Funded by
the European Union

HORIZON-CL5-2022-D4-02

EUROPEAN COMMISSION

European Climate, Infrastructure and Environment Executive Agency

Grant agreement no. 101123238

Smart Grid-Efficient Interactive Buildings

Deliverable D2.5

Data enabling pilot monitoring platforms

Project acronym	EVELIXIA
Full title	Smart Grid-Efficient Interactive Buildings
Grant agreement number	101123238
Topic identifier	HORIZON-CL5-2022-D4-02-04
Call	HORIZON-CL5-2022-D4-02
Funding scheme	HORIZON Innovation Actions
Project duration	48 months (1 October 2023 – 30 September 2027)
Coordinator	ETHNIKO KENTRO EREVNAS KAI TECHNOLOGIKIS ANAPTYXIS (CERTH)
Consortium partners	CERTH, RINA-C, CEA, CIRCE, UBE, HAEE, IESRD, UNIGE, SOLVUS, R2M, EI-JKU, FHB, EEE, EG, ÖE, PINK, TUCN, DEER, TN, ENTECH, SDEF, EGC, KB, AF, Sustain, NEOGRID, MPODOSAKEIO, DHCP, HEDNO, BER, MEISA, ITG, NTTDATA, TUAS, NEOY, HES-SO
Website	https://www.evelixia-project.eu/
Cordis	https://cordis.europa.eu/project/id/101123238

Disclaimer

Copyright Message

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon Europe Framework Programme for Research and Innovation under grant agreement no 101123238. ***Disclaimer:*** *The European Commission is not responsible for any use made of the information contained herein. The content does not necessarily reflect the opinion of the European Commission.*

Deliverable D2.5

Data enabling pilot monitoring platforms

Deliverable number	D2.5
Deliverable name	Data enabling pilot monitoring platforms
Lead beneficiary	Entech Smart Energies
Description	This deliverable is directly linked to the activities foreseen in Task 2.4, focusing on the development and integration of pilot platforms and APIs for local data collection across all pilot sites (PS) within EVELIXIA. This report is considered as the first version of D2.5
WP	2
Related task(s)	T2.4
Type	Report/Demonstrator
Dissemination level	Public
Delivery date	10.04.2025
Main author	Saleh Cheikh (Entech), Florian Wenig (FHB), Per Pedersen (NEOGRID)
Contributors	Mihaela Cretu (TUCN), Ioannis Zornatzis (CERTH), Ana LORENZO (NTTData), Hugo Huerta (TUAS)

Document history

Version	Date	Changes	Author
V1 – first draft	06.01.2025		Entech,
V1 – reviews	14.03.2025	Comments and requests of clarification on specific platform PS	CEA, Neogrid
V1 – consolidated version	21.03.2025		Entech
V2 – reviews	24.03.2025	Minor clarifications on Danish, and Austrian PS	Neogrid, Pink
V2– consolidated version	31.03.2025		Entech
Final version	01.04.2025		Entech
Final deliverable submission	10.04.2025		CERTH

EXECUTIVE SUMMARY

This deliverable, titled "Data enabling pilot monitoring platforms" and classified under Task 2.4 of Work Package 2 (EVELIXIA's Intelligent Technological Repository as Flexibility Enablers), focuses on the development and integration of pilot platforms and APIs for local data collection across all pilot sites (PS) within EVELIXIA. The goal is to establish secure and reliable bidirectional communication channels using EVELIXIA's southbound APIs and connectors. By developing and adapting field platform API connectors, each PS contributes to a unified infrastructure that facilitates seamless data flow and system integration. Moreover, three specific PS will initiate innovative developments of hypervisor platforms to manage unique operational and technological needs. These tasks align with the broader EVELIXIA objectives of advancing secure, reliable, and interoperable systems.

Objectives and Scope

The main objective of this deliverable is to establish secure and reliable bidirectional communication channels using EVELIXIA's southbound APIs and connectors for all pilot sites, while for three specific PS (Austrian, French and Danish), the objective is to develop innovative solutions to manage operational and technological needs in a secure system. In the context of EVELIXIA project, the operational needs consist of overseeing and managing the entire system operation efficiently, ensuring that the communication and control mechanisms can adapt to evolving requirements, providing reliable oversight of system performance, detecting faults, and optimizing operation dynamically, and ensuring seamless data exchange and communication between different systems and components. The technological needs could be developing a robust hypervisor that ensures system security, resilience and compliance with industry standards, developing APIs that facilitate seamless bidirectional communication while maintaining high levels of security and efficiency, and designing solutions that can maintain system availability in case of failures. The methodology includes collaborative development, iterative testing and cross-site standardization.

The three specific PS, that are responsible for developing secure hypervisor platform, reflecting their unique roles, are:

- The Austrian PS will develop controllers, optimization algorithms, and APIs to enable seamless external bidirectional communication of “enerboxx” with EVELIXIA’s middleware layer
- The French PS will develop a hypervisor hosted in a secure cloud environment to oversee the system operation
- The Danish PS will develop a secure cloud-based hypervisor solution to manage the entire system operation

Expected Results

- Deployment of secured APIs across all PS, ensuring a cohesive and interoperable system architecture
- Seamless integration of controllers and API at the Austrian PS to enable bidirectional communication with the middleware layer, advancing optimization algorithms
- Successful implementation of hypervisor solutions for the Danish and French PS, improving system oversight and operational reliability

Contents

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION AND OBJECTIVES

The evolution of digital technologies has boosted the importance of secure, reliable, scalable, and interoperable platforms for effective data management and system integration. Within the scope of EVELIXIA, each PS has a major role in advancing these capabilities through the development of specialized platforms and APIs.

This deliverable focuses on establishing the foundation for seamless data collection, processing, and exchange across all PS. The objectives of this deliverable are developing secure communication channels to ensure reliable data flow between PS and EVELIXIA's middleware layer. Also, three specific PS platforms are responsible of developing innovation hypervisor solutions secured to monitor and oversee the whole operation of the PS.

1.1 Scope and objectives

The deliverable 2.5 is the result of the work done in the tasks 2.4 which is a part of the **WP2 EVELIXIA's Intelligent Technological Repository as flexibility enablers.** It focuses on the development and integration of pilot platforms and APIs for local data collection across all PS within EVELIXIA. Moreover, three specifics PS (Austrian, French and Danish), will initiate innovative developments of hypervisor platforms to manage unique operational and technological needs.

1.2 Structure

This report is divided into several key sections. It begins with an introduction outlining the scope, objectives, followed by a detailed description of the three specific PS (Austrian, French, and Danish). Then, a section dedicated for the development of APIs for all PS. The final sections provide conclusions, lessons learned, and future action plans. The annexes contain detailed technical specifications, calculations, and regulatory documents for each solution.

1.3 Relation to Other Task and Deliverables

The deliverable 2.5 is the result of the work done in the task 2.4 which is a part of the **WP2 EVELIXIA's Intelligent Technological Repository as flexibility enablers.** This task is linked with the innovative solutions developed within WP2, and the interfaces will be connected to EVELIXIA's middleware layer by the Southbound API connector, developed within the work of T3.6 and described in deliverable D3.7, to allow the EVELIXIA partners a bidirectional data access for monitoring and control on the PS.

2 DEVELOPMENT OF PS SPECIFIC PLATFORM

Three specifics PS platforms (Austrian, French and Danish) are among the seven PS involved in this task 2.4 to develop innovative solutions. The three specific PS are responsible for developing secure hypervisor platforms. The Austrian PS will develop controllers, optimization algorithms, and APIs to enable seamless external bidirectional communication of “enerboxx” with EVELIXIA’s middleware layer. The French PS will develop a hypervisor hosted in a secure cloud environment to oversee the system operation. The Danish PS will develop a secure cloud-based hypervisor solution to manage the entire system operation

2.1 Austrian PS Specific Platform

The Austrian PS is located in the Southeast of Austria in the province Burgenland and utilize the existing flexibility sources to tackle the current challenges in the energy sector, improve electric power management, empower associations to actively participate in the market and act as a multifaceted pilot region.

