowledge Graph will be populated with data coming from different sources (sensors, appliances descriptions, user profiles, etc) by means of mapping and modelling tools. In addition, AI-based methods will continuously analyse the data in the Knowledge Graph in order to derive new data/relations.

Overall, the component will interact with the PHOENIX Integration Layer, where data from the different sources are modelled and stored at both the Real Time Broker and the Platform Data Repository. Moreover, other data sources, such as Building Models and user profiling related information, will be modelled and added into the Knowledge Graph repository. Finally, internal AI-based methods will analyse the data acquired by the different sources, and their outcome will be written back into the Knowledge Graph, thereby making them accessible by the upper layer.

List of Features

The detailed list of features supported by the component is presented in the following list:

- **Data ingestion mechanism:** a REST API mechanism where PHOENIX related data can be used to push into the Knowledge Graph for further analysis. Also, communication with external data sources may be supported as data from other external sources (e.g., building models) can be retrieved by the Knowledge Graph via REST APIs.
- **Knowledge Graph definition:** the data coming from the different sources will be semantically exchange and linked in order to set the PHOENIX Knowledge Graph.
- **AI based Data analysis:** the data available from the Knowledge Graph can be further analysed through user defined analytics techniques in order to extract valuable knowledge that will be further incorporated into the KG.
- **Query interface mechanism:** the data in the Knowledge Graph can be queries via a REST API, supporting also the SPARQL query language for respective data queries.

By presenting the core features of the PHOENIX Knowledge Graph, the different modules considered in order to serve the aforementioned functionality are presented in the following section.

Component Functional Elements

The AI-Base Knowledge Engine has three main subcomponents, as shown below:

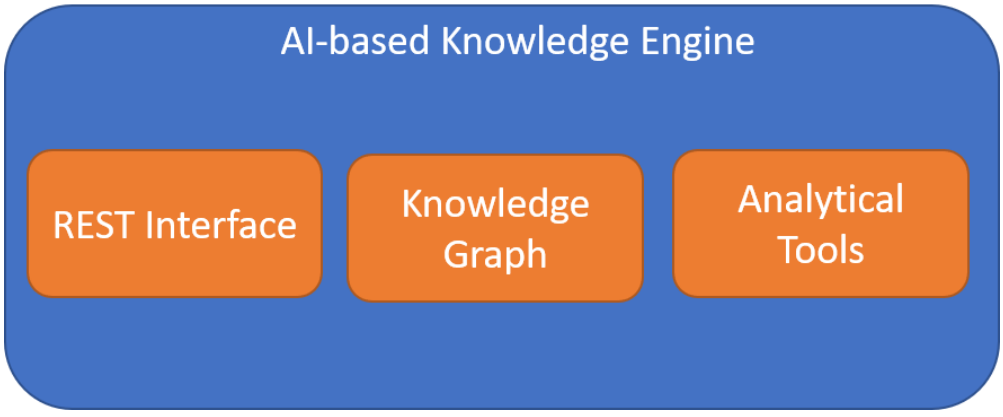

Figure 8 AI-Base Knowledge Engine Overview

The details of the different modules are provided:

- The **REST Interface** mechanism is in charge of the communication with the other components. The component will support both NGSI-LD and RDF/SPARQL APIs, which can be implemented over REST.

- The **Knowledge Graph** is the core of the component. For that, an open-source triple store will be selected and deployed. Data fetched from the Integration Layer (Real time Broker and Platform Data Repository), as well as other sources (e.g., Building models) will be stored in the Knowledge Engine and be made available to other components via the REST interface mechanism presented above.
- The **Analytical Tools** will interface with the Knowledge Graph in both directions: it will retrieve data from the Knowledge Graph that will then be analysed to derive new insights about the data (e.g., events and patterns). The newly derived information, in turn, will be annotated and written back into the Knowledge Graph, so that it can be accessible by the other components.

Following, the list of technical requirements for the AI based KGW is provided.

Req. ID	Description
T_KG.01	The component should implement the NGSI-LD standard notification REST endpoint
T_ KG.02	The component should record and store the data reported by the Real-Time Data Broker
T_ KG.03	The component should record and store all modifications notified via the notifications' endpoint
T_ KG.04	The component will ensure information storage of the different types of measurements examined in the project (energy metering/demand/generation, storage, device operational status, indoor/outdoor environmental and health conditions, occupancy related information, energy prices, building static parameters)
T_ KG.05	The component will ensure information storage of the different attributes (static parameters) required for SRI calculation
T_ KG.06	The component will ensure information storage of the different information entities defined at the function layer of PHOENIX project (and handled by the Real-Time Data Broker)
T_ KG.07	The component should allow access to the historical information recorded for the notified entities, through a REST API (seamless integration of 3rd party data entities to the data management framework)
T_ KG.08	The component should allow access to the historical information recorded following the PHOENIX common information model (state of the art information management)
T_ KG.09	The component should allow the definition of different type of analytics (user defined) over the data handled by the component
T_ KG.10	The component should allow handling of the analytics results (following the internal data structure – KG – of the component) for further utilization by 3rd party applications
T_ KG.11	A data discovery and provisioning service will be supported by the component to support integration of 3rd party application

T_ KG.12	The overall development should be scalable enough in order to ensure storage of information coming from multiple data sources (GWs, BMSs, external data sources) or 3rd party applications.

Table 7 AI-based Knowledge Engine - Technical Requirements

Dependencies/ Inputs/Outputs

The AI-based Knowledge Engine is standing as an application of the PHOENIX platform at the Knowledge layer. The key inputs and outputs are specified:

- PHOENIX Real-Time Data Broker:
    - Get access on real time data associated with the contextual conditions in building environment and further linked to the knowledge graph model.
- PHOENIX Platform Data Repository:
    - Get access on the historical data associated with the contextual conditions in building environment and further linked to the knowledge graph model.

On the other hand, the output linked data are further available to any business application of PHOENIX platform for further exploitation. Both input and output data items that are handled by the Knowledge Graph follow the PHOENIX data model.

Note: Data coming from the application layer may be further available to the knowledge graph for incorporation in the PHOENIX linked data framework. More details about the specific application attributes to be stored in the Knowledge Graph will be provided as part of the development activities of the project.

4.7 User-centric services Analytics Engine

Overview

This is the set of analytics services to allow the development of user-centric services for building's occupants; the user-centric services analytics engine will actively contribute in the provision of data analytics tools towards the assurance of user-centric profiles and comfort-profile preferences. On this basis, the initial scope of this tool is to derive optimal settings on the comfort profiles for each individual building occupant through their initial (provided at each user registration process step on the PHOENIX platform) user profiles, their dynamic actions on sensor devices in the indoor environment, as well as their corresponding direct feedback to the Building Occupants Visualization Dashboard (if any) and the recognition of viable and customized indoor occupancy

conditions. Additionally, user-oriented energy descriptive and predictive analytics will be provided, in order to deliver useful insights to the building occupants.

List of Features

The detailed list of features supported by the component is presented below:

- **Descriptive analytics.** The descriptive analytics on real-time data will be exploited as it is essential for measuring and delivering optimal indoor conditions. Some examples are:
    - Aggregated and historical analytics on Temperature, CO2, humidity, illuminance, and external weather conditions.
    - Total and average building energy consumption, consumption of similar peers and comparative past performance on energy consumption.
    - Aggregated and historical analytics on energy generation and storage (if applicable in the prosumer case).
    - Total and average building energy waste considering the cooling and heating degree days.
    - Relation and patterns of energy consumption with environmental conditions, user actions, demographics, and daily routines.
- **Forecasting analytics:** One of the foundational features of the analytics engine is the deployment of predictive models for efficiently perform predictions on:
    - how the energy consumption is related to other environmental variables.
    - the impact of the notifications and recommendations interventions on energy consumption.
    - the impact of user actions/interactions with the platform on energy consumption.
- **User-driven thermal, visual, occupancy and indoor air quality profiles:** The diverse user profiles will be created and dynamically updated, based on the thermal, visual and air quality established comfort boundaries If the boundaries are violated, a discomfort event is recorded else we assume that comfort is preserved. Occupancy profiling is also made available through providing the appropriate model that labels a state as empty/occupied, since the motion sensors cannot provide a definite categorization on that issue.

Component Functional Elements

By presenting the key features of the component, details about the different functional modules are presented in Figure 9:

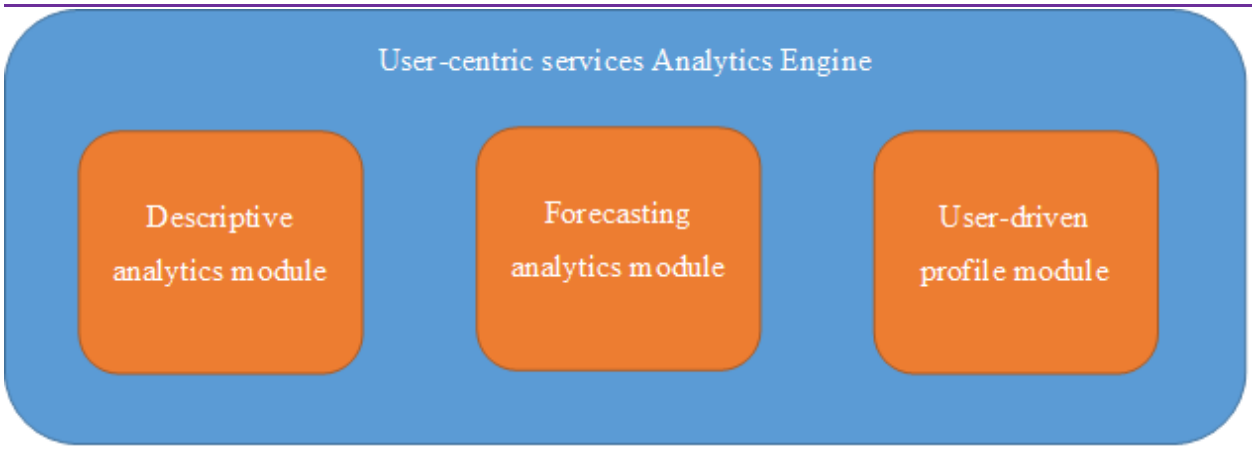

Figure 9 User-centric services Analytics Engine Overview

More specifically:

- **Descriptive analytics module:** The role of this module is the provision of descriptive analytics, as described in the descriptive analytics list of features.
- **Forecasting module:** The role of this module is the provision of ML based predictive analytics, as described in the forecasting analytics list of features.
- **User-driven profile module:** The role of this module is the provision of optimal settings on the user's thermal, visual, air quality and occupancy profiles and the possibility of providing dynamic updates on those profiles near real-time. It interacts with the Comfort, Convenience and Wellbeing Engine because it serves as an input towards the engine (comfort/discomfort events), but it also uses the implicit/explicit feedback from the engine (notifications/control actions) to provide optimizations on the user profiles.

Along with the key elements that consist of the User-centric services Analytics Engine, the list of technical requirements is presented in the table below:

Req. ID	Description
TR_USAE.01	The component should make short term predictions on the occupancy level (occupied/empty)
TR_USAE.02	The component should have access to historic/real-time information about indoor temperature conditions.
TR_USAE.03	The component should have access to historic/real-time information about indoor humidity conditions.
TR_USAE.04	The component should have access to historic/real-time information about indoor CO_2/VOC/PMx conditions.
TR_USAE.05	The component should have access to historic/real-time information about indoor luminance conditions.
TR_USAE.06	The component should have access to historic (at least one year) /real-time information of energy consumption at asset and building level.

TR_USAE.07	The component should have access to historic (at least one year) /real-time information of energy generation at building level.
TR_USAE.08	The component should have access to the building and asset static properties.
TR_USAE.09	The component should have access to real-time and forecasted weather data.
TR_USAE.10	The component should provide aggregated and historical analytics on temperature, humidity, illuminance, IAQ etc...
TR_USAE.11	The component should provide descriptive analytics on the total and average building energy consumption, consumption of similar peers and comparative past performance on energy consumption.
TR_USAE.12	The component should provide aggregated and historical analytics on energy generation and storage where applicable
TR_USAE.13	The component should provide descriptive analytics on the total and average building energy waste considering the cooling and heating degree days.
TR_USAE.14	The component should identify patterns of energy consumption related to environmental conditions, seasonal patterns, demographics, building size etc.
TR_USAE.15	The component should extract the default user's profile based on user's default preferences/settings.
TR_USAE.16	The component should support the extraction of different types of user profiles: thermal, visual, air quality, occupancy.
TR_USAE.17	The component should monitor the comfort limits regarding thermal, visual and air quality preferences.
TR_USAE.18	The component should use the implicit and explicit user feedback/actions to optimize the dynamic user profiles.
TR_USAE.19	The component should use the environmental conditions to optimize the dynamic user profiles.
TR_USAE.20	The component should be able to identify an event as comfort/discomfort event.
TR_USAE.21	The component should make short term predictions of energy consumption based on the indoor and outdoor environmental variables.
TR_USAE.22	The component should make short term predictions on the impact of the notifications and recommendations on energy consumption.
TR_USAE.23	The component should make short term predictions on the impact of user actions/interactions with the platform on energy consumption.

Table 8 User-centric services Analytics Engine - Technical Requirements

Dependencies/ Inputs/Outputs

After the description of the component, it can be deduced that the User-centric services Analytics Engine will have dependencies with other system components of the PHOENIX platform listed below:

- Comfort, Convenience and Wellbeing Engine:
    - To provide updates of the user profiles (both occupancy and comfort related profiles) as calculated by the user-driven profile module.
- Building Occupants Visualization Dashboard:
    - Communicate with the Dashboard server module to provide analytic results and relevant user information.

  - Communicate with the Dashboard server module to ensure access on the list of settings/modes/preferences provided by the end user of the tool.
  - Communicate with the search engine of the Dashboard server module to retrieve any kind of historical data.
- PHOENIX Real-time Data Broker: In order to provide the user centric services analytics, interaction with the PHOENIX Real-Time Data Broker is required. Data flowing from the building environment are made available in order to proceed with the provision of the user analytics services.
- PHOENIX Platform Data Repository: In order to provide the list of user centric services, interaction with the PHOENIX Platform Data Repository is required to ensure access on historical data available from the building environment.

4.8 Grid- centric services Analytics Engine

Overview

In the transforming electricity market environment in Europe, the buildings are considered as active agents that apart from getting energy from the grid when needed are capable of managing their consumption making changes on the schedules and responding to demands of the grid.

With this component the PHOENIX solution will be capable of providing intelligent services that are required in order to support the business functionality of integration of the building with the grid. More specifically, the component will include a variety of algorithms and data analytic techniques to produce the high-level information and added value data that will serve the purposes of the services for grid integration and grid flexibility. Among those methods there will be forecasting and benchmarking techniques based on the available data. More specifically different types of analytics of interest for the incorporation of the buildings as active grid elements are considered, such as demand and generation forecasting, CO_2 intensity of the electricity, battery level forecasting among others. Other analytics such as clustering will be available as the PHOENIX solution could use this, to identify different consumption profiles and with them offering more personalised services to the end users of the platform.

List of Features

The component will support the features detailed in the following list:

- **Data streams benchmarking.** The component will be capable of establishing benchmarks or baselines for the data coming on streams to make possible the services that identify outliers drifting or other anomalies that could be the indication of a rare situation that has to be reported.
- **Forecasting and situation predictions.** For many grid services to be effective they will have to anticipate to a given undesired event. A good example is the one of peak loads. For services aiming at reducing those peaks, it will be necessary to predict them, so demand response events or other mechanism need to be put in place prior their occurrence.
- **Clustering of time series and scalar parameters**. Clustering has been identified as a very powerful tool to identify groups of agents that need to be given the same intervention. For this, the component will have clustering algorithms that will help identifying users' behavioural groups in the case of time series, or groups of elements such as devices or equipment in the case of scalar parameters. This will help with strategies aiming at sending messages to groups of consumers or devices.

Component Functional Elements

As in the previous case, the following figure (Figure 10) show the functional modules of the Grid integration component.

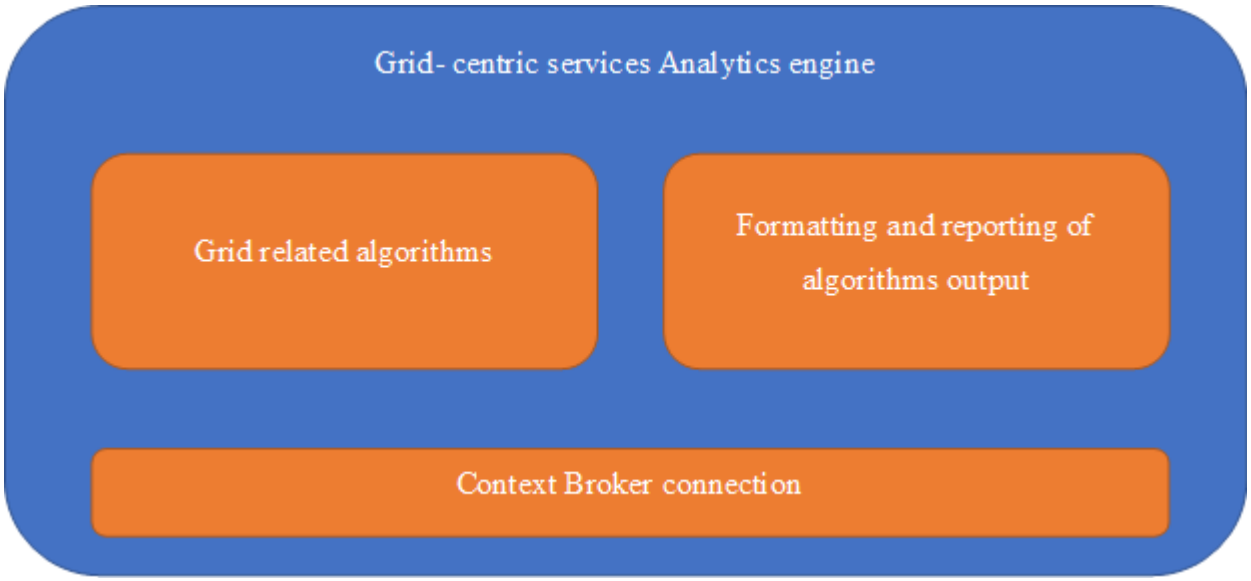

Figure 10 Grid- centric services Analytics Engine Overview

The modules are described here:

- **Context Broker Connection**: The engine will connect to the context broker with two purposes: (1) Obtaining data to feed the algorithms and (2) send the results of the algorithms on a NSGI-LD format to be used by the services.
- **Grid related algorithms**: A variety of algorithms mainly performing forecasting, benchmarking and clustering (non-exclusive) will receive the information from the context broker and add value to the data obtaining extra features.
- **Formatting and reporting to CB of algorithms output**: The engine will have another component that will take the results of the algorithms and put them on the correct format to be added to the context broker, with the correct context. This will make fluid the use of those added value data sets by the services.

A list of technical requirements of the grid flexibility component can be seen on the following table.

Req_ID	Description
TR_GSAE.01	The component will have access to real-time values) of the entities of the context broker
TR_GSAE.02	The component will have access to historical data of the entities of the context broker
TR_GSAE.03	The entities read by the engine will contain information about grid congestion
TR_GSAE.04	The entities read by the engine will contain information about energy generation mix
TR_GSAE.05	The entities read by the engine will contain information about energy demand of the buildings
TR_GSAE.06	The entities read by the engine will contain information about energy use of independent devices
TR_GSAE.07	The entities read by the engine will contain information about energy prices
TR_GSAE.08	The engine will utilize the data formats for demand forecasting at asset and building level (short term)
TR_GSAE.09	The engine will utilize the data formats for generation forecasting (short term)
TR_GSAE.10	The engine will utilize the data formats for storage scheduling forecasting (short term)

TR_GSAE.11	The engine will utilize the data formats for CO2 emissions rate forecasting (short term)
TR_GSAE.12	The engine will utilize the data formats for CO2 emissions rate forecasting (short term)
TR_GSAE.13	The engine will utilize the data for energy generation/consumption benchmarking
TR_GSAE.14	The engine will use the data to apply clustering techniques for further use by the different business applications
TR_GSAE.15	The engine will apply ML based techniques for the extraction of analytics
TR_GSAE.16	The engine will have access to send NSGI-LD formatted data to the context broker with a periodicity of at least a day

Table 9 Grid- centric services Analytics Component - Technical Requirements

Dependencies/ Inputs/ Outputs

The component will have connections with other modules to make it work. This section will explain these connections in terms of inputs and outputs as well as the core features for grid integration:

- PHOENIX Real-Time Data Broker:
    - Get access on real time data related to the situation of the electricity production of the grid: e.g information about the electricity price and the local resources for production (renewable) and storage should be made available.
- PHOENIX Platform Data Repository:
    - Get access on historical time series for electricity related data. Generation, storage evolution, price of electricity or electricity mix among others will be required for the provision of analytics services.

On the other hand, the results of the analytics process are made available to the business applications of the PHOENIX framework, namely:

- Self-consumption optimisation engine.
    - The results of the analytics of this engine will be used by the self-consumption optimisation engine in order to enable the provision of the business functionality.
- Business Stakeholders Interface Engine.
    - Some of the results of this engine will be used by the Business Stakeholders Interface Engine. Aspects such as the forecasting of the demand, the status of the

batteries or the forecasting of price series may be further made available for 3rd party business entities.

4.9 Comfort, Convenience and Wellbeing Engine

Overview

The role of this component is to enable the provision of non-energy services to the building occupants as a key aspect highlighted in the Smart Readiness Indicator methodology. On the basis of user and social driven requirements defined through the analysis at the demo sites, the scope of this component is to first enable identification of poor indoor conditions in buildings that may cause comfort or health issues to the building occupants. Furthermore, the component incorporates smart services that will enable either automated control of the building devices to ensure comfort, health and well-being conditions or context based personalized notifications and messages to building occupants related to the comfort and well-being conditions in premises.

List of Features

The detailed list of features supported by the component is presented in the following:

- **Comfort, Health and Well Being Status Identification.** On the basis of the results of the user centric analytics engine, the component will provide the mechanisms for monitoring the actual comfort and well-being conditions in premises
- **Comfort, Health and Well Being Automation.** By taking into account the contextual conditions in premises, the component will aim to automatize the operation of controllable devices (focus on HVAC, Lights etc..) on the way to ensure the establishment of a comfortable, health and well-conditioned environment in building premises.
- **Comfort, Health and Well Being based Notifications**. In lack of controllable devices in building premises, the scope of the component is to trigger appropriate notifications and messages to the end users in order to trigger their behaviour towards more comfortable, health and well-being conditions in building premises.

Component Functional Elements

By presenting the key functionalities, of the component, details about the different functional modules are presented in Figure 11.

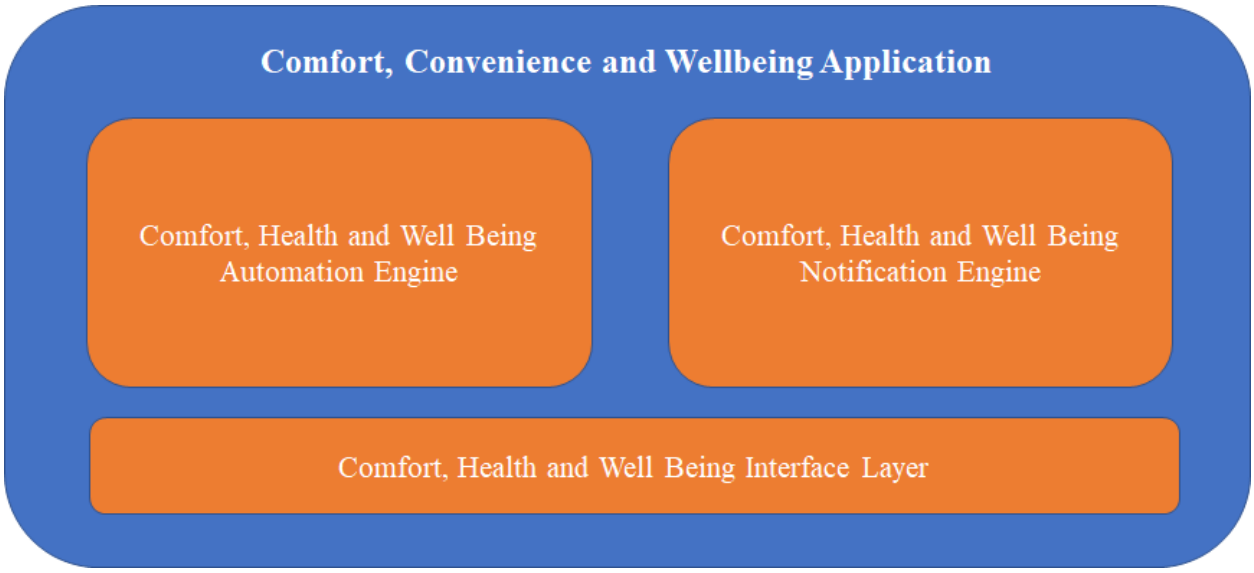

Figure 11 Comfort, Convenience and Wellbeing Engine Overview

More specifically:

- **Comfort, Health and Well Being Interface Layer:** The role of this module is to act as the wrapper of the application in order to ensure information exchange with external components (details provided in the next section).
- **Comfort, Health and Well Being Notification Engine:** The role of this DSS module is to correlate building contextual conditions along with the extracted comfort profiles and user settings in order to generate the appropriate notifications associated with the indoor conditions in premises.
- **Comfort, Health and Well Being Automation Engine:** The role of this DSS module is to correlate building contextual conditions along with the extracted comfort profiles and user settings in order to automatize the operation of controllable devices (focus on HVAC, Lights etc..) on the way to ensure the establishment of a comfortable, health and well-conditioned environment.

Along with the key elements that consist of the Comfort, Convenience and Wellbeing Component, the list of technical requirements is presented.

Req. ID	Description
T_CWA.01	The component should have access on real time information about indoor temperature conditions
T_CWA.02	The component should have access on real time information about indoor humidity conditions
T_CWA.03	The component should have access on real time information about indoor luminance

T_CWA.04	The component should have access on real time information about occupancy level in the building environment
T_CWA.05	The component should have access on real time status of controllable devices (HVAC, lighting, DHW, ventilation)
T_CWA.06	The component should have access on real time information about IAQ conditions (VOC, PM)
T_CWA.07	The component should have access on information about outdoor environmental conditions
T_CWA.08	The component should have access on weather forecast data
T_CWA.09	The component should have access on users demographics (age range, etc) as defined in the project and configured in the system
T_CWA.10	The component should have access on comfort profiling data as periodically extracted by user analytics
T_CWA.11	The component should have access on building occupancy profiling data as periodically extracted by user analytics
T_CWA.12	The component should be able to automate the operation of controllable devices to ensure comfort preservation
T_CWA.13	The component should be able to automate the operation of controllable devices to ensure health conditions in premises
T_CWA.14	The component should be able to automate the operation of controllable devices to ensure users convenience/ occupancy-based automation
T_CWA.15	The component should be able to trigger control actions to the building devices, following automation DSS
T_CWA.16	The component should be able to trigger notifications to ensure comfort preservation
T_CWA.17	The component should be able to trigger notifications to ensure health conditions in premises
T_CWA.18	The component should be able to trigger notifications related to devices available in premises
T_CWA.19	The component should be able to trigger notifications related to the user characteristics of the user in premises
T_CWA.20	The component should be able to trigger notifications related to actual contextual conditions in premises
T_CWA.21	The component should take into account user preferences (e.g., comfort mode, convenience mode etc.…) as provided by the Building Occupants Dashboard
T_CWA.22	The component should take into account user settings (e.g., opt out from automation, notifications etc...) as provided by the Building Occupants Dashboard
T_CWA.23	The component should be able to retrieve the feedback provided by the users to the different notifications in order to fine-tune the DSS

Table 10 Comfort, Convenience and Wellbeing Component – Technical Requirements

Dependencies/ Inputs/Outputs

By defining the core internal functionalities of the component, the dependencies with other system components of the PHOENIX platform along with inputs/outputs are defined in this section. From the presentation of the core features, it is evident that the Comfort, Convenience and Wellbeing Application interfaces with:

- PHOENIX Real-Time Data Broker:
    - Get access on real time data associated with the contextual conditions in building environment and the operational status of the different controllable devices.
    - Trigger control actions associated with the decisions as defined by the automation module of the component.
- PHOENIX Platform Data Repository:
    - Get access on historical contextual conditions at the building environment to calculate the performance indicators related to comfort, convenience and health conditions.
- User-centric Services Analytics Engine to:
    - Get access on the results of the analytics engine related to the extraction of comfort profiles for building occupants in premises.
- Building Occupants Visualization Dashboard:
    - Get access on the list of settings/modes provided by the end user of the tool.
    - Trigger notifications/messages associated with the decisions as defined by the notification module of the component.
    - Get access on the interaction of the user to the respective notification/message for further evaluation.
    - Provide information related to comfort, convenience and health evaluation for visualization.

4.10 Predictive Maintenance Engine

Overview

Part of the added value of the PHOENX platform for building managers will be the possibility of performing predictive maintenance thanks to the data collected from the sensors within building spaces and from the building services (HVAC, ventilation, battery current yield and so forth). The predictive maintenance engine will be highly based on the modelling capabilities of the building smartness hub, to perform the detection of anomalies. It is for this reason that the data used for the PDE will be mainly in the form of time series. This together with the context information and other third-party data such as weather observations will make possible to develop the intelligent algorithms that will inform about the situation of machinery and other devices in the building.

List of Features

The detailed list of features supported by the component is presented in the following:

- **Real time status of the machinery.** The building manager, house owner, or other parties interested would be able to check the status of the machinery in real time, to get a filling of the operation of the systems. An example of this includes the current use by ventilation systems what can be representative of the status of the filter.
- **Intelligent agent for parts replacement.** The system will know at what moment on time devices have been purchased and installed, as they had to be registered on the context broker, this means that a background process of the smartness hub would be able to anticipate on when a component of the services of the building will have to be replaced.
- **Alert system for equipment malfunction**. The observation of real data in real time will allow to evaluate if variables related to the operation of the components of the services are drifting towards inadequate values. That drift can be identified with services based on analyses done on the smartness hub. A system of alarms could then notify the household that some component should be ordered and replaced.

Component Functional Elements

The schema below shows which are the modules of the engine that will make possible the features of predictive maintenance of the PHOENIX solution.

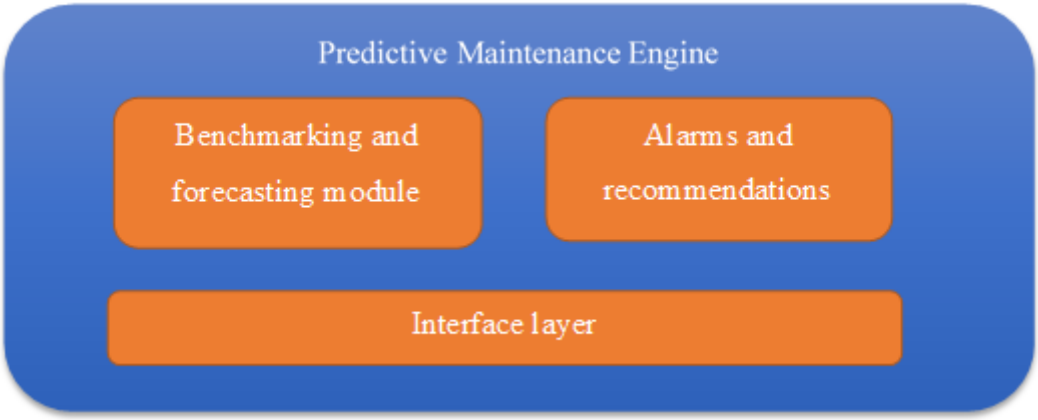

Figure 12 Predictive maintenance modules overview

More specifically:

- **Interface layer:** This layer will integrate the data onto the different components that will perform the required tasks to make possible carrying out the different features. This will

include the time series of the sensors available on the context broker as well as the data models of the components on the building.

- **Benchmarking and forecasting module:** This module will perform the benchmarking of the variables related to maintenance. With this it will be possible to identify the benchmarks at which the components should be operating as well as the detection of anomalies and drifting of the parameters. The periodicity of the algorithms will depend on the operation of the component.
- **Alarms and recommendation module:** To get the most of the predictive maintenance module, it will be necessary to send the high-level information of the smartness hub to the concerning user. This will be done via the dashboard of the solution, and it will be in the form of real time information as well as timed alerts.

Considering the features that this engine should offer, a list of technical requirements is reflected below.

Req. ID	Description
T_PM.01	The component should have access on real time data about components of building via the context broker
T_PM.02	The component should have access to the data models of the buildings and the equipment
T_PM.03	The component should have access on real data about meteorological conditions
T_PM.04	The component should have access on real time information about occupancy level
T_PM.05	The component should have access on real time status of controllable devices (HVAC, lighting, DHW, ventilation)
T_PM.06	The component should have access on real time information about IAQ conditions (VOC, PM)
T_PM.07	The component should be able to obtain actual contextual conditions in premises
T_PM.08	The component should have access to the building smartness hub to extract benchmarking and forecasting results related to maintenance (energy consumption profiling)
T_PM.09	The component should provide services for preventive maintenance for the building systems
T_PM.10	The component should provide services for responsive maintenance for the building systems
T_PM.11	The component should be able to trigger notifications about maintenance related (failure/lack of communication) concerns

Table 11 PHOENX predictive maintenance engine- Technical Requirements

Dependencies/ Inputs/Outputs

The features of the predictive maintenance engine have been reflected above. Although it can be anticipated from the functionalities the list of interfaces with other system components, a detail explanation of the Inputs/Outputs to the components have been explained below.