2.1.1 Overview of the PS

Referencing to Deliverable D1.3, the Austrian pilot covers different private and public buildings, renewable energy plants and a district heating grid as well as a local power grid. For the Austrian PS, no specific energy management system exists. The PS provides the relevant asset infrastructure and within the EVELIXIA project, the communication interfaces for monitoring and control are set up and installed. The interfaces will be connected to EVELIXIA’s middleware layer by the Southbound API connector to allow the EVELIXIA partners a bidirectional data access for monitoring and control. The control signals itself needs to be provided by the innovative solution services participating in EVELIXIA. Two use cases have been defined:

- Thermal flexibility of district heating: Enhancing thermal flexibility in the district heating grid by decentralized thermal storage solutions at the end-user (district heat water (DHW) storage system “enerboxx”) and/or at critical network locations (puffer storages)
- Local optimization of electricity consumption: Flexible operation of heat pumps, electrical water heaters and/or possibly consuming assets at

individual consumer and prosumer level to meet requirements of implicit and explicit demand side flexibility incorporating local production and distribution system operator requirements.

For both use cases, the PS specific platform provides the communication interfaces between the innovative solution providers and the selected hardware in participating in the PS.

2.1.2 Development of the Austrian PS Specific Platform

Referencing to deliverable D5.3, the PS specific platform at the Austrian PS needs to handle datapoints from four different pilot specific APIs, representing the four hardware environments participating in the pilot case:

The **buffer storage and domestic hot water solutions** from Pink are equipped with local controllers from “Technische Alternative” (TA) connected to proprietary web gateways. TA provides a JSON-API which is used to access the actual values of the connected energy assets. Additionally, the TA web portal, build for manual remote maintenance, can be tricked to allow for an automated access on control variables of available actuators in the system.

The existing **district heating grid** at the PS is equipped with (historical) Danfoss district heating transfer stations and heat meters for billing processes. Energy consumption data and individual available operational data of the transfer stations as well as underlying building automation data is collected once per day and send to the InfluxDB. Furthermore, operational data from the district heating central is additional provided in the database. No actuators are available in the district heating grid.

The Shelly **smart home ecosystem** is installed in multiple private households participating in the EVELIXIA project and used to monitor the device specific electricity consumption of heat-pumps, individual household devices like white appliances, building integrated photovoltaic (PV) generation and the grid connection point. The indoor environmental quality is determined by temperature and humidity measurements on floor level and several smart lights with dimming and RGB colour functionality are installed to allow for soft and intuitive user interactions. Proprietary energy assets are made controllable by smart switches connected to digital inputs at e.g. the heat-pump interfaces (remote shut on/off or setpoint increase by SGready) or by connecting to smart plugs with remote shut on/off functionality. Both sensor and actuator datapoints are available by the Shelly Cloud control API and stored with a sample rate of 1 min.

The local **electricity grid operator** outsourced the data collection, storage and management of the customer smart meters used for billing purposes to the provider “energyservices”. FHB managed to gain access to their database by a REST API and is able to provide the anonymised electricity consumption and production data of several smart meters within the Austrian PS to the EVELIXIA partners through the framework. In accordance with the regularity requirements the data is published in 15 min time resolution once a day for the previous day.

The overall framework handles and manages authentication and data access to all four plant-specific APIs, continuously collects anonymized operational data to a connected time series database and provides secured access to the EVELIXIA southbound API connector. The detailed description of the PS architecture and the applied data model is stated in 3.1.1.

2.1.3 Simulation Study and Results

In the outgoing year of 2024, the Austrian PS Specific Platform went online and started operation with a first patch of available datapoints, mainly for test and evaluation purposes. Several performance tests showed promising results, although adaptions in the data handling procedure were required. In particular, data storage method was upgraded from fixed time step logging to a change of value logging strategy. The connectors to the thermal storage systems (“Technische Alternative”) and the smart home ecosystems (“Shelly”) are fully functional and already in use. The connectors for district heating and electricity grid data are under development and currently successfully tested in an offline simulation environment.

2.1.4 Future Steps and Planned Timeline

From the PS platform perspective, the next step is to finalize the connectors for the district heating and the electricity grid data. The data transfer from the district heating operator to the pilot specific EVELIXIA platform is being finalized, and regular operation is going to start latest at the end of January 2025. The data transfer from the distribution system operator is currently under development, final tests are scheduled for February 2025 and regular operation is planned to start within March 2025 for the first electricity metering points.

In parallel, the already existing platform part will be continuously filled with new timeseries and metadata as the hardware in the Austrian PS gets available online.

2.2 French PS Specific Platform

The French PS is developing a hypervisor solution. The hypervisor will be hosted by a secure cloud provider to manage and oversee the operation of the whole system. Appropriately designed APIs will enable the connection with EVELIXIA's middleware layer and the necessary data ingestion from the connected grid platforms. The data communication protocol will be TCP/IP, with cycle times between 1min to 1hr.

2.2.1 Overview of the PS

The E-Factory building is the headquarters of Entech smart energies. The aim of the French PS specific platform is to demonstrate that an industrial and tertiary activity, beyond consuming as little energy as possible and producing renewable one, could also achieve maximum self-consumption, while reducing operation costs and maximizing the revenues through hybrid long term storage strategy. The challenge is to respect the regulatory framework and grid constraints.

2.2.2 Development of the French PS Specific Platform

The innovative solution proposed by the French PS is a hydrogen based long-term storage systems to optimize the self-consumption, achieve maximum possible revenues, manage and oversee the whole operation of the system on the hypervisor.

The French PS specific platform integrates energy monitoring and energy management system (EMS). Measuring devices are installed through E-Factory building to accurately monitor electricity consumption units. An EMS is under development with the objective of maximizing self-consumption, while reducing operating costs and maximizing the revenues. The EMS integrates model predictive controls that allows defining optimal energy management of resources, based on forecast data and optimisation models. The EMS can perform energy management optimizations across various time horizons and resolutions, and at each electric vehicle (EV) arrivals, to enhance its performance. The EMS strategy that is a hydrogen based long-term storage systems will consider day-ahead, intraday, and annual scheduling to maximize the annual self-consumption ratio. The optimal energy dispatch will be calculated based on PV forecast and consumption forecast at 10' resolution for day-ahead/intraday scheduling, and at 30' resolution for annual scheduling. The power management system (PMS) controls and optimizes in real time the whole system. The PS also provides visualization of system interaction and dynamics. The PS could operate in grid-connected mode or isolated in case of grid failure, or to carry out tests on Entech products.

The EMS is a containerized solution in Python 3.9+ exploiting, detecting and mitigating capacity of Pandas and Pyomo libraries. The optimization problem is formulated as a mixed integer linear programming solved by CBC solver. The EMS integrates the following models:

1. PV power plant model
    - Operates using weather forecast as inputs
    - A time series-based model that uses theoretical maximum PV production base on forecast data and estimates possible curtailment based on the forecasted power and the energy state.
2. Energy Storage System (ESS) model with Auxiliaries (AUX)
    - Models the power efficiency input/output of the ESS, using a constant efficiency coefficient of charging/discharging
    - Operates within limits of maximum charge/discharge power and state of charge (SOC) boundaries (SOCmax_ESS and SOCmin_ESS)
    - Includes an AUX that accounts for auxiliary power consumption, such as constant operational power, heating, and cooling power to mitigate heat losses.
3. Infrastructure for recharge of electric vehicles (IREV)
    - Modelled similarly to ESS without AUX, considering the maximum/minimum charging power of the EV and charging terminal (CT)
    - Requires inputs such as the time of arrival/departure, SOC of arrival and desired SOC at departure, and the energy capacity of the EV.
4. Building consumption model

- A time series-based model representing the building energy consumption.

5. Hydrogen Storage System (HSS)
    - Hydrogen Storage System is the chain of electrolyser (ELY), Compressor (COMP), Hydrogen Tank (HT), and the Fuel Cell (FC)
    - Models the H2 molar produced in the ELY and H2 molar consumed by the FC, assuming constant efficiency of ELY and FC, as well as proper gas properties
    - Operates within limits of maximum/minimum power of ELY and FC, and within limits of state of charge of H2 (SOC_H2, SOCmax_H2, SOCmin_H2) that depends on H2 pressure in the HT
    - The operation of ELY and FC simultaneously is prohibited

Additionally, the power grid model supports power supply and injection within specified limits. The EMS incorporates PV forecast, load forecasts and potential grid energy prices. It is hosted on the cloud, which input, and output parameters exchanged in JSON format via API REST. Real-time operational data from the systems are sent to the cloud platform, where they are stored in a time-series database. Optimization results, parameters, and key performance indicators (KPIs) are stored in relational database. System operation monitoring is available and built on Grafana technology. This interface is designed to visualize the PS, track the operational status of all equipment, and evaluate system performance effectively.