Application interfaces with:

- PHOENIX Real-Time Data Broker:
    - Get access on real time data associated with the contextual conditions in building environment and the operational status of the different building services components.
- PHOENIX Platform Data Repository:
    - Get access on real time data associated with the contextual conditions in building environment and the operational status of the different building services components. Access on historical data is required for the predictive maintenance models.
- AI-based Knowledge Engine:
    - Get access on the results of the analytics engine related to benchmarking operation of the different devices.

On the other hand, the results of the predictive maintenance engine will be made available to the building occupants (owners and inhabitants). More specifically:

- Building occupants Visualization Dashboard:
    - Get access on the list of settings/modes provided by the end user of the tool.
    - Provide a visualisation of the status of the building de vices on real time.
    - Trigger notifications/messages associated with the decisions as defined by the notification module of the component.

4.11 SRI/ EPC Evaluation Engine

Overview

In the PHOENIX context, using a common language to evaluate energy efficiency and smartness is crucial. In particular, the Energy Performance Certificate (EPC) and the Smart Readiness Indicator (SRI) are considered the best solution to rate the energy performance and smartness in a standardized way. This component will automatically generate the evaluation of the EPC and the SRI of the building considered. Having the evaluations automatically will have great consequences on users' awareness and it will allow understanding at a glance which ones are the strength and weaknesses of the building, hence its energy potential. The component will operate in two different

ways for the calculation of the SRI and for the calculation of the EPC. As the calculation will be done with different methodologies, the engine will require data at two different levels.

For the case of the SRI, the engine will inspect the entities available on the context broker and the Knowledge Graph, and with that, it will be capable of having an estimation of the level at which each one is.

In the case of the EPC, the engine will use data in the form of time series of energy consumption and environmental conditions (such as temperature) in addition to meta data of the building (using the NSGI-LD building data type). With this, the engine will be capable of estimating the energy that is required for maintaining the comfort levels of the building, and with that estimate an in-use EPC. This will be complemented with data concerning the installations, in terms of emissions, efficiency, connectivity and automation.

List of Features

The detailed list of features supported by the component is presented below:

- **Communication with the PHOENIX Real-Time and historical data repositories:** Building data models to get the information about dimensions and other relevant data (e.g., external weather data) can be obtained from the PHOENIX Data Broker, and the Knowledge Graph. From those, the level of smartness of the building can be automatically derived (e.g., actuator active in the context broker to turn off/on the EV charger implies a level 3 on that service).
- **Time series analytics.** There is a great deal of information on the time series collected of a building that can help on the calculation of the EPC. This feature will contribute with the evaluation of them to acquire knowledge of the characteristics of the building. This includes internal temperatures and heating power for envelope efficiency or CO_2 and PIR for occupancy inferring.
- **Smart services' analytics.** According to the SRI framework, data from sensors will give information about each domain considered. Specific data will be required depending on the system considered, regarding in particular how the installations are controlled, monitored, automatized, how information is reported, and so forth.
- **Hardware upgrade recommendation.** The component will point to upgrading methodologies for the devices that are found connected to the context broker. Depending on the device at hand, the user or building manager will be pointed to a given guideline.

Component Functional Elements

The SRI/EPC Evaluation Engine consists of three functional modules, as shown in Figure 13:

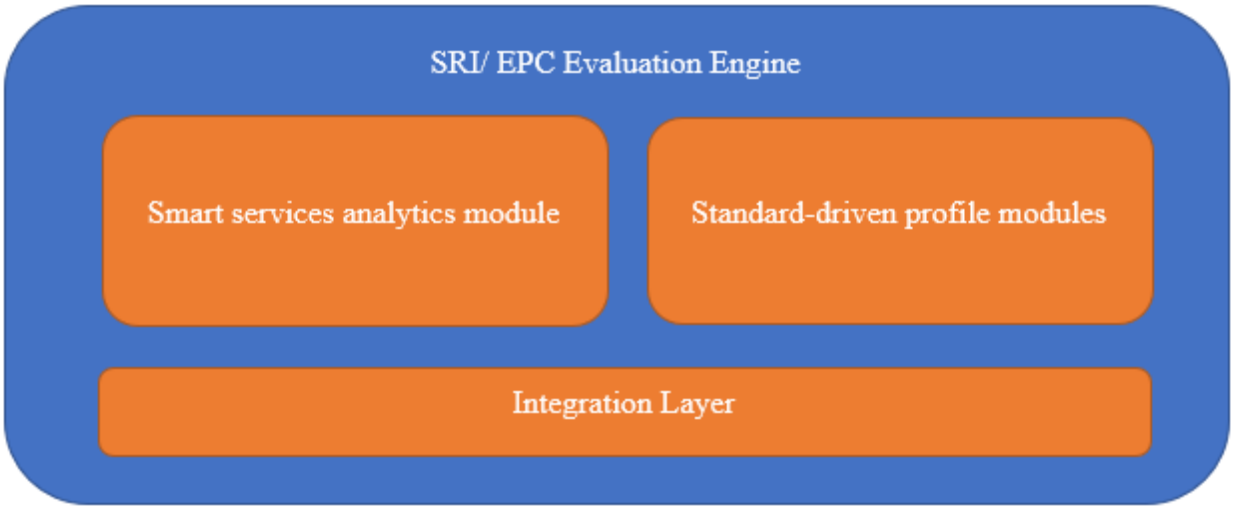

Figure 13 SRI/ EPC Evaluation Engine Overview

More specifically:

- **Smart services' analytics module:** The role of this module is the provision of smart services' analytics that in this case will be the EPC automatic calculation. In addition, some automatic calculation for SRI principles will be considered.
- **Standards-driven profile module:** While EPC evaluation works with measurable values, the SRI framework is structured on levels that depend on characteristics that are hard to measure. For that reason, the Standards-driven profile module will investigate the entities that are available on the Context Broker, and with them, it will identify levels of smartness of the different domains. At the same time, this module will be capable of pointing, based on the status of the devices, to options for upgrading the smartness.
- **Integration Layer:** The component will interact with the Integration and Knowledge Layer of the PHOENIX platform. In particular, the integration layer is an interface where data from the building (consumptions, emission and location data collected in the descriptive analytics module and systems and services' data collected in the smart services' analytics module) can communicate with the standardized criteria and indices of the SRI and EPC framework.

Following, the list of requirements is presented:

Req. ID	Description
T_SRI.01	The component should have access to historic (at least one year) information of energy consumption at asset and building level.
T_SRI.02	The component should have access to historic (at least one year) information of CO_2 emissions
T_SRI.03	The component should have access to the climate zone of the building, weather data of the site, localization of the site (coordinates), degree days, irradiance.
T_SRI.04	The component should have access to available data of the building construction.
T_SRI.05	The component should have access to the energy performance indices of each country involved.
T_SRI.06	The component should have access to the SRI criteria (official spreadsheet)
T_SRI.07	The component should have access to different types of measurements examined in the project (energy metering/demand/generation/storage, device operational status, indoor/outdoor environmental and health conditions, occupancy related information, connection to the grid, electric vehicles)
T_SRI.08	The component should provide estimations of building energy performance considering the KPIs defined also in the regulation
T_SRI.09	The component should provide a comparison between the building's consumptions and emissions and the ranges established by standards (according to the country)
T_SRI.10	The component should provide an automatic assignment of the functionality level for each domain considered, according to the SRI framework.
T_SRI.11	The component should provide a function to facilitate building occupants for the selection of best fitted hardware solutions
T_SRI.12	The component should guide building occupants about technological upgrades that should satisfy their needs
T_SRI.13	The component should promote the installation of costless smart hardware solutions
T_SRI.14	The component should promote the installation of energy preserving or efficient smart hardware solutions

Table 12 PHOENX SRI/ EPC Evaluation Engine - Technical Requirements

Dependencies/ Inputs/Outputs

The dependencies with other modules are described in this section. Again, access on real time and historical data is required in order to support SRI and EPC evaluation. Therefore:

- PHOENIX Real-Time Data Broker
  - Get access to the data associated with the building data models.
  - Get access to the data associated with location and climate zone.

- PHOENIX Platform Data Repository
  - Get access to the historical data associated with the contextual conditions in the building environment.
  - Get access to the data associated with building internal conditions.
  - Get access to the data associated with building energy meters.

- AI-Based Knowledge Graph:
  - Get access to the data associated with the building data models handled as triple store.
  - Get access to the preliminary SRI calculation as performed by the AI-Based. Knowledge Graph.

On the other hand, the results as extracted from SRI and EPC evaluation are made available to the building users for visualization, thus:

- Building Occupants Visualization Dashboard:
  - EPC / SRI information and recommendation: The SRI information and recommendation will serve as input to the interface to illustrate the level of the SRI. Also, it will have links to upgrading advice depending on the situation of the devices.

4.12 Building Occupants Visualization Dashboard

Overview

The main purpose of the Building Occupants Visualization Dashboard is to serve as the user-centric reporting tool of the PHOENIX platform. Thus, the delivery of human-building interaction services to the building occupants will be enabled for everyone to understand, compare and improve their energy metrics and usage. As a basis, this tool will incorporate the visualization of the historic, aggregated and real-time data context from metering and sensing of smart devices across the entire topology landscape. All the descriptive and predictive analytics results mentioned in 4.7 will also be visualised in the dashboard. The user will also be able to make queries on the static building configuration such as select specific zones, rooms, sensors per room etc. On top of

these features, user-driven comfort-profile preferences of each occupant will be exposed in the dashboard, so as to comprise the insertion point for the entire PHOENIX platform leading to ensured comfort preservation and well-being without causing discomfort. Based on these preferences, notification/messages will be depicted as originated from optimized decisions from the Comfort, Convenience and Wellbeing. Additionally, information regarding predictive maintenance alerts, SRI/EPC indicators regarding the building performance, demand flexibility KPIs, parameters regarding smart contract offers and suggestions on self-consumption optimization will also be made available, thus a concrete list of energy and non-energy services to the building occupant will be offered via a friendly and intuitive UI.

List of Features.

The detailed list of features supported by the component is presented below:

- **Sensor data and energy KPIs visualization:** The Building Occupants Visualization Dashboard will be used to visualize the real-time sensor data and energy KPIs, collected from the entire infrastructure topology.
- **Descriptive and predictive analytics results and graphs visualization:** All the analytics results from the User-centric services Analytics Engine will be made available in the dashboard.
- **Provision of user personal optimal settings and preferences**: Upon registration, user will have the opportunity to interact with the dashboard by defining his default preferences on thermal, luminance and air quality conditions as well as give information related to his every day schedule such as time spent at home, work day schedule etc.
- **Depiction of comfort-based and user-centric notifications**. The component will display notifications/messages associated with the decisions on altering the contextual conditions in the building, regarding several comfort-based conditions. Additionally, interaction of the user to the respective notification/message will be enabled so as to optimize the desired indoor conditions based on the recorded uncomfortable sensations.
- **Visualization of services regarding user-building-grid interaction.** The visualization of any building maintenance alerts, SRI and EPC building indicators, information on flexibility smart contracts signed between a retailer and an occupant along with the demand flexibility KPIs for the participation in demand response events and energy efficiency KPIs regarding the self-consumption optimization of prosumers, are examples of such important user-building-grid interactions.

Component Functional Elements

By presenting the key functionalities, of the component, details about the different functional modules are presented in Figure 14:

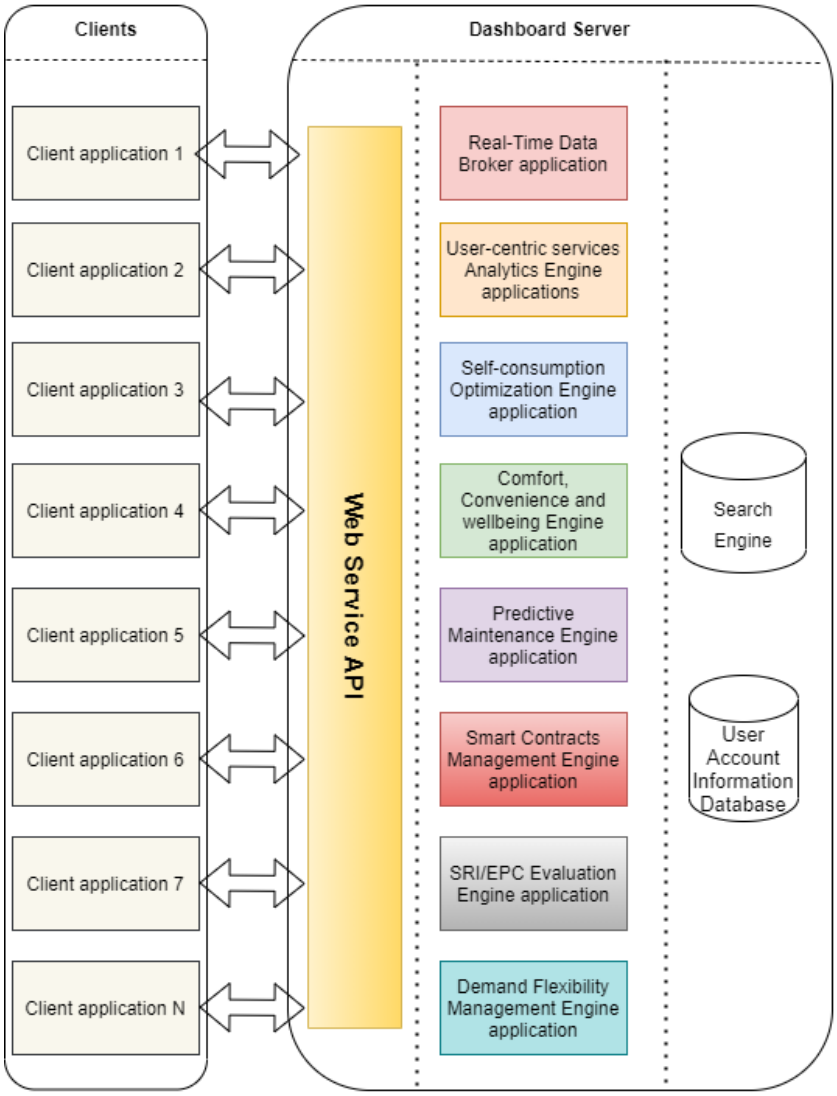

Figure 14 Building Occupants Visualization Dashboard Overview

More specifically:

- **Client module hosts all the client applications:** This module is responsible to deliver the front-end part of the user dashboard web application in an intuitive and easy to use way. The diverse background of the users is taken into account for the design of the front-end UI. The design is in such a way that different client applications can make requests to different backend application services.
- **Dashboard server module hosts all the backend applications:** Based on a three-layer architecture as seen from left to right of Figure 14, (web service API layer, micro-services

layer, database layer), this module serves as the backbone of the dashboard. It forwards all the static and dynamic user requests either to be handled by internal application engines (user-centric services analytics engine, self-consumption optimization engine) or by external servers. After processing, the results are forwarded respectively to the client applications. The server module includes two different databases in its database layer. The User Account Information Database holds the user data upon registration to the user dashboard like personal data, optimal comfort settings, optimal control settings, optimal savings settings etc. The Search Engine is a replica of the platform data repository and is used for low latency optimized user queries.

Along with the key elements that consist of the Building Occupants Visualization Dashboard, the list of technical requirements is presented below:

Req. ID	Description
TR_BOV.01	The component should present historic/real time information about indoor temperature conditions.
TR_BOV.02	The component should present historic/real time information about indoor humidity conditions.
TR_BOV.03	The component should present historic/real time information about indoor luminance.
TR_BOV.04	The component should present historic/real time information about CO2, VOC etc. level in the building environment.
TR_BOV.05	The component should present real-time and forecast weather data (outdoor conditions).
TR_BOV.06	The component should enable user registration, user login, user authentication and user password reset.
TR_BOV.07	The component should provide access to a simple user questionnaire with basic questions regarding demographic details, interaction with the Phoenix platform settings and comfort preference settings.
TR_BOV.08	The component should provide a simple “at a glance” page with a summary of the main energy and non-energy KPIs.
TR_BOV.09	The component should provide the ability to update general user login profile information.
TR_BOV.10	The component should provide the ability to add and change user settings such as comfort preferences, platform interaction preferences and building areas preferences.
TR_BOV.11	The component should give the choice for the user to retrieve static building information like rooms/zones per building, available sensors per room, available control actions per sensor.
TR_BOV.12	The component should provide real and historical sensor data feeds about raw sensor data.
TR_BOV.13	The component should depict descriptive and predictive analytics regarding the building/dwelling energy consumption
TR_BOV.14	The component should depict descriptive and predictive analytics regarding the building energy generation and self-consumption.

TR_BOV.15	The component should depict information about energy and cost savings at specific time frames
TR_BOV.16	The component should depict the total energy waste and waste per device type.
TR_BOV.17	The component should depict information to building occupants to increase awareness about energy efficiency and smart programs
TR_BOV.18	The component should provide real time/aggregated information to building occupants about comfort levels (thermal, visual, air quality)
TR_BOV.19	The component should depict notifications to ensure health conditions, coming as recommendations from the Comfort, Convenience and Wellbeing Engine.
TR_BOV.20	The component should provide the user options of opt-out, manual, automated, remote device control and scheduling device operations.
TR_BOV.21	The component should provide information on the building EPC indicators.
TR_BOV.22	The component should provide information on the building SRI indicators.
TR_BOV.23	The component should depict maintenance alerts if any of the sensors monitoring building services component is moving towards unacceptable values.
TR_BOV.24	The component should provide information on flexibility smart contracts parameters and billing so the USEF interpreter has all parameters to send a standardized message of transaction.
TR_BOV.25	The component should give information on the flexible assets of the dwelling and on the flexibility events i.e., the devices that the user is willing to let control by the utility to provide flexibility.
TR_BOV.26	The component should provide all measurements updates at real time, day, week and month basis.
TR_BOV.27	The component should provide a friendly web app interface so that it can be addressed to a variety of users regardless of their age, educational level, computer literacy level etc.
TR_BOV.28	The component should consider the coverage of GDPR requirements when required, considering the applicability of the regulation in all demo countries.
TR_BOV.29	The component should provide access to building occupants via different means (mobile devices, web etc..)
TR_BOV.30	The component should by design aid users to understand the different features provided by the applications.
TR_BOV.31	Application localization should be supported by the different applications for the building occupants

Table 13 Building Occupants Visualization Dashboard - Technical Requirements

Dependencies/ Inputs/Outputs

After the description of the component, it can be followed that the Building Occupants Visualization Dashboard will have dependencies with other system components of the PHOENIX platform, listed below:

- PHOENIX Real-Time Data Broker:
    - Real-time data retrieval of building context through the available sensors, devices and data sources
    - Control actions associated with the decisions as defined by the Comfort, Convenience and Wellbeing Engine
- User-centric Services Analytics Engine:
    - Receive and illustrate historical graphs
    - Receive and illustrate results related to the extraction of comfort profiles for building occupants in premises
    - Receive and illustrate descriptive and predictive results on energy and non-energy KPIs

In addition, interaction with the business applications of the PHOENIX framework is also considered:

- Comfort, Convenience and Wellbeing Engine:
    - Forward the diverse generated notifications from the Comfort, Convenience and Wellbeing Engine to the users
    - Provide the feedback of the user on the diverse notifications to the Comfort, Convenience and Wellbeing Engine
- Self-consumption Optimization Engine:
    - Receive and illustrate results on energy generation, energy storage and energy self-consumption KPIs
- Predictive Maintenance Engine:
    - Receive and illustrate maintenance alerts
    - Real-time monitoring of the key devices for maintenance
    - Depict basic analytics such as early detection of device malfunctions
- Smart Contracts Management Engine
    - Depict the transaction records between energy parties that have agreed on a smart contract- Log of transactions being done
    - Provide the ability for the user to define and view relevant smart contract parameters such as contract duration and terms

- SRI/EPC Evaluation Engine
    - Show the SRI level of each device as calculated by the respective engine
    - Show the SRI level per SRI domain as calculated by the respective engine
    - Show the EPC as automatically calculated by the respective engine
    - Dedicated page to upload excel file with SRI, and visualization of the SRI matrices and final score together with an automatically calculated SRI based on entities of the CB
    - Navigation potential to recommendations for SRI upgrading
- Demand Flexibility Management Engine
    - Dedicated page to visualize potential flexibility of the building offered by the devices of the user

4.13 Smart Contracts Management Engine

Overview

The smart contract management engine will be designed to make possible the use of the PHOENIX solution as the platform to perform contracting of energy products between the different parts. These have been anticipated to be building users with other business entities such as building managers/ESCOs, retailers and aggregators among others. The smart contracts will consist of energy related business transactions carried out among the different parties. The PHOENIX solution will use the USEF framework as the canonical lingua in which those transactions are done, and a secured log based on security mechanisms will make possible the management of those smart contracts on a safe and sound way. A USEF-like interpreter will be created so the transactions can be sent to the context broker of PHOENIX to make it aware of the different contracts.

List of Features

The list of features of this component will be the following:

- **USEF compatible module for transactions communication.** This module will use part of the USEF flexibility framework specifications to produce messages on XML format between the different agents of a transaction. The module will also “inform” the context broker of the transactions performed to facilitate the application of AI on the transactions (timing, type, quantities, prices…).

- **Security mechanisms for supporting the auditing and trust between stakeholders**. The transactions on PHOENIX will have the technology to make possible the full trust among stakeholders. The mechanisms will ensure transparency and fairness in the process.

Component Functional Elements

The following schema (Figure 15) shows how the engine for smart contracts have been designed in the PHOENIX solution.

Figure 15 PHOENIX Smart Contracts Management Engine

The description of the functional elements is as it follows:

- **USEF communication module:** The USEF communication module will use the USEF standard to produce contractual messages related to energy transactions. Different types of fine-grained contract types will be defined during the demonstration activities of the project.
- **USEF interpreter for PHOENIX Context Broker:** The transactions performed using the USEF language will be performed within the PHEONIX platform, in that way, the tools that exist on the building smartness hub could be exploited. As the Context Broker has its own language (NSGI-LD) the interpreter will be in charge of making the real time communication between the USEF interpreter and the PHOENIX Context Broker.
- **Interface layer for Smart Contracts:** The smart contract engine will have connection to other third-party components to make possible the realisation of its tasks. This includes the

dashboard of the platform, in which the users could voluntarily decide to perform transactions with other agents.

In the following, the technical requirements for the Smart Contracts component have been detailed.

Req. ID	Description
T_SC.01	The component needs to be in connection with the context broker in real time
T_SC.02	The component needs connection to the dashboard to facilitate performing the transactions to the building users
T_SC.03	The component needs connection to the dashboard to facilitate performing the transactions to the building managers
T_SC.04	The component needs connection to the dashboard to facilitate performing the transactions to the grid operators
T_SC.05	The component needs connection to the dashboard to facilitate performing the transactions to the aggregators
T_SC.06	The component needs to gather data in real time of the electricity price each hour
T_SC.07	The component needs to be secured in terms of access, making possible the access to different options depending on the role of the users of the platform
T_SC.08	The component needs to be secured in a way that the privacy of the users is preserved
T_SC.09	The component needs to fulfil the standard format of messaging between actors of the new electricity paradigm (smart grid)
T_SC.10	The component needs to be capable of archiving the transactions for accounting purposes
T_SC.11	The component needs to fulfil the standard format of contract among the different entities
T_SC.12	The component needs to provide a flexible template for the establishment of contractual agreements among the different entities.

Table 14 Smart contracts engine - Technical Requirements

Dependencies/ Inputs/Outputs

For the core functionalities of the Smart Contract Engine to work, it is necessary to have a series of dependencies with other components as well as receiving inputs and outputs from other sources. The details of these connections are shown below:

- Building Occupants/users Visualization Dashboard:
    - Inform building users about the possibilities of triggering smart contracts to sell or purchase kilowatts or “negawatts” with other agents.
    - Trigger notifications of contracts occurring by the near-by agents and that can affect the user.

- Business Stakeholder Interface Engine:
    - Inform business actors about the possibilities of triggering smart contracts to the building occupants.

We have to point out that the contractual details are provided as static configuration parameters for the following business applications (demand flexibility management, self-consumption optimization). Nevertheless, as this is static configuration the detailed interfaces are not explicitly named in the following sections.

4.14 Demand Flexibility Management Engine

Overview

On the new paradigm, where buildings are active agents of the electricity ecosystem, one of the most relevant contributions of them is the demand flexibility. Demand flexibility is the capability of the consumers of changing their demand at their will. This is done to reduce peaks of the power consumption, and it is in most cases incentivised via some financial incentive. Several variants of the programs have been designed and tested but as defined in the DoA, direct load control is the demand side management strategy to be examined in PHOENIX project.

The PHOENIX solution needs an engine to make possible the management of demand flexibility. Once the building has a variety of devices that are connected to the PHOENIX platform, it will be possible to read the parameters that show the potential actuation over them. Examples of these on the PHOENIX pilots are: Electric Vehicle Charging, Air conditioners thermostat, ventilation equipment, and diesel generators. The actuation over these devices it will make possible the demand flexibility. The demand flexibility engine, will allow to calculate the demand flexibility via the integration of the devices over which the PHOENIX platform can actuate.

List of Features

The demand Flexibility engine will have the following list of features:

- **Identification of potential automatic actuations that result on Demand Flexibility:** The engine will include a system to recognize the devices that have actuation within the context of the project. The devices will open the door of demand flexibility on an automatic manner using the PHOENIX platform exclusively without human intervention.

- **Module for manual demand flexibility potential:** The user is the ultimate interface for actuation of devices. The Demand Flexibility engine, will need a module that will have the option to include manual changes on demand thanks to availability of flexibility due to the willingness of the occupant to change behaviour. Examples of these can be the setting up of the washing machine or the dishwasher.
- **Demand flexibility triggering system:** The demand flexibility triggering system will perform the necessary calculations to identify schedules that can result on the demand reduction asked by the demand response program. The demand flexibility triggering system will perform the calculations and notify either the context broker (automation) or the user (manual) to accomplish the reduction on demand.

Component Functional Elements

By presenting the key functionalities, of the component, details about the different functional modules are presented in Figure 16.

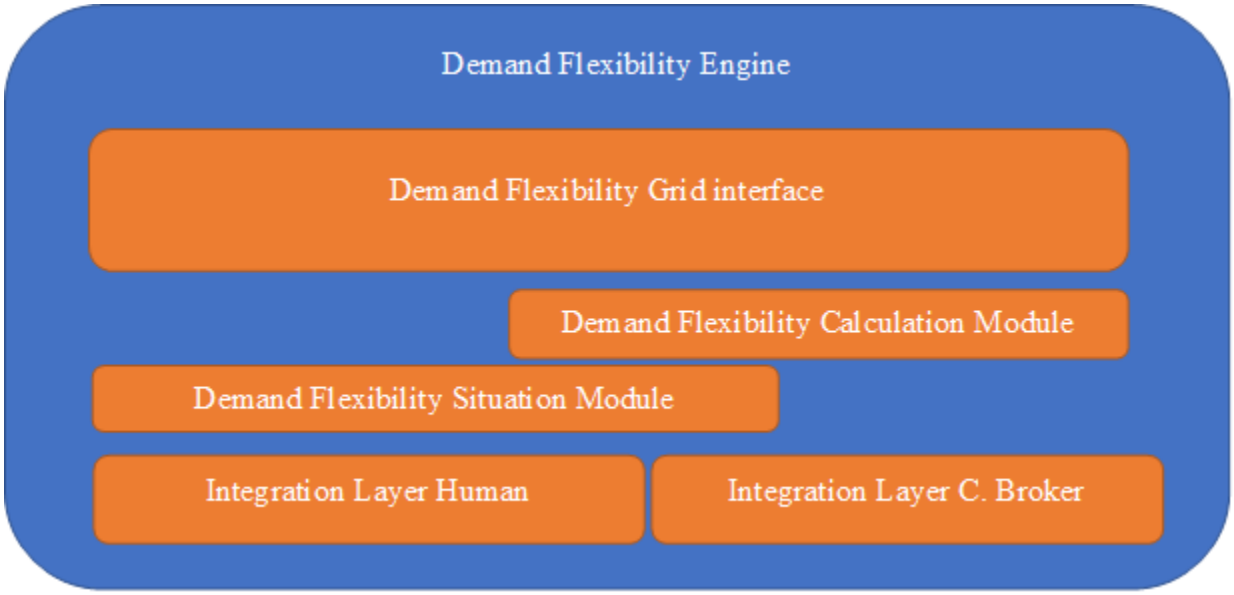

Figure 16 Demand Flexibility Engine Overview

The components are as it follows:

- **Demand Flexibility Grid interface:** For demand response programs it is important that the grid is capable of sensing signals that imply that a demand response event needs to occur. In addition to that, and to enrich even more the communication with the grid. We have anticipated the possibility of the grid of being informed about the level of flexibility

that is possible in each one of the buildings so the demand response events can be designed individually.

- **Demand Flexibility Calculation module:** The PHOENIX platform has to respond to a Demand Response event triggered by a 3rd party entity (aggregator following grid request), by generating its own device level demand response events. For this, it will be necessary a stage in which the demand to be suppressed is calculated and identify the way in which it is accomplished.
- **Demand Flexibility Situation module:** the engine will require a module that is capable of reading the potential flexibility that exists in the building, either by automatically identifying flexibility sources in the context broker or via the input of the users.
- **Integration Layer human / Context Broker:** Either if the operation of the given system is changed via the context broker (auto control) or by the user (manual control), it will be necessary to create the order. This will need to happen thanks to a module that will design the order and send it to the right agent to execute the demand response event.

The definition of the Demand Flexibility Engine suggested the following technical requirements.

Req. ID	Description
T_DFE.01	The component should have access to historic information of energy consumption at asset and building level
T_DFE.02	The component should have access to data of the devices in the building
T_DFE.03	The component should have access to the nominal characteristics of controllable devices
T_DFE.04	The component should have access to the baseline analytics results about the typical operation of the different devise
T_DFE.05	The component should have the possibility of interfering with the devices via the context broker
T_DFE.06	The component should provide recommendations of actuations under a demand response event
T_DFE.07	The component should provide an automatic assignment of the flexibility based on devices installed on the context broker
T_DFE.08	The component should have access to 3rd party entities requesting flexibility from the building environment
T_DFE.09	The component should have access to pricing of the electricity on an hourly basis.
T_DFE.10	The engine needs to have connection to the dashboard to inform users about flexibility status
T_DFE.11	The engine will be capable of sending real time recommendations for demand flexibility events

Table 15 Demand Flexibility Engine - Technical Requirements

Dependencies/ Inputs/Outputs

The dependencies with other modules are described in this section.

- PHOENIX Platform Data Repository
  - Get access to the historical data associated with consumption of the different devices that could provide flexibility
- PHOENIX Real Time Data Broker.
  - Get access to the data associated with the devices data models to estimate flexibility potential.
  - Trigger control strategies to the controllable devices in case of demand side management strategies
- Grid - centric services Analytics Engine
  - Get access to the baseline information required for the extraction of demand flexibility profiling of controllable assets
- Business Stakeholder Interface Engine
  - Get access to (grid related) demand response requests triggered by 3rd party entities (e.g. Aggregator)
- Building Occupants Visualization Dashboard
  - Information about the demand flexibility potential at the building environment to be made available through the dashboard to the building occupants

4.15 Self-Consumption Optimization Engine

Overview

The self-consumption optimization engine is responsible to provide energy savings optimization (towards the energy utility operators as well as the building occupants) through optimizing the day-ahead forecasting of energy generation, energy storage and energy consumption at building level. For the clear formulation of the optimization problem both the important static properties of the DERs (PV, EV, battery) and demand side as well as their dynamic measurements will be included in the optimization problem. In a problem like this, the goal is to use as much self-generated energy as possible and lose as little as possible by giving back to the grid. In case no local micro-generation is available, the demand side flexibility will be exploited in order to ensure maximum energy savings. This more flexible relationship between the building and the grid will be evaluated in detail, as a great level of savings in energy use and in infrastructure (at both sides) may occur if done correctly.

List of Features

The detailed list of features supported by the component is presented below:

- **Current and day-ahead optimizations for energy generation, storage and demand:** Data analytics methods are made available, in order to maximize energy efficiency and thus energy savings, both from the perspective of the energy stakeholders as well as the building occupants.
- **Energy services promotion:** The component will provide a mechanism to promote energy savings through the maximization of self-consumption by de-prioritizing for example flexibility or comfort services (a ranking system according to what services will be available in each demo site).
- **Reporting of relevant information to ESCOs/Utilities:** The component will gather, process select and feed the Business Stakeholder Interface Engine with any information that is relevant to specific energy stakeholders.