The hybrid long-term storage strategy, aiming to maximize the self-consumption while reducing operating costs and maximizing the revenues, integrates the following constraints:

- PV power can be curtailed to avoid over injection to power grid
- Power balance between production and consumption
- Only PV could inject power to the grid; ESS and FC are not allowed to discharge into the power grid
- Power exchange between ESS and ELY/FC is not allowed, whereas power grid could supply power to the ESS to be charged and to the ELY to produce H2
- FC and ESS can be used to satisfy loads (E-Factory and IREV)
- The cycles for turning on/off the ELY and FC is limited to custom-defined value

The hybrid long-term storage strategy is conceived on two layers optimization; annual optimization and day-ahead/intraday optimization. The annual optimization is considered the higher layer of the EMS. The inputs of the annual optimization are PV power as PV forecast over the year and the aggregated load power as load forecast, it includes the building consumption of E-factory and the IREV. Moreover, E-Factory is subcontracted to a specific energy contract with 5 tariffs periods:

- Low season, from April to October inclusive, with two tariffs:
    - On-peak periods from 7am to 11pm only during weekdays
    - Off-peak periods from 11 pm to 7am only during weekdays, full weekends, and public holidays
- High season, from Novembre to March inclusive, with three tariffs:
    - Peak hours from 9am to 11am and 6pm to 8pm from Decembre to February inclusive
    - On-peak periods from 7am to 11pm only during weekdays
    - Off-peak periods from 11pm to 6am only during weekdays, on full weekends, and public holidays.

The energy prices are inputs to the two layers of optimization. The objective of the higher layer of EMS is to optimize the use of hybrid storage systems. The outputs of the higher layer are the SOC_ESS and SOC_H2 dynamics over the year, reflecting the use of energy storage systems ESS and HSS. These outputs will serve to define the SOC of ESS and H2 to reach by the end of each day, which will be used as a starting SOC of ESS and HSS in the lower layer of optimization. The EMS objective is formulated as a multi-objective function, combining maximizing self-consumption and minimizing grid energy cost, through a weighted sum approach [1], where it should be equal to one and the weight of each objective function is in [0, 1]. However, these objectives are inherently conflicting, as prioritizing one tends to compromise the other. Specifically, maximizing self-consumption entails maximizing the utilization of PV power-either by satisfying load demands (e.g. E-Factory building or IREV), charging ESS or producing H2 in the ELY-while simultaneously minimizing reliance on grid power for supply or injection. In contrast, the goal of minimizing grid energy costs requires a different strategy: reducing grid power consumption and maximizing grid power injection to optimize revenue. These conflicting priorities necessitate careful tuning of the objective weights to balance self-consumption and grid energy cost effectively. Therefore, various simulations will be carried out to analyse the performance and impact of the weighted sum. Self-consumption can be expressed either in energy units or as a percentage, while grid energy costs are expressed in monitory terms. Therefore, unit standardization is a key element to avoid bias in the multi objective function and ensure comparability, ultimately guaranteeing balanced outcomes. To achieve this, self-consumption is monetized as cost savings, representing the reduction in expenses from purchasing energy from the power grid.

2.2.3 Simulation Study and Results

The simulation is done, to estimate and test the performance of the hybrid long-term storage systems, with a loop of simulations of time step configurable using physical model of components and control models. The energy tariffs for purchasing energy applicable for E-Factory are detailed in ***Table 1***, the feed-in tariff is considered 11.41 c€/kWh for installation between 100-500 kWp [2].

Table 1. Energy tariffs for E-Factory.

Season	Periods	Tariffs
High season	Peak hours	5.35 c€/kWh
	On-peak periods	4.01 c€/kWh
	Off-peak periods	2.45 c€/kWh
Low season	On-peak periods	1.08 c€/kWh
	Off-peak periods	0.7 c€/kWh

The PV installed capacity is 244 kWp. The limits of power grid supply and injection are 1500 kW. The parameter values of the ESS and AUX used for the simulation are presented in ***Table 2***.

Table 2. ESS and AUX parameters used for simulation E-Factory.

	Parameter	Value	Units	Description
ESS	Enom_ESS	700	kWh	Nominal energy capacity of ESS
	PmaxDch_ESS	300	kW	Maximum discharging power of ESS
	PmaxCh_ESS	300	kW	Maximum charging power of ESS
	EffCh_ESS	0.9	p.u.	Efficiency of charging ESS
	EffDch_ESS	0.9	p.u.	Efficiency of discharging ESS
	SOCmin_ESS	0.1	p.u.	Minimum state of charge ESS
	SOCmax_ESS	0.95	p.u.	Maximum state of charge ESS
	SOCinit_ESS	0.5	p.u.	Initial state of charge ESS
AUX	PauxInv_NoLoad	0.37	kW	Auxiliary power of power conversion system (PCS) at no load
	PauxInv_PartialLoad	1.8	kW	Auxiliary power of PCS at partial mode of charge/discharge
	PauxInv_FullLoad	8.1	kW	Auxiliary power of PCS at full mode charge/discharge
	PessPu_PartialLoad	0.75	p.u.	Fraction of maximum power PCS correspond to partial mode
	PthermMin	0.898	kWt	Minimum thermal power
	PthermMax	8.98	kWt	Maximum thermal power
	PauxCst	2.51	kW	Constant auxiliary power
	COP	2.2	p.u.	Coefficient of performance of climatization

The parameter values of the HSS used for simulation are presented in ***Table 3***.

Table 3. Parameter values of the HSS used for simulation E-Factory.

	Parameter	Value	Units	Description
ELY	Pmax_ELY	30	kW	Maximum power of ELY
	Pmin_ELY	10	kW	Minimum power of ELY
	Eff_ELY	0.57	p.u.	Efficiency of EL
	PcompPu	0.026	p.u.	Power ratio of compressor correspond to ELY power
	Ploss_ELY	0	kW	Power loss of ELY
	Cycles_OnOff_ELY	2	p.u.	Number of cycles to turn on and off the FC per day
HT	PrMin_H2	10	Bar	Minimum pressure of H2
	PrMax_H2	300	Bar	Maximum pressure of H2
	PrInit_H2	150	Bar	Initial pressure of H2
	V_H2	0.45	m3	Volume of HT
	LossRate_HT	0	p.u.	Loss rate in HT
FC	Pmax_FC	1	kW	Maximum power of FC
	Pmin_FC	10	kW	Minimum power of FC
	Eff_FC	0.52	p.u	Efficiency of FC
	Ploss_FC	0	kW	Power loss of FC
	Cycles_OnOff_FC	2	p.u.	Number of cycles to turn on and off the FC per day

The EV data inputs used for the simulation are presented in ***Table 4***. These data inputs are repeated for the annual simulation for simplicity. In the simulation the number of CT is considered two, and each terminal has two charging points (CP), therefore, four charging points are available for EV charging. The parameter values of CT are presented in ***Table 5***.

Table 4. EV data inputs used for simulation E-Factory.

	EV1	EV2	EV3	EV4	Units	Description
Arrival time	07:30	08:00	08:30	09:00	HH:MM (AM)	Arrival time of EV
Departure time	05:00	05:30	06:00	06:30	HH:MM (PM)	Departure time of EV
SOCmin_EV	0	0	0	0	p.u.	Minimum state of charge of EV
SOCmax_EV	1	1	1	1	p.u.	Maximum state of charge of EV
SOCarr_EV	0.1	0.1	0.1	0.1	p.u.	State of charge of EV at arrival
SOCdep_EV	0.95	0.95	0.95	0.95	p.u.	State of charge of EV desired at departure
Enom_EV	75	75	75	75	kWh	Nominal capacity of EV
EffCh_EV	0.95	0.95	0.95	0.95	p.u.	Efficiency of charging EV
PmaxCh_EV	50	50	50	50	kW	Maximum power of charging EV
PminCh_EV	0	0	0	0	kW	Minimum power of charging EV
PlossFix_EV	0	0	0	0	kW	Fixed power loss of EV

Table 5. Parameter values of the IREV used for simulation E-Factory.