Component Functional Elements

The next figure shows the functional elements of the self-consumption optimization engine:

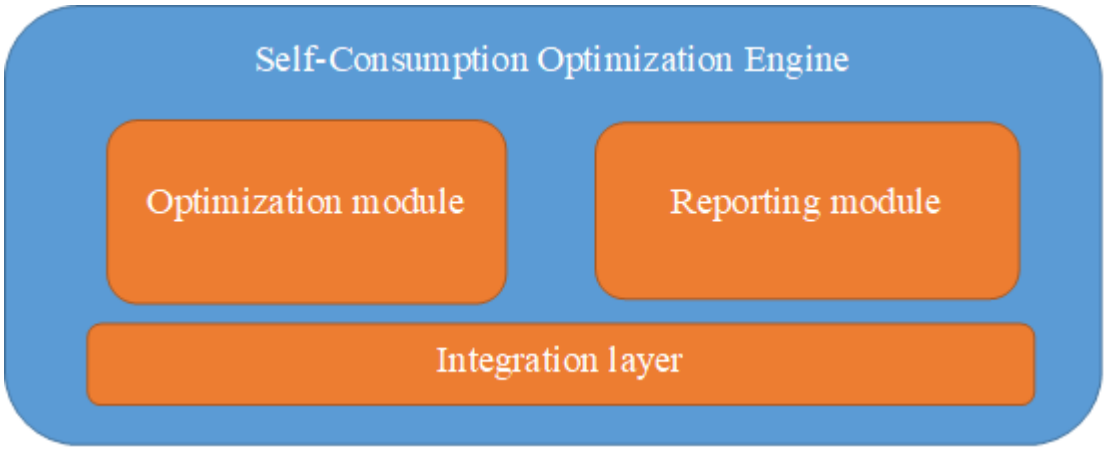

Figure 17 The architecture of the self-consumption optimization engine

The list of modules that consist of the self-consumption optimization engine are provided:

- **Reporting module:** This module acts as data integrator, processor and reporting tool of the historical and/or real data needed, to visualize information both in the building occupant's visualization dashboard as well as the business stakeholder interface engine.
- **Optimization module**: This module is responsible to deliver the multi-parameter optimization algorithm taking into account all the static parameters of PV, EV, battery, demand side systems and dynamic energy generation, storage and consumption. The relation of the dynamic parameters with weather conditions and any seasonality effects will also be taken into account.

- **Integration layer:** The role of this module is to ensure integration with the data sources required for the optimization process. Information from the building environment as well as from the different analytics services will be made available for further exploitation.

The list of technical requirements for the self-consumption optimization engine is listed below:

Req. ID	Description
TR_SCOE.01	The component should have access to real data information regarding weather, energy generation, energy storage and energy consumption entities.
TR_SCOE.02	The component should have access to all historic data information regarding weather, energy generation, energy storage and energy consumption entities (at least 1 year's data at an hourly level).
TR_SCOE.03	The component should have access to forecast data information regarding weather, energy generation, energy storage and energy consumption entities (day ahead at an hourly level) as available from external weather services/ grid centered analytics services
TR_SCOE.04	The component should have access to demand flexibility related data in order to seize the controllability potential of the demand side elements of the building
TR_SCOE.05	The component should extract the static data assets properties to be used as constraints in the optimization problem.
TR_SCOE.06	The component should provide services to ESCOs to ensure maximum self-consumption as a Service.
TR_SCOE.07	The component should provide self-consumption optimization forecasts to the ESCOs at a day-ahead level.
TR_SCOE.08	The component should provide the mechanism to promote energy savings from the self-consumption optimization when necessary.
TR_SCOE.09	The component should be able to trigger the control actions in order to ensure maximum self-consumption.
TR_SCOE.10	The component should provide information to the Building Occupants Visualization Dashboard for reporting purposes.
TR_SCOE.11	The component should provide information to the Business Stakeholder Interface Engine for reporting purposes.

Table 16 Self-Consumption Optimization Engine - Technical Requirements

Dependencies/ Inputs/Outputs

The self-consumption optimization engine will have dependencies with other system components of the PHOENIX platform, listed below:

- PHOENIX Real-Time Data Broker
    - Get access on the energy consumption, generation and storage information entities.
    - Trigger control actions to the controllable devices as extracted from the optimization process on the way to maximize self-consumption/ increase energy savings.
- PHOENIX Platform Data Repository:
    - Get access on the history of data about energy consumption, generation and storage.
- Grid - centric services Analytics Engine
    - Get access on analytic results provided by the engine such as the demand consumption, generation forecasting required for the optimization process.
- Demand Flexibility Management Engine
    - Get access on analytic results provided by the flexibility management engine to use in the optimization problem.

On the other hand, KPIs related to the optimization process will be made available to the different business actors of the project:

- Building Occupants Visualization Dashboard:
    - Provide relevant information regarding self-consumption to the building occupants via the dashboard.
- Business Stakeholder Interface Engine:
    - Provide reports to the energy stakeholder (ESCO in this case scenario).

4.16 Business Stakeholder Interface Engine

Overview

The scope of this component is to act as the business layer for the grid-oriented services as presented above. The scope of this component is twofold. At first to provide a graphical user interface with reports for the business stakeholders addressed by the PHOENIX. This could be of use for agents such as ESCOs or aggregators that may appear in the near future on the European energy market. In addition, the role of this component is to act as the unified interface with third party business entities interacting with the different business services as presented above.

Component Functional Elements

The detailed list of features supported by the component is presented below:

- **Reporting of relevant information to ESCOs/Utilities:** The component will present any information that is relevant to specific energy stakeholders taking into account feedback gathered from the grid related business components as presented above.
- **API Interface for third party business entities.** In order to examine the services mentioned above (smart contracts establishment, demand side management strategies), input from third party entities is required. Towards this direction, this interface will facilitate third party entities to provide the business parameters required for the functionality as specified in previous components.

It is evident that this component is acting as a supportive component to the building-oriented functionality of the PHOENIX platform in order to facilitate the participation of the building as an active entity to grid related business scenarios.

Component Functional Elements

The next figure shows the functional elements of the Business Stakeholder Interface Engine:

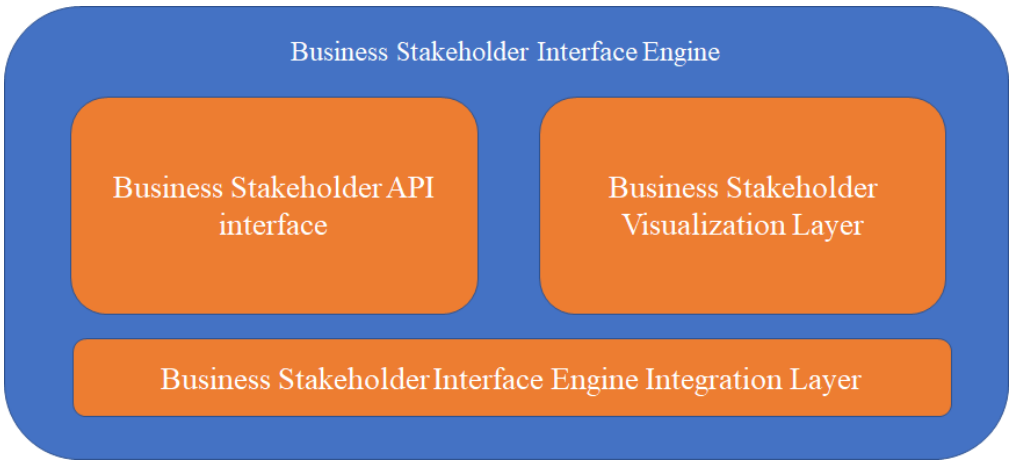

Figure 18 Business Stakeholder Interface engine Overview

The list of modules that consist of the Business Stakeholder Interface Engine are provided:

- **Business Stakeholder Reporting module:** This module acts as the visualization layer for the business stakeholders. This component will consume data as extracted from the grid related business components and then visualize information to the business actors of the project.
- **Business Stakeholder API interface**: Apart from information visualization, the component should facilitate 3rd party business entities (ESCOs/ Aggregators) to set the parameters as interaction with the building environment. Information about smart contracts

management as well as interface for demand response strategies should be available and thus the role of this module is to facilitate the execution of the different grid-triggered business scenarios examined in the project.

- **Business Stakeholder Interface Engine Integration Layer:** Similar to the other components, this is the integration layer to ensure integration and data handling with other components of the PHOENIX platform.

This is a stand-alone component that has the following list of technical requirements:

Req. ID	Description
TR_BSIE.01	The engine needs to have access to grid related analytics results (demand forecasting) to be further utilized by 3rd party business entities
TR_BSIE.02	The engine needs to have access to grid related analytics results (generation forecasting) to be further utilized by 3rd party business entities
TR_BSIE.03	The engine needs to have access to the results of demand flexibility engine for further visualization at portfolio level
TR_BSIE.04	The engine needs to have access to grid related clustering analytics results to be further utilized by 3rd party business entities
TR_BSIE.05	The engine will need access to the results of self-consumption optimization engine (demand schedule, storage schedule) to be made available to the business stakeholders
TR_BSIE.06	The engine needs to have access to the Smart Contract Engine to facilitate the contractual process; contract offers and contract management process should be made available from this component (business stakeholder side)
TR_BSIE.07	The engine will need connections with the grid operator/ 3rd party business entity to trigger demand side management campaigns
TR_BSIE.08	The component should provide all measurements updates at real time and daily
TR_BSIE.09	The component should by design aid users to understand the different features provided by the applications.

Table 17 Business Stakeholder Interface Engine - Technical Requirements

Dependencies/ Inputs/Outputs

As stated above, the Business Stakeholder Interface Engine will have dependencies with the grid related business components of the PHOENIX platform, listed below:

- Smart Contracts Management Engine:
    - The business stakeholder should be able to trigger the contractual process as well as to have access to information about potential smart contracts. Therefore, interface with Smart Contracts Management Engine.
- Demand flexibility management engine:
    - Aggregators will be capable of accessing the information about the flexibility available in the building connected via the PHOENIX platform. That information accessible via an API will come on the form of disaggregated flexibility or overall flexibility reports. The API will also allow triggering for events using USEF-like messages.
- Self-consumption optimization engine:
    - ESCOs will be capable of accessing the information about the results of energy optimization process available in the building connected via the PHOENIX platform in order to facilitate the execution of the business scenarios examined in the project.
- Grid Centric Services Analytics:
    - Information such as energy generation forecasting, or demand prediction of buildings could be of value for other stakeholders. They will be able to access to that via a user interface on the form of an API.

We presented above a set of dependencies with the different grid related components of PHOENIX platform. A more detailed specification of these interfaces will be made available as part of the work in WP6.

5 PHOENIX Process View

Following the detailed presentation of the system components that consist of the PHOENIX solution, the presentation of the dynamic view of the system is reported in this section. The analysis is taking into account the list of use cases as defined in D2.1 (presented in Table 18) along with the definition of the component interfaces as presented in previous section, setting a clear mapping of PHOENIX business objectives with technical implementations.

UC ID	UC Description
1	Adapt & Play integration of domestic appliances, legacy equipment and building systems
2	Building knowledge enhancement to upgrade the smartness of buildings
3	Services for building occupants to maximize their energy efficiency and increase overall building performance
4	Provision of comfort, convenience and wellbeing services to building occupants
5	Portfolio flexibility analysis and configuration to optimize grid operation
6	Flexible billing services and smart contracts for the retailer customers
7	Advanced energy services to promote self-consumption optimization

Table 18 List of PHOENIX use cases

The representation of sequence flow diagrams is delivered by using UML notation. We have to point out that the analysis remains at the specifications level, the technical details about the implementation of the interfaces will be reported in WP3 - WP6 as part of the documentation of the software development process.

5.1 Adapt & Play integration of domestic appliances, legacy equipment and building systems

The optimal management of energy systems in the building environment requires that various types of smart systems and components to be installed and the integration of these technologies

and building communications solutions. Therefore, a key use case is the establishment of a coordination framework that will ensure integration of different types of legacy equipment and technology building systems with diverse communication technologies and heterogeneous protocols while also ensuring the prompt management of the information retrieved from these systems. Towards this direction, we are considering three different layers of integration with physical assets and external data services, namely: PHOENIX External Data Source Adapter, PHOENIX Building System Adapter, PHOENIX IoT system Adapter. Data coming from the heterogeneous sources are made available to the PHOENIX Real-Time Data Broker and then stored to the central repository of the project, the PHOENIX Platform Data Repository. As stated in the following schema (Figure 19) two alternatives means of integration are supported in order to address both the request-response and pub-sub paradigms as presented also in the functional definition of the PHOENIX Real-Time Data Broker.

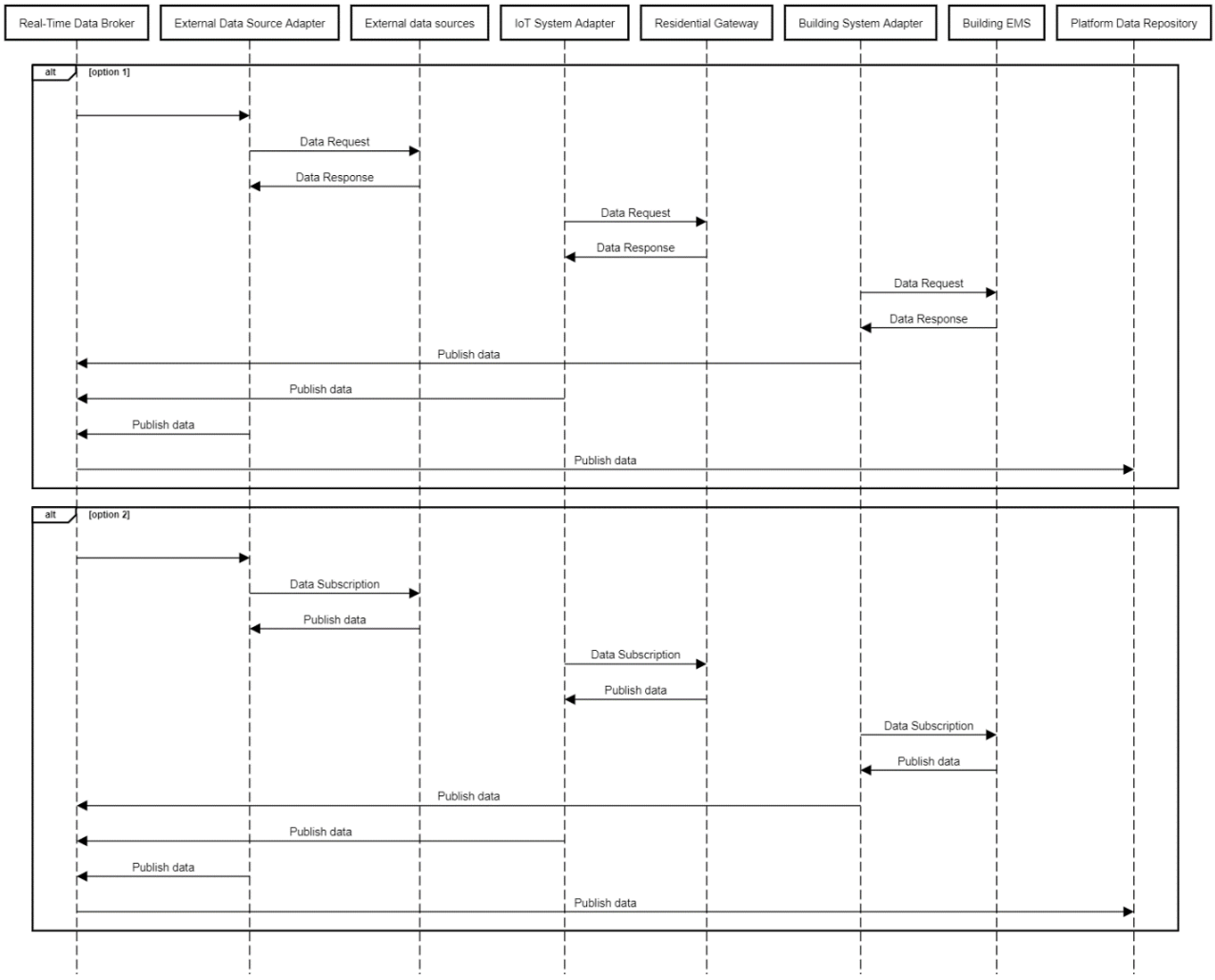

Figure 19 PHOENIX UC01- Process View

5.2 Building knowledge enhancement to upgrade the smartness of buildings

The data available from the sensor and building interfaces should be further consumed by intelligent data management and analysis techniques in order to extract knowledge from the raw the data - key information patterns within a multidimensional domain – which will further allow the realization of the smart buildings concept. Other data sources, such as building models, user profiles and external third-party data also play a key role to enable the provision of added value services in PHOENIX. As specified above, there are three different analytics components that consist of the knowledge extraction layer of PHOENIX framework, namely the AI-based Knowledge Engine, the User - centric services Analytics Engine and the Grid - centric services Analytics Engine which consume data from the PHOENIX Real-Time Data Broker and the PHOENIX Platform Data Repository components. The knowledge extracted from the different analytics layer will be further exploited by the business applications of the project in the following use cases. The different steps of the process towards the establishment of the enhanced knowledge layer of the PHOENIX platform are presented in the following figure (Figure 20).

Note: As specified also in the component's documentation, the AI-based Knowledge Engine may be defined also as a replicate data repository of the project where data may be retrieved by applying linked data base techniques. This alternative is also presented in Figure 20.