	Parameter	Value	Units	Description
	Pmax_CT	100	kW	Maximum power of CT
	Pmin_CT	0	kW	Minimum power of CT
	EffCh_CT	0.95	kW	Efficiency of charging CT
CT	EffDch_CT	0.95	kW	Efficiency of discharging CT
	Pmax_CP	100	kW	Maximum power of CP
	Pmin_CP	0	kW	Minimum power of CP
	PlossFix_CT	0	kW	Fixed power loss of CT

Annual optimization – higher level optimization layer

A balanced 50/50 weighting approach is applied to the multi-objective function. The annual optimization requires PV production, load consumption, as shown in ***Figure 1***, and the energy tariffs as described earlier. The dynamic variation of SOC_ESS and SOC_H2 through the year is presented in ***Figure 2***. The grid energy variation is presented in ***Figure 3***. The PV production is high in low season and low in high season, whereas as the load consumption is slightly higher in high season than in low season.

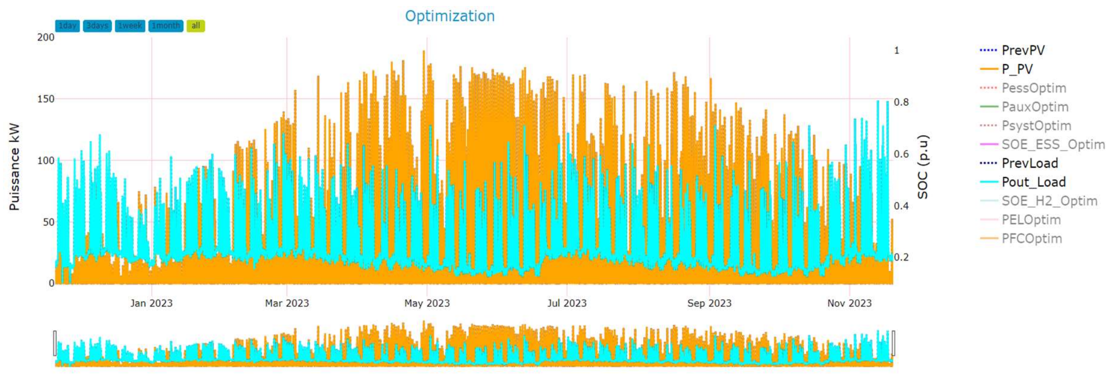


Figure 1. PV production and load consumption over a year at E-Factory.

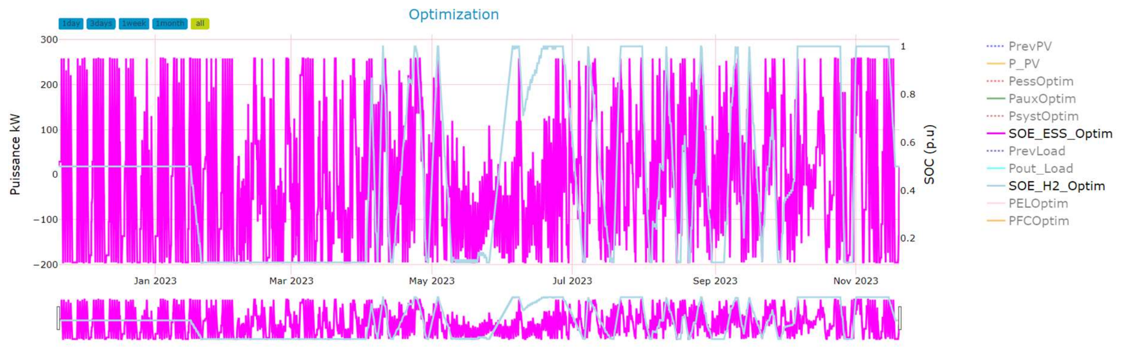


Figure 2. Annual Optimization of SOC_ESS and SOC_H2 for E-Factory.

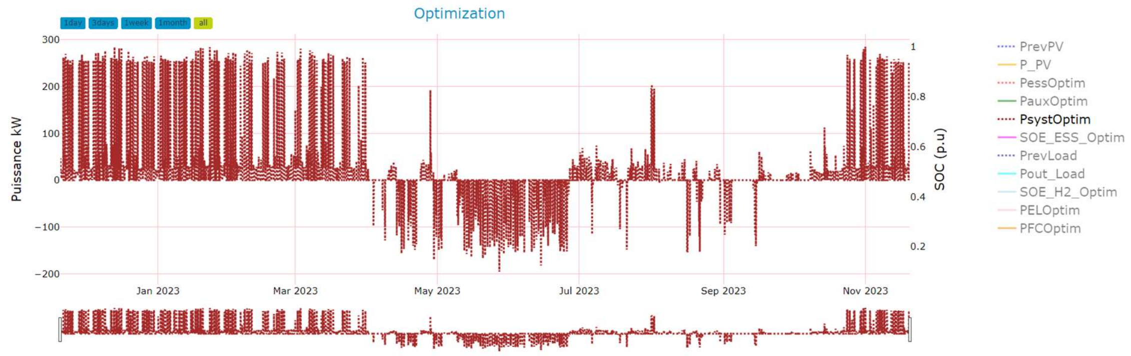


Figure 3. Annual Optimization of Power grid variation for E-Factory.

The annual optimization results show the dynamic variation of SOC_ESS and SOC_H2 through the year, as shown in ***Figure 2***. The ESS is charged and discharged daily to increase the rate of self-consumption and reduce the dependency on grid purchases. In low season, the ESS operates in reduced margin as PV production is high and load consumption is low, therefore, PV production can satisfy load demand and PV excess can be injected into the power grid, and thus making revenues, as shown in ***Figure 3***. As for the HSS, during high season, the HSS is less used as PV production low and load consumption is high, therefore, HSS is discharged once and reaches the SOCmin_H2 and then recharged during low season, to be thereafter discharged during the following periods. This annual optimization achieves the desired objective by maximizing the self-consumption, minimizing grid energy costs, and achieving maximum revenues.

Day-ahead/Intraday optimization – lower-level optimization layer

The outputs from the higher-level optimization, SOC_ESS and SOC_H2, are extracted and passed to the lower-level optimization layer to enforce the SOC_ESS and SOC_H2 targets at the end of each day in the simulation. The lower-level optimization layer operates on a daily basis as part of an annual simulation. Within this layer, the EMS performs day-ahead optimization using PV and load forecasts. The EMS is updated multiple times throughout the day–specifically, four times when new PV forecasts are received and at each EV arrival. These updates adjust the actual statuses of the SOC_ESS and SOC_H2, correcting any deviations from the targets. Since E-Factory building assets are inflexible loads, they must always be satisfied. However, the IREV could be controlled and optimized to maintain system balance. A 3-day simulation in low season is presented in ***Figure 4***, and a 3-day simulation in high season is presented in ***Figure 5***.

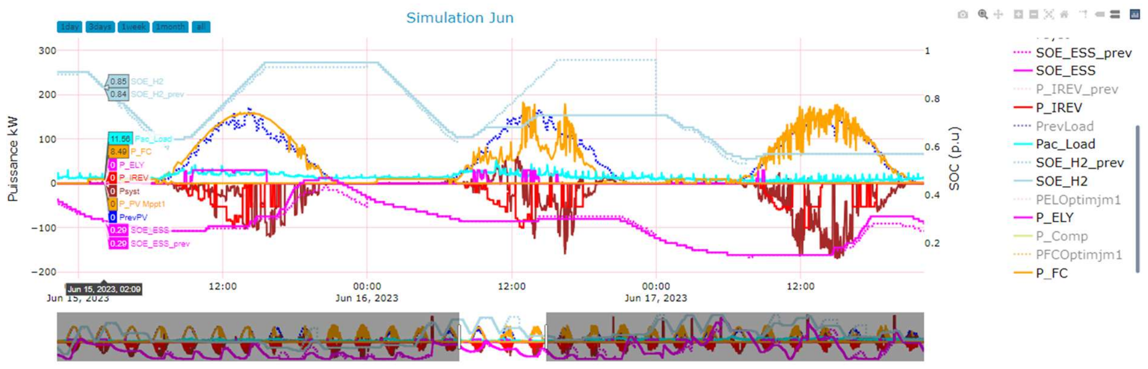


Figure 4. 3-day simulation in low season at E-Factory.

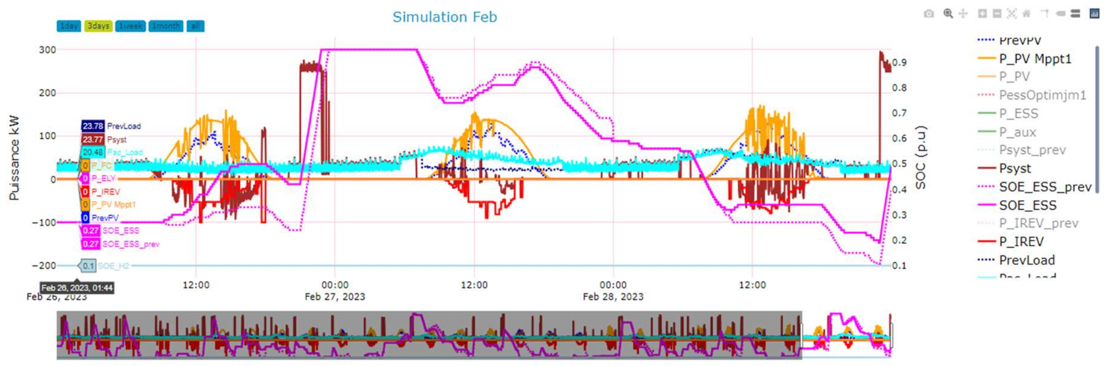


Figure 5. 3-day simulation in high season at E-Factory.

In conformity with the higher-level optimization layer, the HSS maintains SOCmin_H2in February, as shown in ***Figure 5***. This simulation illustrates the dynamic variations in SOC_ESS as it strives to meet its end-of-day target. In conformity with the higher-level optimization layer, SOC_ESS and SOC_H2 strive to meet their respective end-of-day targets during the low season, as shown in ***Figure 4***. During this period, grid injection increased, along with reduced ESS charging and lower HSS requirements. Conversely, in the high season, grid supply increases to meet higher load demands, as the HSS remains empty. The result of the monthly energy data is presented in ***Figure 6***. In high season, the grid energy consumption is significant as the PV production is low, representing higher grid-to-load energy consumption. On the other hand, in low season, the grid energy consumption is significantly lower than that in high season as PV production is high, representing higher PV-to-load energy consumption and higher grid energy injection.

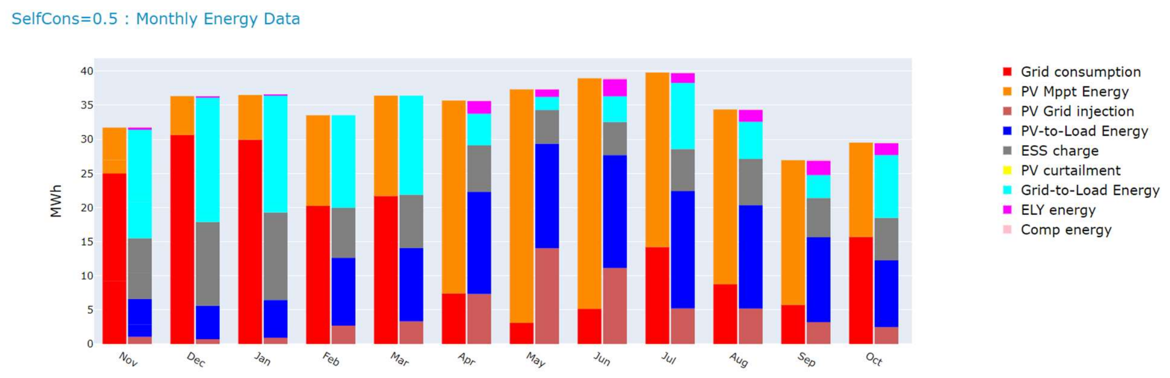


Figure 6. Monthly energy data of an annual simulation of E-Factory.

The monthly KPIs results of a balanced 50/50 weighting approach are presented in ***Table 6***. The annual KPIs results of a balanced 50/50 weighting approach are presented in ***Table 7***.

Table 6. Monthly KPIs of a 50/50 weighting approach for E-Factory.

KPIs	Nov 2022	Dec 2022	Jan 2023	Feb 2023	Mar 2023	Apr 2023	May 2023	Jun 2023	Jui 2023	Aug 2023	Sep 2023	Oct 2023	Nov 2023
Net revenues (k€)	-14. 83	-47.50	-44.20	-17.34	-15.40	46.58	94.49	73.60	28.94	31.33	19.07	9.31	-20.20
Revenues (k€)	0.67	4.81	6.23	18.38	22.71	50.18	96.01	76.22	35.68	35.58	21.90	16.93	6.52
Grid costs (k€)	15.51	52.31	50.43	35.73	38.11	3.60	1.51	2.62	6.73	4.24	2.83	7.62	26.73
Savings self-consumption (k€)	4.37	11.88	13.11	24.52	24.84	11.94	11.81	13.58	11.55	11.71	10.33	6.54	8.04
Grid total energy (MWh)	9.09	29.94	29.03	17.56	18.37	0.07	-10.94	-5.97	9	3.56	2.55	13.23	14.86
Grid consumption (MWh)	9.19	30.65	29.94	20.24	21.69	7.4	3.09	5.16	14.21	8.76	5.74	15.7	15.82
Grid injection (MWh)	0.1	0.7	0.91	2.69	3.32	7.33	14.03	11.13	5.21	5.2	3.2	2.47	0.95
Grid-to-load consumption (MWh)	5.32	18.23	17.12	13.58	14.52	4.63	1.93	3.77	9.7	5.44	3.36	9.2	10.59
PV-to-load consumption (MWh)	1.82	4.91	5.54	9.95	10.75	14.98	15.33	16.56	17.21	15.13	12.45	9.79	3.74
PV self-consumption (MWh)	1.86	4.98	5.65	10.61	11.39	20.95	20.21	22.64	20.36	20.41	18	11.33	3.8
Self-consumption rate (%)	94.94	87.61	86.12	79.8	77.44	74.07	59.03	67.03	79.61	79.7	84.9	82.08	79.93
Self-sufficiency rate (%)	11.15	8.02	9.41	31.04	31.41	71.19	85.86	79.42	56.03	67.35	72.49	35.83	16.19
Grid dependency (%)	88.85	91.98	90.59	68.96	68.59	28.81	14.14	20.58	43.97	32.65	27.51	64.17	83.81
ESS dependency (%)	30.82	30.32	31.08	19.81	20.08	21.34	19.17	15.48	15.23	20.52	20.88	20.35	23.08
FC dependency (%)	0.24	0.22	0.36	0	0	2.4	1.84	3.49	1.52	2.79	3.4	2.03	1.02
ELY energy (MWh)	3.49	3.58	1.49	0	0	6.71	3.17	7.57	5.77	6.86	10.06	12.79	5.22
ESS nb cycles equivalent	4.9	16.3	16.7	10.1	10.1	8.3	6.3	6.3	8.5	8.3	6.7	7.9	6.6
PV curtailment (%)	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 7. Annual KPIs of a 50/50 weighting approach for E-Factory.

KPIs	Value
Net revenues (k€)	143.85
Revenues k€)	391.90
Grid costs (k€)	248.04
Savings self-consumption (k€)	164.22
Grid total energy (MWh)	130.35
Grid consumption (MWh)	187.59
Grid injection (MWh)	57.25
Grid-to-load consumption (MWh)	117.38
PV-to-load consumption (MWh)	138.16
PV self-consumption (MWh)	172.17
Self-consumption rate (%)	75.05
Self-sufficiency rate (%)	43.78
Grid dependency (%)	56.22
ESS dependency (%)	22
FC dependency (%)	1.42
ELY energy (MWh)	5.79
ESS nb cycles equivalent	117.2
PV curtailment (%)	0

Given that the optimization problem is a multi-objective function, various simulations were conducted to evaluate the performance and impact of the weighted sum approach. Five scenarios were analysed, where the weight of self-consumption is assigned to 1, 0.75, 0.5, 0.25, and 0 respectively. The results of these simulations are presented in ***Figure 7***, ***Figure 8***, and ***Figure 9***. Additionally, pareto fronts are presented in ***Figure 10***, and ***Figure 11***. Detailed KPIs results of the five simulations are presented in Annex 1.