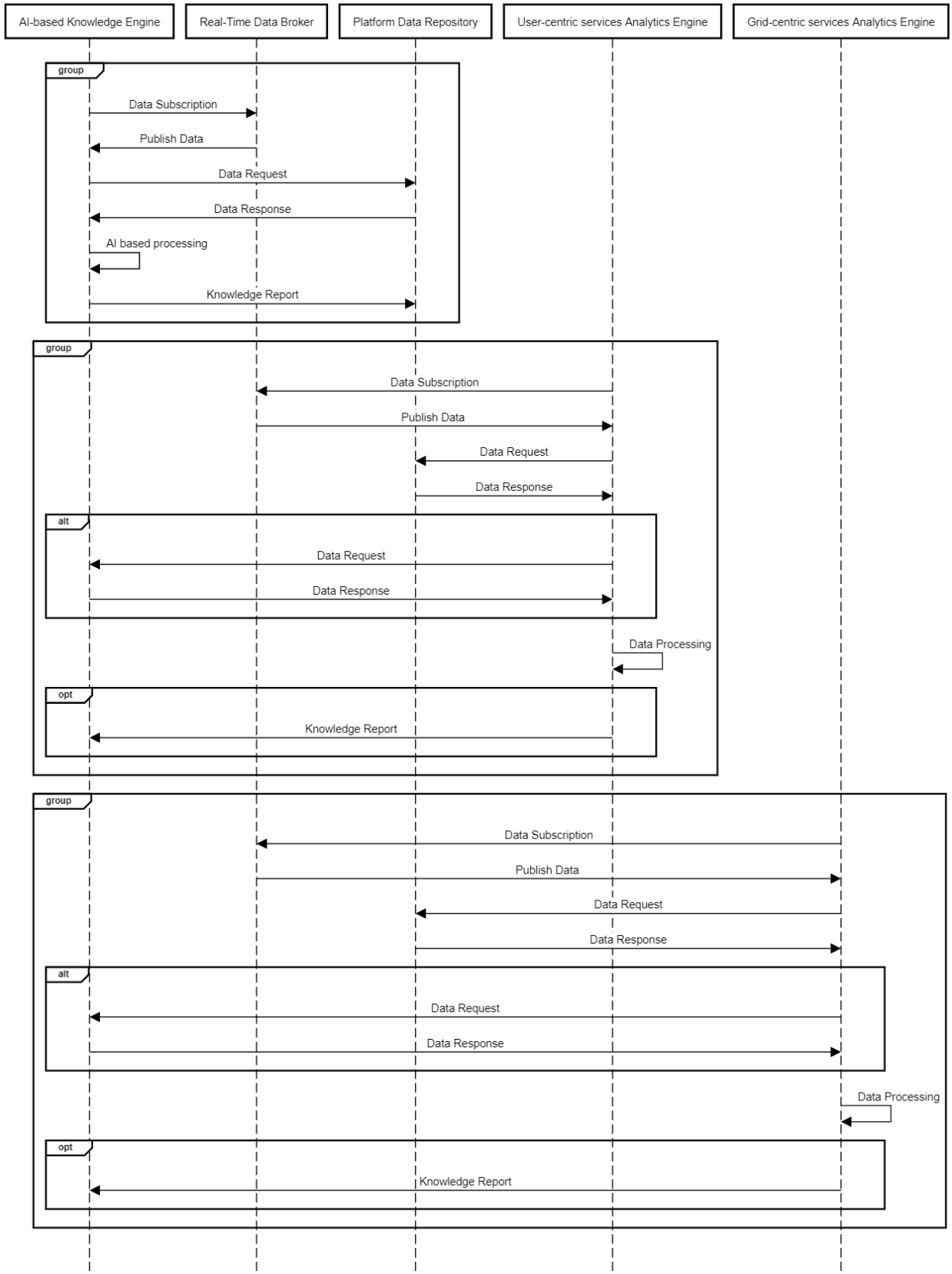

Figure 20 PHOENIX UC02- Process View

The aforementioned UC-01 and UC-02 focus on data and knowledge extraction as the information required to support the different business applications to be delivered in the project; UC-01 and

UC-02 flows are prerequisite for the business-related sequence flows (UC-03 - UC-07 to be presented. *Note: Any business functionality to be presented relies on the availability of real time and historical information and thus interfaces with the PHOENIX Real-Time Data Broker and the PHOENIX Platform Data Repository (data retrieval part) are not explicitly depicted in the following figures (to avoid repetition of the same information)*

5.3 Services for building occupants to maximize their energy efficiency and increase overall building performance

In order to maximize building efficiency and increase energy building performance, appropriate tools and services should be provided to allow end users accessing real time and easy to digest information about their energy behaviour; as a means to increase their engagement about building performance and subsequently get motivated towards actions that will increase energy (and cost) savings. Information as extracted from descriptive analytics at the building environment as well as information related to SRI/EPC performance (SRI/ EPC Evaluation Engine) should be available. In addition, information about the maintenance support and activities (Predictive Maintenance Engine) should be also accessible via the visualization dashboard (Building Occupants Visualization Dashboard).

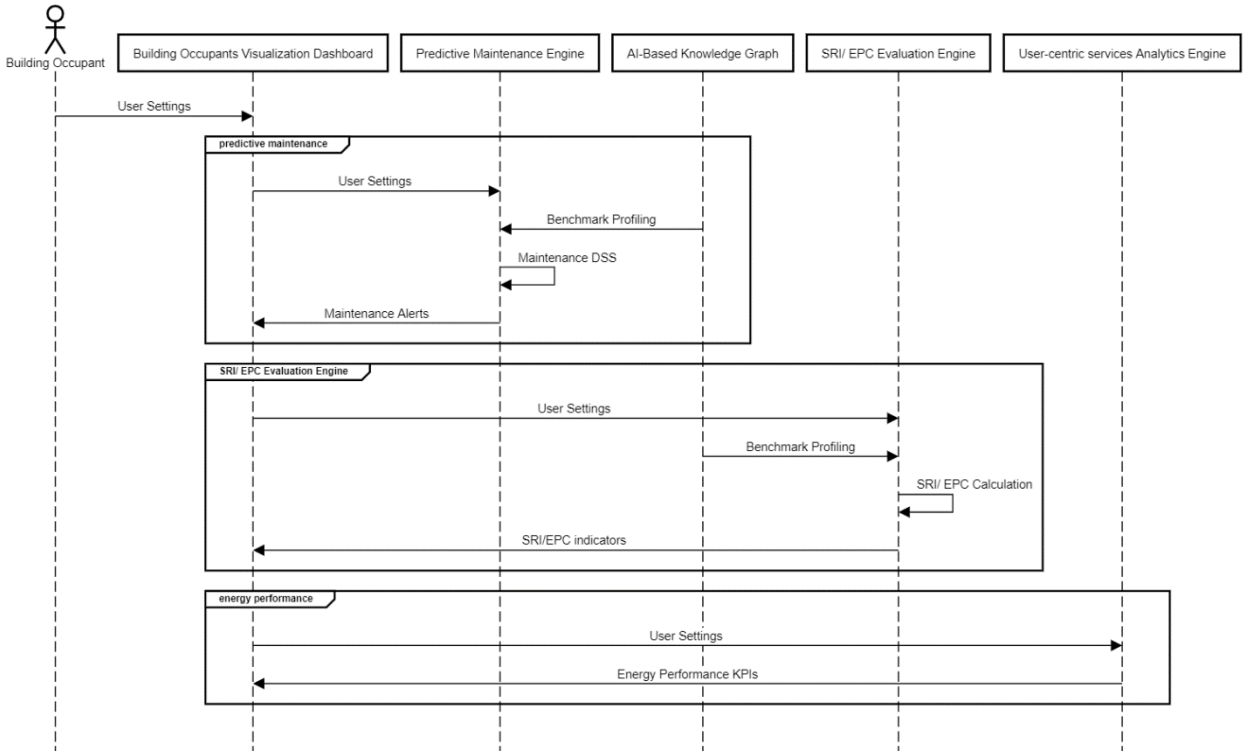

Figure 21 PHOENIX UC03- Process View

5.4 Provision of comfort, convenience and wellbeing services to building occupants

Apart from the obvious needs for reducing their energy and the associated costs, the end customers are also of interest about solutions that will fully preserve their daily preferences and comfort needs, improve their well-being and ensure indoor hygienic conditions. Any building smartization attempt shall take into account not only energy performance but also comfort and health aspects properly balancing the typically conflicting energy and well-being requirements.

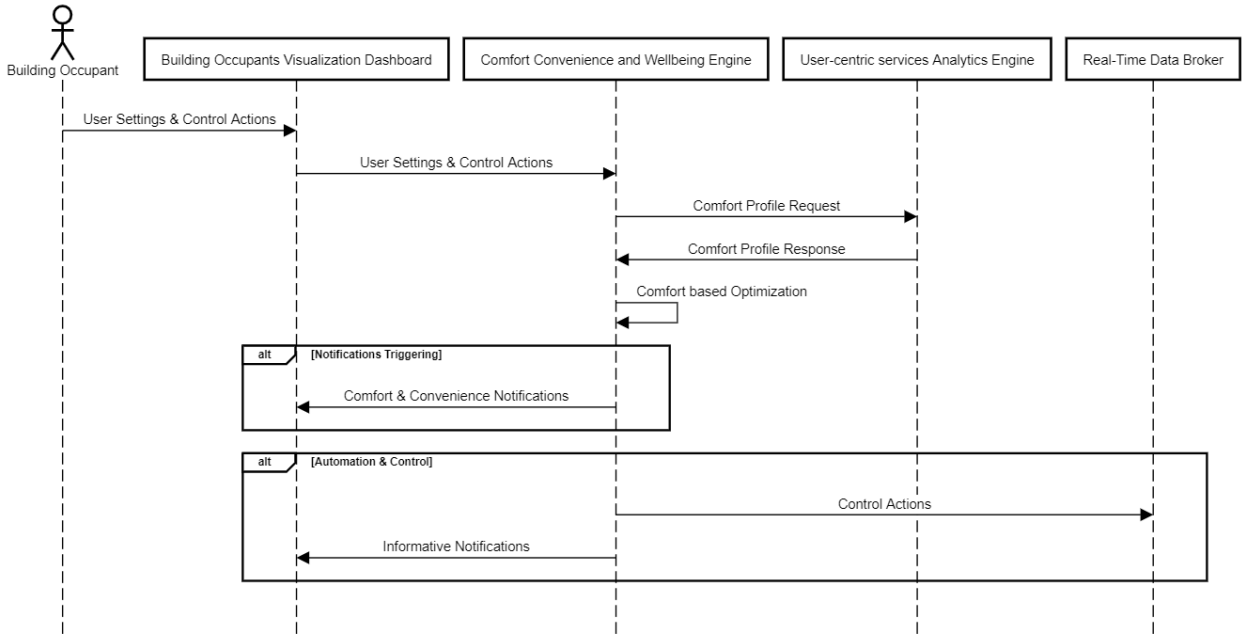

Figure 22 PHOENIX UC04- Process View

5.5 Portfolio flexibility analysis and configuration to optimize grid operation

The active participation of buildings on the provision of flexibility related services to network operators in a robust and reliable manner is considered is a key priority towards increasing building smartness and the reason why buildings should not just be digitalized, but also become active systems, enabled by the smart solutions present in buildings which have flexible consumption patterns. Therefore, appropriate tools and services needs to be developed to fully understand the building dynamics and the flexibility characteristics of their energy systems on the way to address evolving requirements from network operators about the provision of ancillary services from buildings.

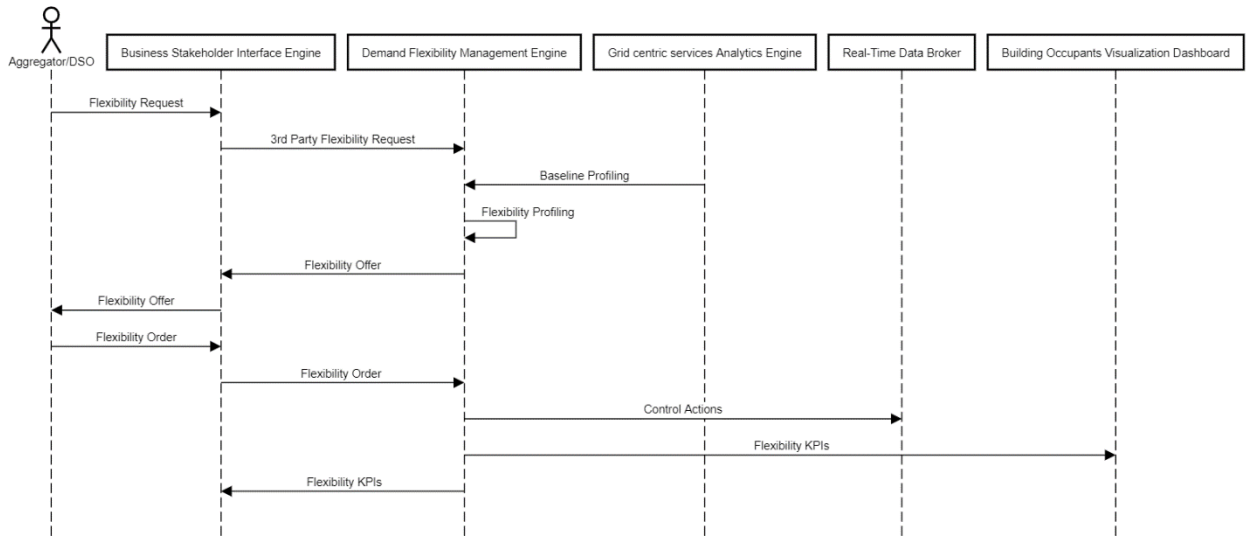

Figure 23 PHOENIX UC05- Process View

5.6 Flexible billing services and smart contracts for the retailer customers

In the deregulated market environment, traditional retailers need to transform themselves into smart energy service providers offering different types of services to their customers. Towards this direction, the definition and deployment of smart services (considering also contracts management and offer promotions definition) is a key requirement for the promotion of building smartization and customers participation in innovative business models.

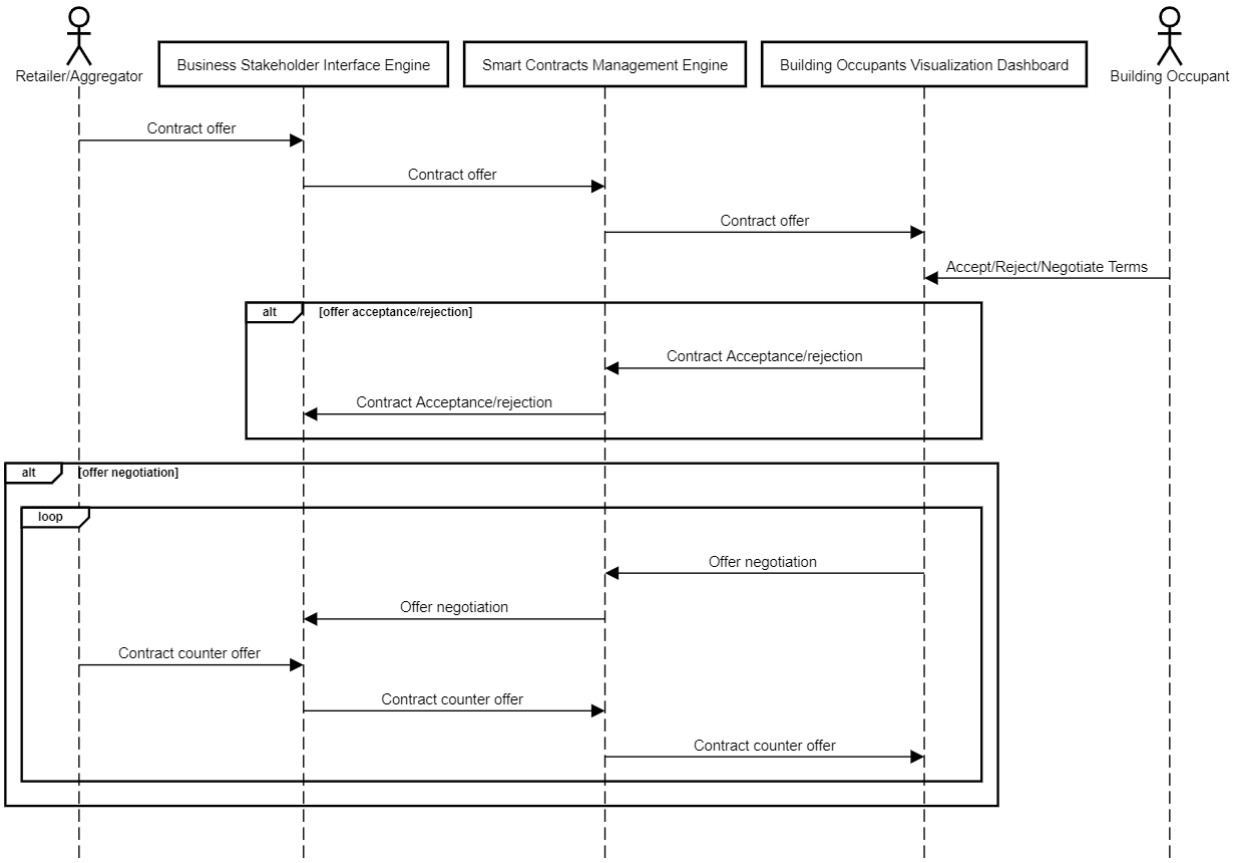

Figure 24 PHOENIX UC06 - Process View

5.7 Advanced energy services to promote self-consumption optimization

The proliferation of micro-generation renewable energies, the availability of domestic storage and the popularization of Electric Vehicles (also potentially used as storage) can be an opportunity for the energy stakeholders to gain a certain level of flexibility and further offer added value services to their customers. More specifically, intelligent support tools should be delivered to the electricity actors to properly design flexible control strategies and interacting mechanism with the dynamic entities of the portfolio network (batteries, EVs, buildings) in order to be able to maximize the level of self-consumption (contributing also to peak demand saving) of their customers and therefore the reduction of electricity costs for both parties.

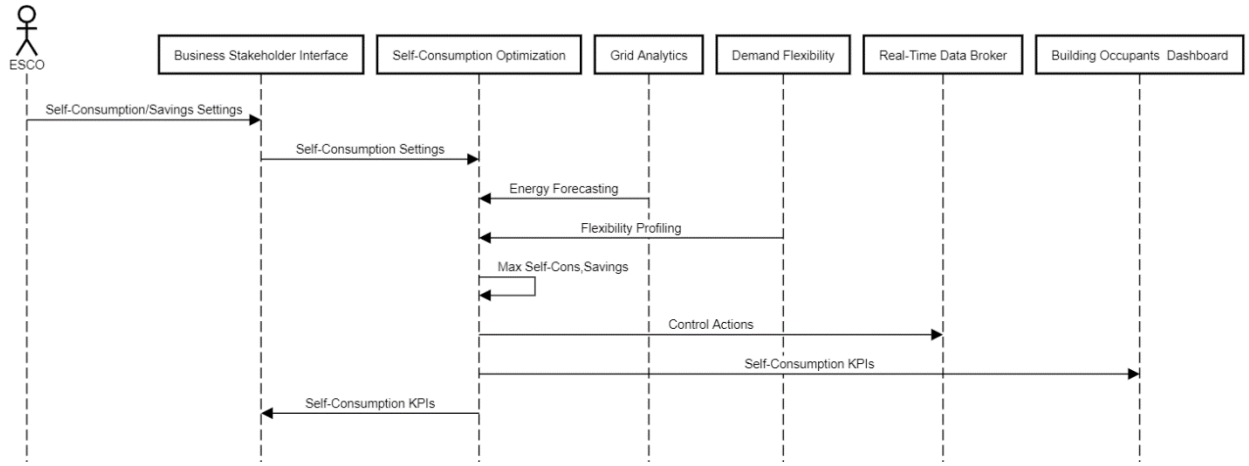

Figure 25 PHOENIX UC07 - Process View

The definition of the process view provides an overview of the different information exchanges among the different system components as well as a clearer view about the components to contribute at the fulfilment of the PHOENIX use cases as defined earlier in the project. In the following section, the deployment and deployment view of the PHOENIX solution is provided as part of the architecture definition.

6 PHOENIX Development View

In this section, the preliminary development view of the PHOENIX system is provided, covering aspects such as: (a) the use of existing software (along with corresponding IPR issues), (b) the technical dependencies of the different system components, as well as (c) the programming languages/technologies that may be used for the development of each system component.

More specifically, following the definition of the core modules that consist of the PHOENIX system (along with the associated functionalities), each partner (responsible for the development of the respective module) is also defining the framework to be considered for the development of its components taking also into consideration interoperability issues about the communications and interactions with other PHOENIX components.

Starting with the list of pre-existing technologies, in the following table we present the list of already available software developments to be considered also in the project.

PHOENIX Software Module	Existing Software Module	Responsible Partner	Description
PHOENIX Platform Data Repository	IoT platform for cross domain data management	ODINS	Interoperable Smart Building Platform for context-awareness data management and by-design security and privacy mechanisms
User Centric Analytics Services	Streaming Statistics and Analytics Framework	UBITECH	A software module to facilitate the extraction of real time analytics over the raw data available from the context broker
User Centric Analytics Services	MAESTRO Multi-cloud Orchestrator	UBITECH	This software will be the backbone of the different applications delivered by UBITECH in the project to facilitate any information exchange
User Centric Analytics Services	Big Data Management and Analytics Workbench	UBITECH	This software will be the backbone of the different applications delivered by UBITECH in the project to act as the local repository and the backbone for the

			different analytics services delivered in the project
User Centric Analytics Services	PART/PART2 CI/CD Framework	UBITECH	A CI/CD framework for Kubernetes platforms that provides a standard cloud-native CI/CD experience with containers. To facilitate the deployment of the different services of UBITECH

Table 19 PHOENIX Software Development – Pre-existing solutions

By providing the details about the use of existing software, the next step of the work is the short description of the development environment to be considered (by each responsible partner) for the delivery of each software module of the solution. This information is provided in the following table.

Software Component	**Development Framework**
PHOENIX External Data Source Adapter	Python 3.6 Node.js Node red flows/MQTT broker Java OpenJDK 1.8.200+
PHOENIX Building System Adapter	Python 3.6 Node.js / Node red flows Java OpenJDK 1.8.200+
PHOENIX IoT system Adapter	Python 3.6 Node.js Open-zwave library (Z-wave connectivity) Node red flows/MQTT broker
PHOENIX Real-Time Data Broker	Python 3.6, Java OpenJDK 1.8.200+ Orion Context Broker (Fiware)
PHOENIX Platform Data Repository	Python 3.6, Java OpenJDK 1.8.200+ TIAMAT is the technology used for handling the history of data retrieved from the different components of the project. It follows the NGSI-LD principles and is based on MongoDB though it supports connectivity with other repositories (Fiware)

AI-based Knowledge Engine	Triple store DB (Virtuoso or equivalent) Python 3.6, Java OpenJDK 1.8.200+
User-centric services Analytics Engine	In the backend layer, the Django web framework is used. In the internal data storage layer: PostgreSQL is used as the relational database for the user account and user profile relevant information. Elasticsearch is used as the internal storage of sensor and building information entities relevant to the user analytics.
Grid- centric services Analytics Engine	Python 3.6 Node.js DEEP as a Service (for analytics)
Comfort, Convenience and Wellbeing Engine	In order to provide its intended functionalities, the Comfort, Convenience and Wellbeing Engine will build on state-of-the art technologies, namely: (a) in the back-end layer, the Nest (NodeJS) web framework as a mature framework for delivering efficient, reliable and scalable server-side applications; (b) in the internal data storage layer, PostgreSQL (as the relational database for the management of the parameters of the application).
Predictive Maintenance Engine	Python 3.6 Node.js
SRI/ EPC Evaluation Engine	Python 3.6 Node.js
Building Occupants Visualization Dashboard	In the frontend layer, Angular framework is used. In the backend layer, the Django web framework is used. In the internal data storage layer: PostgreSQL is used as the relational database for the user account and profile relevant information. Elasticsearch is used as the internal storage of sensor, user and building information entities relevant to information needed by all engines that interact with the Dashboard.

Smart Contracts Management Engine	Python 3.6 Node.js / Node red flows USEF Reference Implementation
Demand Flexibility Management Engine	Python 3.6 Node.js / Node red flows
Self-consumption Optimization Engine	In the backend layer, the Django web framework is used. In the internal data storage layer: Elasticsearch is used as the internal storage of sensor and building information entities needed to address the self-consumption optimization problem.
Business Stakeholder Interface Engine	In the frontend layer, Angular framework is used. In the backend layer, the Django web framework is used while also additional frameworks will be incorporated to support connectivity with other system components. PostgreSQL is used as the relational database for the user account and profile relevant information. Elasticsearch is used as the internal storage for the grid related information to be reported via the user interface/GUI

Table 20 PHOENIX Software Development Specifications

Different system components consist of the overall PHOENIX platform. As heterogeneous development processes and framework are utilized by the corresponding partners, a "web services" based integration approach is considered. The partners will develop their individual components by using existing or new development modules, though the integration of these will be ensured by the definition of a common data model and common agreement on system interfaces. The definition of a common data model is required at the early phase of the project. Related to the tis, the specifications of the PHOENIX common information are provided as a dedicated task, part of the work in D4.1: PHOENIX Smartness Hub implementation - Initial Version incorporating the work performed as part of Task 4.2 Integrated Data Models, Knowledge Graphs and Automatic Semantics. In addition, the detailed interface specifications will be reported as part of the development and integration activities in WP3 - WP7.

Disclaimer: The analysis reported in this section is considered as the initial version of the development view while the updated and final version will be defined during the development phase of the respective components (WP3 - WP7).

7 PHOENIX Deployment View

This section covers the deployment view of the PHOENIX architecture, which deals with the PHOENIX solution deployment at the different demo sites. This viewpoint is mainly oriented to the physical implementation, involving components, hardware, connections and other physical constraints. The analysis of the deployment view is divided in two parts:

(a) the deployment of the IoT agents responsible for interconnection with the physical systems along with data gathering and

(b) the deployment of the PHOENIX ICT software solution as a whole.

Starting with the former, the deployment of the different IoT gateways will be specific per demo site. The details of this deployment were presented in D3.1 but a brief overview is provided in this section focusing mainly on network requirements and run-time needs which are further analysed per demo site:

The UMU pilot needs to be considered as a special case where the connectivity effort comes inherited from the PoC. In this case scenario, integration with the different types of external data sources is considered:

- PHOENIX External Data Source Adapter to retrieve weather data from external data sources. A middleware agent is being developed to enable the access to the APIs of the data source and send this information to the PHOENIX hub using a MQTT/SSL/ NGSI-LD interface. This agent runs on docker and its software requirements are 1 CPU core, 1 GB of RAM and 100MB of HD space.

- PHOENIX Building System Adapter to integrate with the BMS in UMU premises which is based on IoT platform " Pleiades Building". Two software modules have been deployed to track data from the BMS system. This adapter is composed of several docker containers which require 2 CPU cores, 2GB of RAM and 100MB of free HD space.

- PHOENIX IoT Adapter to integrate with 3rd party sensors or other systems available in several zones of the UMU site. In this case, the PHOENIX IoT Adapter is running in the local/demo environment through the deployment of several hardware components that are able to run this software implementation. As specified in D3.1 the pilots are going to use a Raspberry Pi device (Raspberry Pi 3B+) as the IoT Gateway along with the USB Z-Stick Gen5 to ensure Z-Wave Integration of sensors and actuators. In case a MODBUS device is available, a software agent is also deployed at the local hardware (along with the

appropriate hardware component in case of RS-485 connection) to track communication via MODBUS.

By defining the hardware details for the PoC, the deployment at the actual demo sites is provided. In the case of Miwenergía, the installation of PHOENIX IoT system Adapter running in Raspberry Pi devices will be considered. In total, 23 Raspberry Pi devices will be installed in one office building and four dwellings. In addition, the proper network configurations will be performed per gateway, following the instructions provided in D3.1 to ensure connectivity with the PHOENIX platform. Also, the availability of weather data from the implementation performed at the PoC will be considered.

In the case of KaMa, the installation of PHOENIX IoT system Adapter running in Raspberry Pi devices will be considered. In total, one Raspberry Pi device will be installed in one building. In addition, the proper network configurations will be performed per gateway, following the instructions provided in D3.1 to ensure connectivity with the PHOENIX platform. Also, the availability of weather data from the implementation performed at the PoC will be considered.

In the case of LTU, the connection of the sensors and actuators are already in place and connected to a platform managed by LTU (out of the scope of the project). Therefore, the aim is to develop a software module (as an IoT agent) that will transfer the data from the existing platform to the PHOENIX platform. This software will run locally at LTU. Also, the availability of weather data from the implementation performed at the PoC will be considered.

The ARDEN pilot is formed by two different sites. In the case of the commercial building, the sensors and actuators are connected to a Building Management System that allows control of services. For the connection of this BMS to the PHOENIX platform, ARDEN is going to implement a middleware that is capable to connect the Delta Controls BMS to the PHOENIX platform, with this, it will be possible to connect to the sensors and actuators already existing on the commercial building. This software (Building System Adapter) will run in the cloud with required connectivity and capacity and will send this information to the PHOENIX hub using a MQTT interface. With respect to the two domestic sites, the ARDEN pilot will install new hardware that will be selected with the purpose of easing the connectivity to the PHOENIX platform. The MyEnergi home portal solution will be used at both sites (one site already has a MyEnergi hub and one will be installed at the other site) and their manufacturers have been contacted to verify if they offer connectivity options. An API has been provided by the manufacturer to enable the connection of the hubs with the PHOENIX platform via middleware being developed to run on the cloud and to send data to the PHOENIX platform using a MQTT

interface. Also, the availability of weather data from the implementation performed at the PoC will be considered.

Towards the deployment of the PHOENIX ICT software platform, as stated in previous section the overall solution will be developed and deployed following a microservices based approach. The different software bundles are deployed as micro services which are interconnected via the internet. Towards this direction, the focus in this section is about the early characterization of the hardware needs for the different software modules.

PHOENIX Software Component	Hardware Requirements
PHOENIX Real-Time Data Broker	8th Generation or newer Intel i7 (quad-core equivalent VM), 8 GB RAM, 512 GB SSD Ubuntu Server 20.04 LTS on server or Debian 11. Docker image will be created to support deployment
PHOENIX Platform Data Repository	8th Generation or newer Intel i7 (quad-core equivalent VM), 8 GB RAM, 512 GB SSD Ubuntu Server 20.04 LTS on server or Debian 11 Docker image will be created to support deployment
AI-based Knowledge Engine	8th Generation or newer Intel i7, 32 GB RAM, 512 GB SSD Ubuntu Server 20.04 LTS on server Docker image will be created to support deployment
User-centric services Analytics Engine	Apache Software Foundation CloudStack VM, 16 GB RAM, 350 GB, 4 v-CPUs Centos-7 on Server The engine will be part of a multi-node Kubernetes cluster.
Grid- centric services Analytics Engine	8th Generation or newer Intel i7 (quad-core equivalent VM), 4 GB RAM, 64 GB SSD Ubuntu Server 20.04 LTS on server or Debian 11. Docker image will be created to support deployment
Comfort, Convenience and Wellbeing Engine	Intel i7 (4 core ore more), 8 GB RAM, 160 GB SSD Ubuntu Server 20.04 LTS on server Docker image will be created to support deployment
Predictive Maintenance Engine	8th Generation or newer Intel i7 (quad-core equivalent VM), 4 GB RAM, 64 GB SSD Ubuntu Server 20.04 LTS on server or Debian 11. Docker image will be created to support deployment
SRI/ EPC Evaluation Engine	8th Generation or newer Intel i7 (quad-core equivalent VM), 4 GB RAM, 64 GB SSD Ubuntu Server 20.04 LTS on server or Debian 11. Docker image will be created to support deployment
Smart Contracts Management Engine	8th Generation or newer Intel i7 (quad-core equivalent VM), 4 GB RAM, 64 GB SSD Ubuntu Server 20.04 LTS on server or Debian 11. Docker image will be created to support deployment

Demand Flexibility Management Engine	8th Generation or newer Intel i7 (quad-core equivalent VM), 4 GB RAM, 64 GB SSD Ubuntu Server 20.04 LTS on server or Debian 11. Docker image will be created to support deployment
Self-consumption Optimization Engine	Apache Software Foundation CloudStack VM, 16 GB RAM, 350 GB, 4 v-CPUs Centos-7 on Server The engine will be part of a multi-node Kubernetes cluster.
Building Occupants Visualization Dashboard	Apache Software Foundation CloudStack VM, 16 GB RAM, 350 GB, 4 v-CPUs Centos-7 on Server The dashboard will be part of a multi-node Kubernetes cluster.
Business Stakeholder Interface	Apache Software Foundation CloudStack VM, 16 GB RAM, 350 GB, 4 v-CPUs Centos-7 on Server The dashboard will be part of a multi-node Kubernetes cluster.

Table 21 PHOENIX Software Deployment Specifications

The focus of this section is to specify the deployment characteristics as well as specific packages considered for the deployment of the different components. The aforementioned microservices based approach supports deployment either in a distributed way or centralized to deployment a central infrastructure taking into account the requirements of the business partners of the project. A more detailed presentation of the actual deployment to be performed in the PHOENIX project (at the different demo sites) will be reported as part of the work in Task 5.4 Integration, Testing and Refinement and T6.4: Integration, Testing and Refinement respectively.

8 Summary

The scope of the document is to present the architectural design of the PHOENIX system along with the corresponding technical specifications. At first, the methodological framework for the delivery of the system architecture was defined. Then, the PHOENIX conceptual architecture was specified as well as the logical, process, development and deployment views of the system. More specifically:

- The PHOENIX conceptual architecture provides the system overview, highlighting the different layers and components that consist of the overall solution
- The PHOENIX logical view provides the functional details of each of the component that are part of the PHOENIX solution. The study incorporates the technical requirements specification, data inputs and outputs, interrelation between system entities towards the definition of the common semantics among the components to further proceed with the development of interfaces during the integration phase of the project.
- The PHOENIX process view provides the dynamic interactions among the component that are part of the PHOENIX solution on the way to address project specific use cases.
- The PHOENIX development view provides the developments details of each of the component that are part of the PHOENIX solution. The study incorporates the different software frameworks and approaches to be considered at the development phase of the project.
- The PHOENIX deployment view provides the deployment details of the PHOENIX solution.