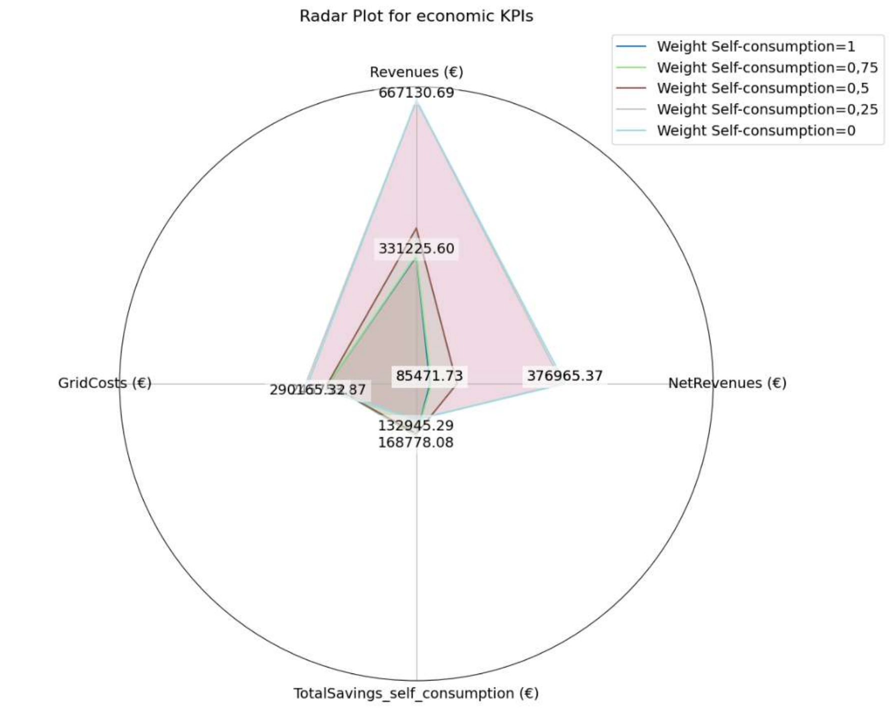


Figure 7. Radar plot for economic KPIs of E-Factory.

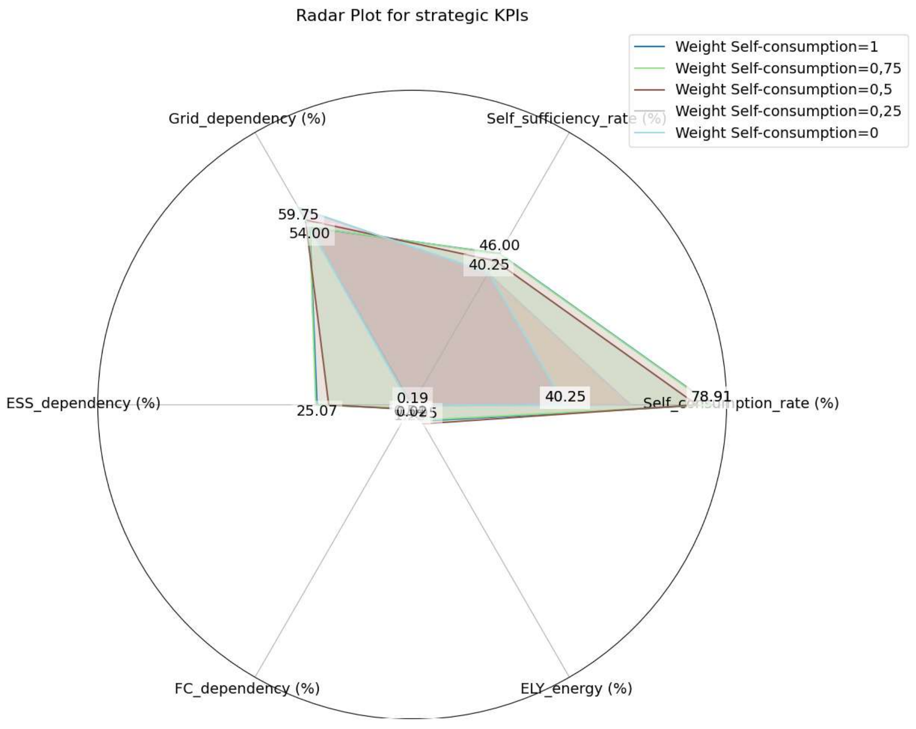


Figure 8. Radar plot for strategic KPIs of E-Factory.

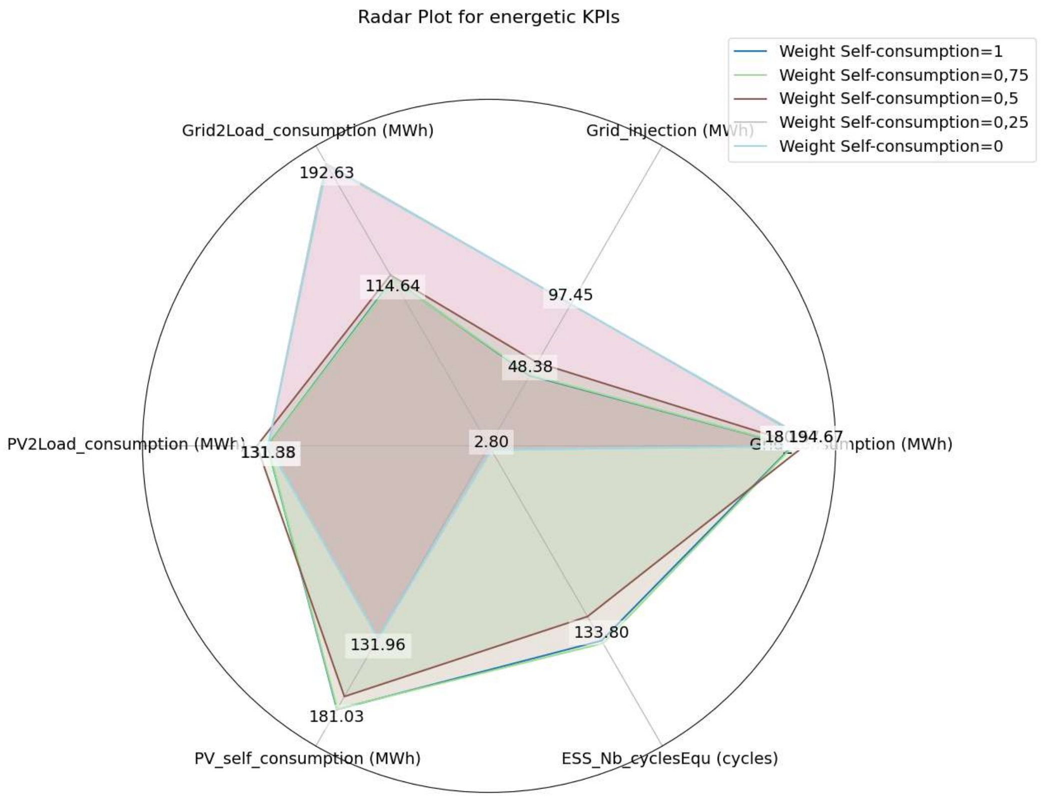


Figure 9. Radar plot for energetic KPIs of E-Factory.

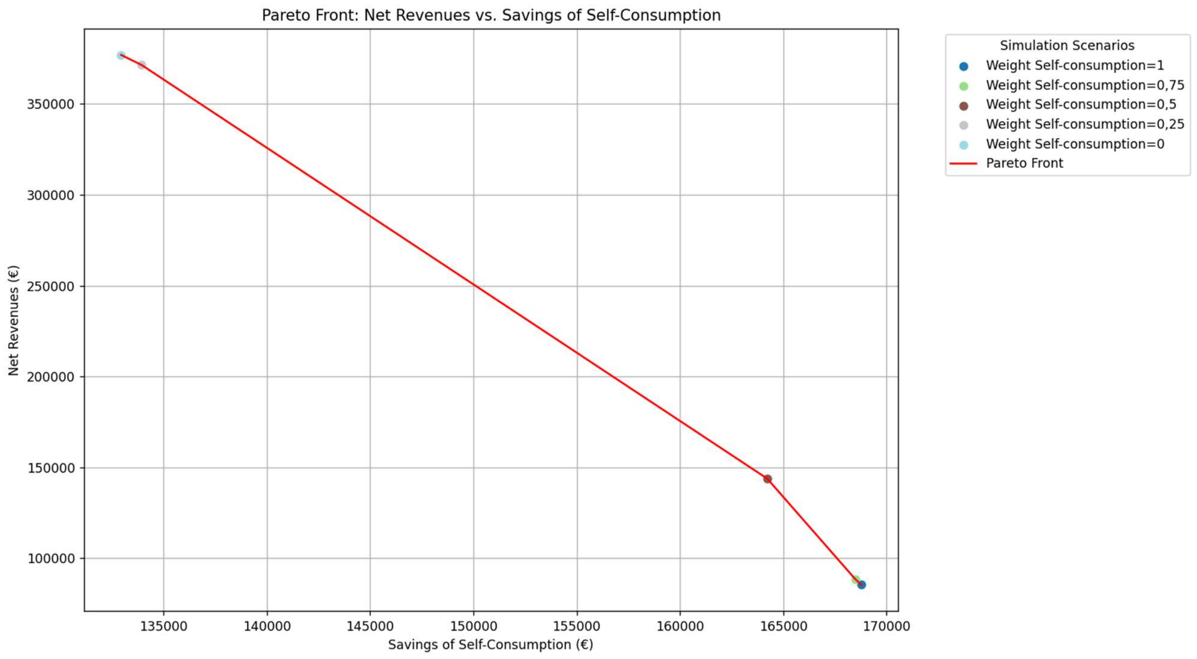


Figure 10. Net revenues vs savings of self-consumption of E-Factory.

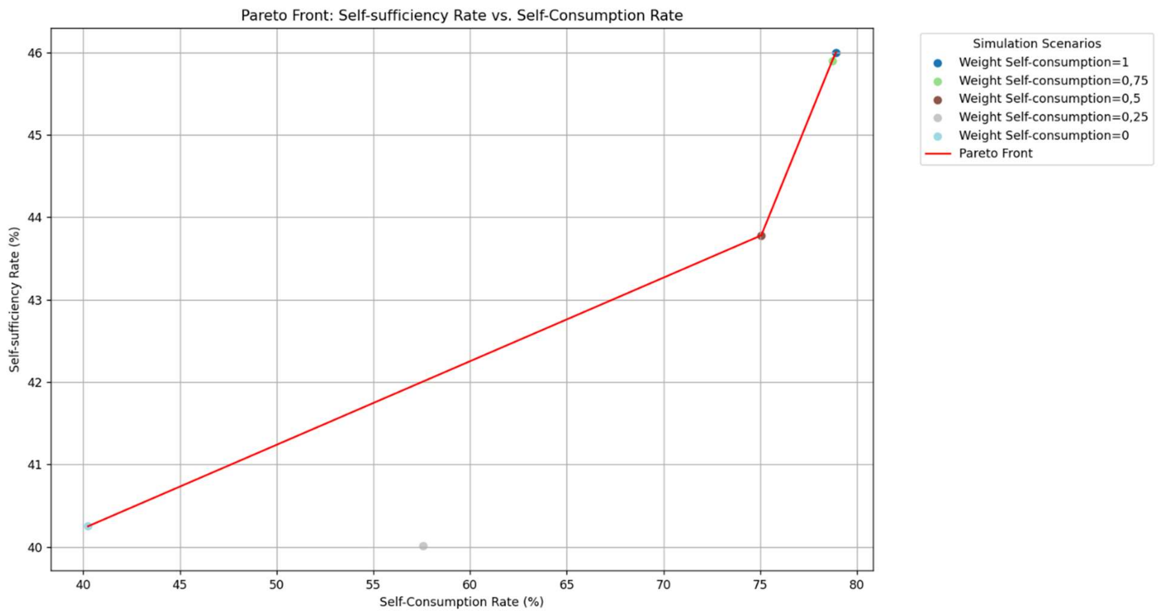


Figure 11. Self-consumption rate Self-sufficiency rate of E-Factory.

The comparison of these key KPIs shows that when the weight of the objective function is shifted from prioritizing self-consumption to minimizing grid energy cost:

1. Net revenues increase, but the savings generated through self-consumption decrease
2. The total energy consumed and injected from the grid increase. Consequently, the amount of energy supplied from the grid to meet the load demand increases, resulting in a significant rise in grid dependency
3. The PV to load consumed energy remains generally stable, except in the case of balanced weighting (50/50), where it reaches its peak. However, PV self-consumption energy decreases, leading to a significant drop in the self-consumption rate (from 78% to 40%) and the self-sufficiency rate (from 46% to 40.5%). No excess PV production is observed, leading to no PV curtailment
4. Dependency on the ESS significantly decreases, dropping from 25% to 0.54%. This also results in a drastic reduction in the annual equivalent cycles of the ESS, which fall from 133 to only 2.8 cycles per year
5. The energy used by the ELY decreases from 5.79% to 0.3%. This also leads to a reduction in the dependency on the HSS, which decreases from 1.4% to nearly 0.02%.

For the simulation results over the five scenarios considered, the following conclusions can be drawn:

- When the weight of self-consumption > 0.5, maximizing self-consumption is prioritized. This leads to a preference for charging the SOE_ESS first and the SOC_H2 second, with no energy injection into the grid (i.e., no revenue from grid injection) and increased grid consumption
- Alternatively, when the weight of self-consumption < 0.5, minimizing grid energy cost becomes priority. This results in a preference for injecting energy into the grid (generating revenue from grid injection) while reducing grid consumption. However, in this case, self-consumption Is not adequately achieved or guaranteed. This often results in low SOE_ESS and SOC_H2 targets at the end of the day during peak seasons, requiring more energy to be drawn from the power grid
- A balanced weighting (50/50) yields better results. In this scenario, PV energy is sold during periods of PV surplus while ensuring sufficient SOE_ESS and SOC_H2 targets at the end of the day to meet the energy needs of the following day, especially during periods of high PV production
- Additionally, during high seasons, a guaranteed SOE_ESS at the end of the day ensures the system starts the next day ready to maximize self-consumption. During low season, the ELY produces H2 to maximize self-consumption and energy efficiency.

A balanced weighting (50/50) of the multi-objective function could achieve a desirable compromise. This configuration provides:

- A high self-consumption rate of approximately 75.05%
- A significant self-sufficiency rate of around 43.78%
- Substantial contributions from the storage system, with the ESS accounting for 22% and the HSS for 1.42%, resulting in a total hybrid storage contribution of 23.42%
- Considerable net revenue generated of 143.85 k€

2.2.4 Future Steps and Planned Timeline

The next step is transitioning EMS to a secure, hypervisor-based cloud platform. This will enable centralized control and oversight of system operations, enhancing security, and operational efficiency.

2.3 Danish PS Specific Platform

The Danish PS is developing a hypervisor solution in a secure cloud computing environment to manage and oversee the operation of the whole system in the Danish PS. Appropriately designed APIs will enable the data exchange with EVELIXIA's middleware layer from the connected grids with cycle times varying from less than 1min to 5min, dependent on the service offering.

2.3.1 Overview of the PS

The Danish pilot site, located in Aabenraa, Southern Denmark, is part of the Kolstrup Housing Association. The pilot focuses on typical Danish housing association buildings from the 1970s, which were renovated in 2015 with PV panels and batteries. The main objectives are to:

- Maximize local consumption of electricity generated on-site.
- Optimize local electricity usage based on variable local tariffs and electricity prices.
- Utilize PV electricity to power an electric heater installed in a domestic hot water (DHW) tank.
- Test concepts and business cases for exporting energy to the district heating (DH) grid.
- Collaborate with the Distribution System Operator (DSO) to explore optimized grid load management and reduce grid bottlenecks.
- Evaluate the potential for delivering balancing services to Balance Responsible Parties (BRPs) and the Transmission System Operator (TSO).

In Denmark, Energinet currently procures balancing services from aggregators, establishing a market for grid stability. However, these services are predominantly provided by large power plants, with no local models available for community-level aggregators. This pilot aims to address this gap by leveraging batteries, PV panels,

and hot water buffer tanks to enhance energy flexibility. These technologies will support both the electricity grid and the local district heating system.
Through these solutions, the pilot seeks to demonstrate how residential buildings can actively contribute to grid services, advancing sustainable energy use at the community level.