The architectural views and viewpoints offered in this report will further push the design and implementations during the project's period. The diagrams demonstrate the main elements of the PHOENIX system, the core functionalities resulting as well as the way they interrelate with the system.

The PHOENIX architecture document delivers a sound basis for the technical developments of the system that will take place in WP3, WP4, WP5 and WP6 and will act as a reference document during the deployment and demonstration period (WP7). Although no major alterations are anticipated to the overall system architecture and its major elements, this report can be considered as a living document that will address minor improvements that might be essential in case of new unforeseen restrictions that will arise during the implementation phase.

9 References

[1]. Description of Actions, PHOENIX Project
[2]. PHOENIX D2.1 - Business, market & regulatory requirements
[3]. PHOENIX D2.2 - Social barriers and enablers, building use cases definition and requirements
[4]. Kruchten, Philippe. (1995). The 4+1 View Model of Architecture. IEEE Software. 12. 45-50. 10.1109/52.469759.

10 Abbreviations List

Abbreviation	Description
DoA	Description of Action
DSOs	Distributed System Operator
BMS	Building Management System
IPR	Intellectual Property Rights
IoT	Internet of Things
ISO	International Standards Organization
DHW	Domestic Hot Water
HVAC	Heating Ventilation Air Condition
EMS	Energy Management Solution
BMS	Building Management Solution
EaaS	Energy as A Service
ESCOs	Energy Service Company
KPI	Key Performance Indicator
PIR	Passive Infrared
DR/RDR	Residential Demand Response
VOC	Volatile Organic Compounds
DSS	Decision Support System
CO_2	Carbon Dioxide
CO	Carbon Monoxide
PM	Particle Material
RES	Renewable Energy Source
CB	Context Broker
KG	Knowledge Graph
GW	Gateway
IAQ	Indoor Air Quality
EPC	Energy Performance Certificate
API	Application Programming Interface
NGSI-LD	Next Generation Service Interfaces Linked Data
ML	Machine Learning
UI	User Interface

ICT	Information Communication Technologies
USEF	Universal Smart Energy Framework
GDPR	General Data Protection Regulation
SRI	Smart Readiness Indicator
EV	Electric Vehicle
PV	Photovoltaic
UC	Use Case
PoC	Proof of concept

11 Annex I External data source adapter

External data sources are most usually available for integration via APIs which appear to have become the standard means of providing data for interoperability and integration with third party applications.

An example is the electricity market data provided via API by the Irish market operator – SEMOpx. SEMOpx is a company that provides intraday and day-ahead electricity market trading information for Ireland and Northern Ireland. The information of importance for this manual is the price in Euro per Mega Watt hour for the "Day Ahead" energy market that is uploaded daily within a CSV file.

The daily csv file can be accessed with a simple http request made to the SEMOpx web address as detailed following. Middleware has been developed to interrogate SEMOpx and to send the data to the Phoenix MQTT broker.

The API request (to search for a given file name) is composed of a base url followed by a series of specifications that are added to filter for the appropriate report.

They take the following format:

Base Url : https://reports.semopx.com/api/v1/documents/static-reports?

Search Fields :

Group = ["Market Data"*] : Report Group

DPuG_ID =["EA-001"*, "EA-002", "EA-003", … etc] : Report Spec

ReportName =["ETS Market Results"] : Report Type

ResourceName = ["MarketResults_SEM-DA_PWR-MCR"*] : Report Prefix Name

Date = >=YYYY-MM-DD<= YYYY-MM-DD : Date Range

The format of the http request is as follows;

Format: Base URL + Field1 + = + Value1 + & + Field2 + = + Value2 + & + …

https://reports.semopx.com/api/v1/documents/static-reports?Group=Market Data&DPuG_ID=EA-001&ReportName=ETS Market Results&ResourceName=MarketResult_SEM- DA_PWR-MRC& Date=>=2021-07-01<=2021-07-02 *

*An example of a request made for the price list csv filename on July 1st 2021

The request will return a json file with the file name listed as “Resource Name”;

Format : “ResourceName”:”THE_NAME_OFTHE_CSV_FILE.csv”

EG : "ResourceName":

“MarketResult_SEM-DA_PWR-MRC-D+1_20210630100000_20210630103605.csv”

N.B. if no csv file exists an empty json list will be returned

The timestamps and price values are listed between the start identifier "Index prices;60;EUR\r\n" and end identifier "\r\nIndex prices;60;GBP". Each row of the csv file is divided by the characters “\r\n” and each value in the row is separated by the “;” character. The data can be extracted using these bits of information to separate each row and their values.

An example of the data returned form the HTTP request between the start and end identifier is as follows;

2021-05-31T22:00:00Z;2021-05-31T23:00:00Z;2021-06-01T00:00:00Z;2021-06-01T01:00:00Z;2021-06-01T02:00:00Z;2021-06-01T03:00:00Z;2021-06-01T04:00:00Z;2021-06-01T05:00:00Z;2021-06-01T06:00:00Z;2021-06-01T07:00:00Z;2021-06-01T08:00:00Z;2021-06-01T09:00:00Z;2021-06-01T10:00:00Z;2021-06-01T11:00:00Z;2021-06-01T12:00:00Z;2021-06-01T13:00:00Z;2021-06-01T14:00:00Z;2021-06-01T15:00:00Z;2021-06-01T16:00:00Z;2021-06-01T17:00:00Z;2021-06-01T18:00:00Z;2021-06-01T19:00:00Z;2021-06-01T20:00:00Z;2021-06-01T21:00:00Z

\r\n
80,310;77,120;74,820;76,000;77,310;83,070;87,650;90,750;107,330;159,490;156,900;151,000;148,390;147,000;125,200;103,200;98,660;98,660;104,330;106,200;101,090;92,970;89,250;81,350

Keep in mind that the times are listed in GMT so need to be converted to Irish Standard Time. It is also worth noting that the only way to obtain data for a larger time period is to harvest information from each day's csv file within that time range. And finally, the energy prices are listed at hourly intervals.

Also, as integration of third-party data it was integrated the data from Spanish TSO Red Eléctrica Española, so data from the price of electricity on each hour of the day can be sent to the platform. In order to integrate hourly energy pricing for the two Spanish pilots, we have created a Node Red flow application which extracts data from the REE API, adapting this information to the NGSI-LD data-model and ontologies used in PHOENIX, in order to be introduced in the Context Broker. This flow is shown in the following figure for the example of PVPC (Precio de Venta de Pequeño Consumidor: selling price for the small consumer) that send the data hourly, and converts the information into the defined NGSI-LD data model, representing information in terms of entities, properties and relationships. After that adaptation, the information is created or updated in the Context Broker through its REST API.

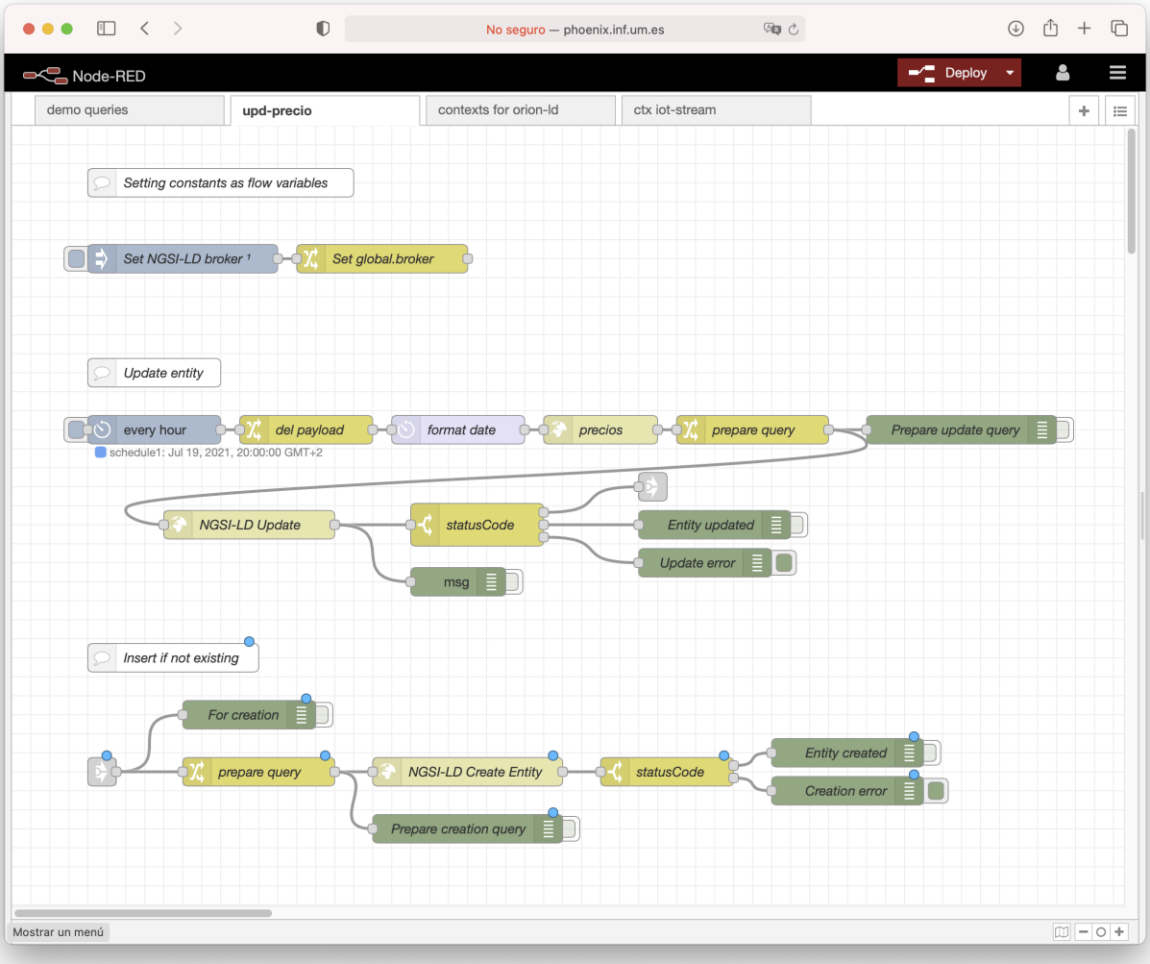

Figure 26 Node Red flow for REE hourly pricing updates

In addition to this, REE also offers data of the generation mix, and therefore the carbon intensity of the electricity The API of REE also offers the possibility of consulting in real time the mix of production for that given moment that opens the door to use AI to optimise not only the scheduling for reducing peaks in kilowatts, but also on reducing peaks in CO_2 emissions. An example of the output that one can get from the API of REE can be found on Figure 27.

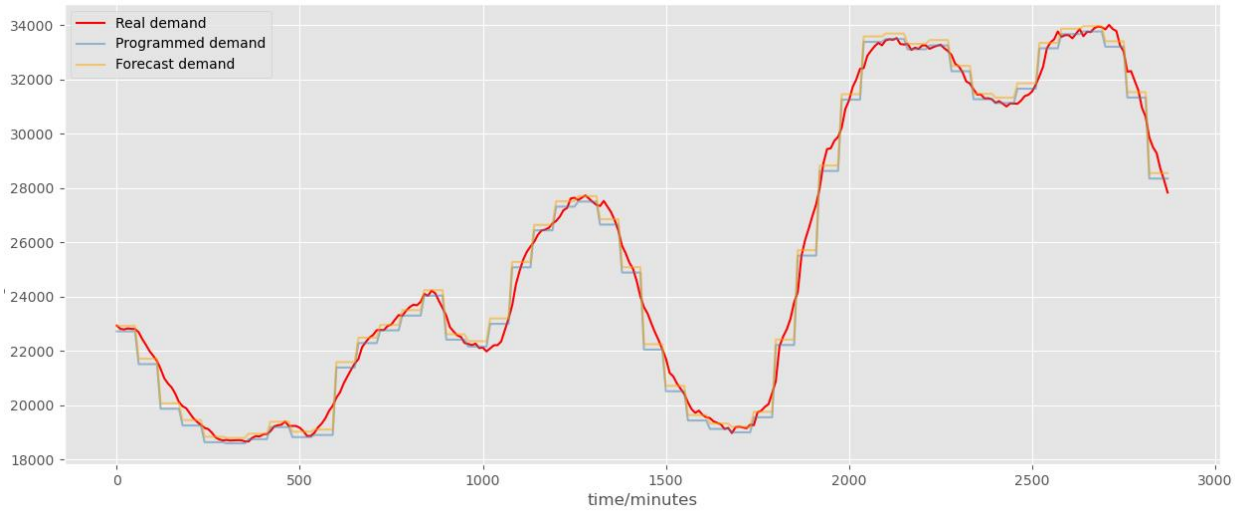

Figure 27 Integration of PHOENIX with Red Eléctrica Española data repo

ENTSO-E is the European association for the cooperation of transmission system operators (TSOs) for electricity. The mission of ENTSO-E is "*ensuring the security of the interconnected power system in all time frames at pan-European level and the optimal functioning and development of the European interconnected electricity markets*"[1].

As ENTSO-E has a European scope, the connection to it was seen strategic on PHOENIX as it allows to have data about generation of electricity for all pilot sites. For PHOENIX, ENTSO-E has been identified as an excellent source of data for developing the services for grid flexibility. The data apart from including generation mixes, it also offers the possibility of getting congestion signals what can be of great use for the design and testing of grid signals for demand flexibility program designs. Figure 28 shows an example of the data available on the ENTSOE platform.

[1] *https://www.entsoe.eu/about/inside-entsoe/objectives/*

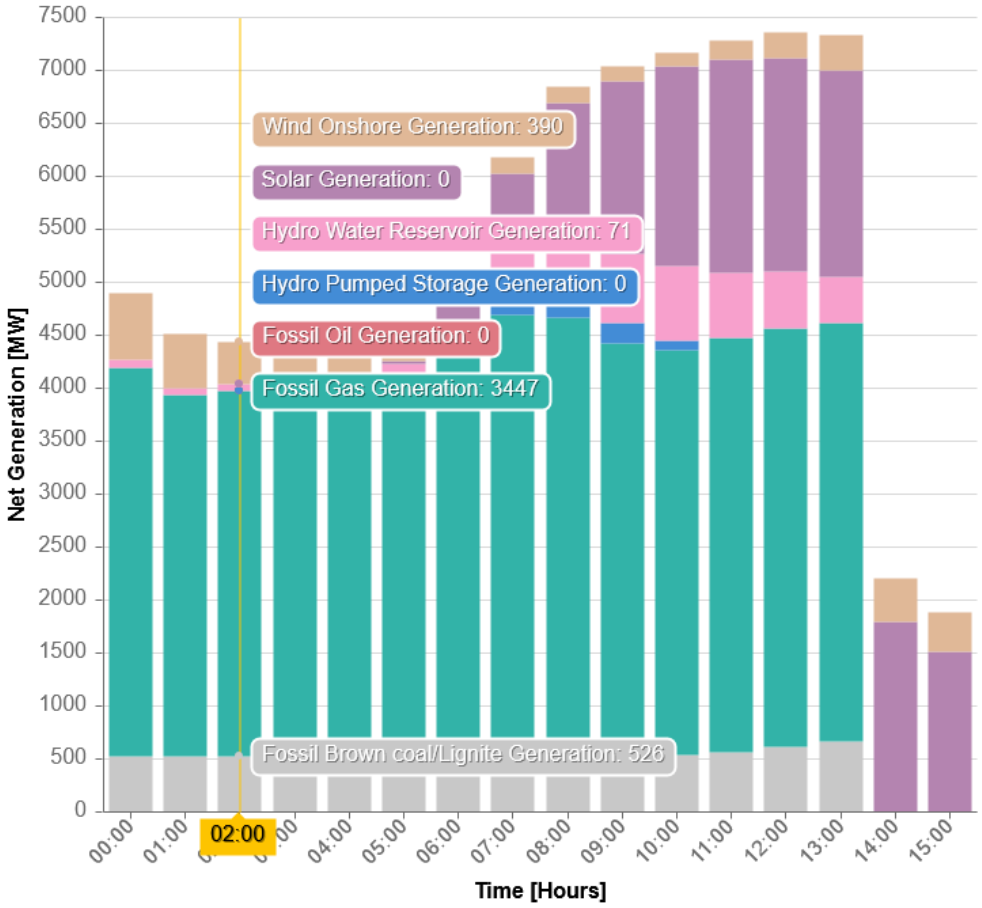

Figure 28 Example of data available on the ENTSO-E platform