2.3.2 Development of the Danish PS Specific Platform

In the Danish pilot, various activities are being conducted. Each activity is described below, along with its current status.

Analyse buildings in department 15

The EVELIXIA pilot site in Aabenraa covers department: "Afd. 15 - Hørgaard, Frueløkke og Uglekær – Aabenraa" of the Kolstrup Housing Association, which includes the following addresses:

- Frueløkke 2-16 (6 buildings)
- Uglekær 2-16 (2 buildings)
- Hørgård 6-40 (6 buildings)

It has been decided to retrofit the following two buildings within this department:

- Uglekær 2-8
- Hørgård 36-40

The remaining buildings will be used as reference buildings.

Status: Done

Collect data from Existing installation

Description: In the department there is one shed per building, which controls the Huawei PV system, consisting of batteries and inverter. An example of a shed is shown in ***Figure 12***. Inside the shed the batteries and inverters are installed as shown in ***Figure 13***.

Figure 12. Shed with Huawei battery and inverter.

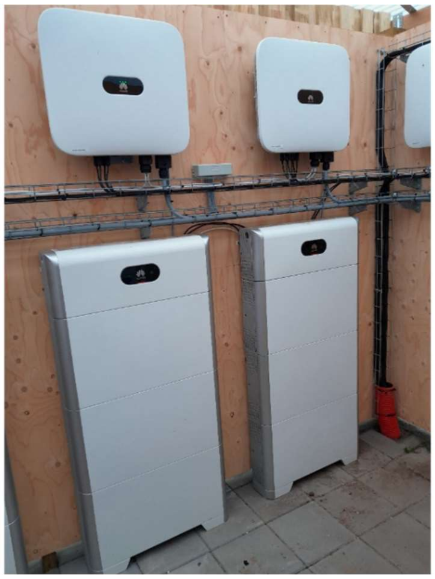
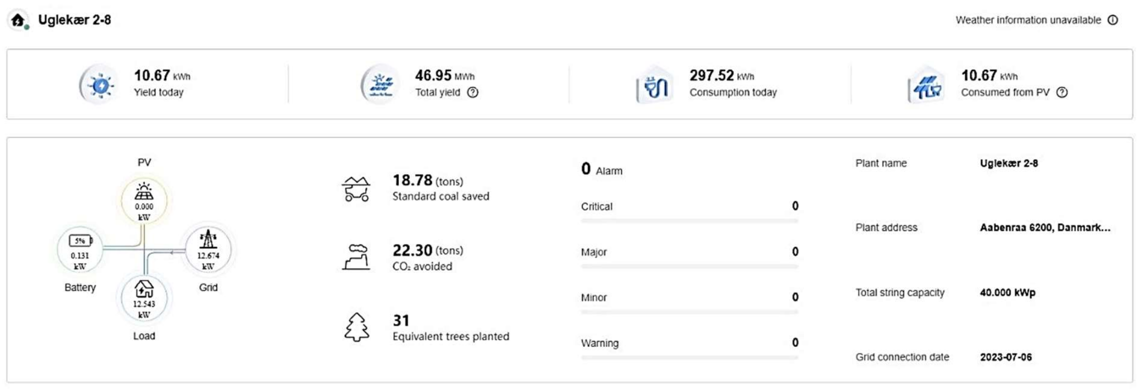


Figure 13. Huawei battery and inverter.

An API to Huawei is setup and data is collected and stored. An example of what type of data is available is shown in ***Figure 14***.

Funded by the European Union



Uglekær 2-8

Weather information unavailable

10.67 kWh Yield today

46.95 MWh Total yield

297.52 kWh Consumption today

10.67 kWh Consumed from PV

PV 0.000 kW

Battery 5% 0.131 kW

Grid 12.674 kW

Load 12.543 kW

18.78 (tons) Standard coal saved

22.30 (tons) CO₂ avoided

31 Equivalent trees planted

0 Alarm

Critical	0
Major	0
Minor	0
Warning	0

Plant name	Uglekær 2-8
Plant address	Aabenraa 6200, Danmark...
Total string capacity	40.000 kWp
Grid connection date	2023-07-06

Figure 14. Typical content available from Huawei server via API.

For the two test buildings, heat and DHW are produced by district heating, and they are controlled by 2 Danfoss ECL 310. A diagram for those is shown in ***Figure 15***.

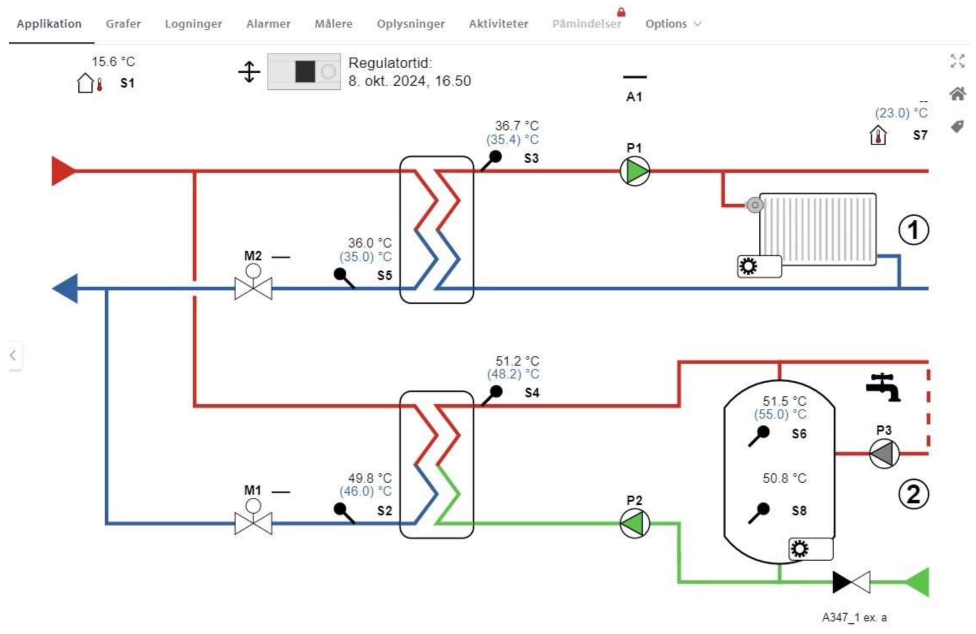


Figure 15. Schematic of heat and DHW control for reference building.

To get data collected, Neogrid's gateway "predator" is installed in the technical room and is connected via RS-485 to the ECLs. Thereby data with minute resolution flow to Neogrid server. A front picture of the predator is shown in ***Figure 16***.

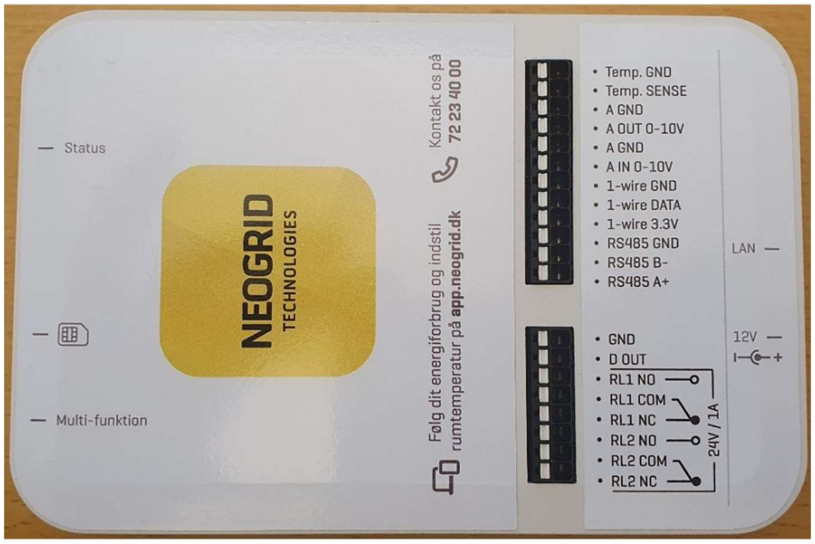


Figure 16. Neogrids gateway "predator" which connects to ECLs.

Consumption of district heating is measured in each building by a Kamstrup meter as shown in ***Figure 17***. This meter also transmits wM-Bus which is then collected by Neogrids gateway.

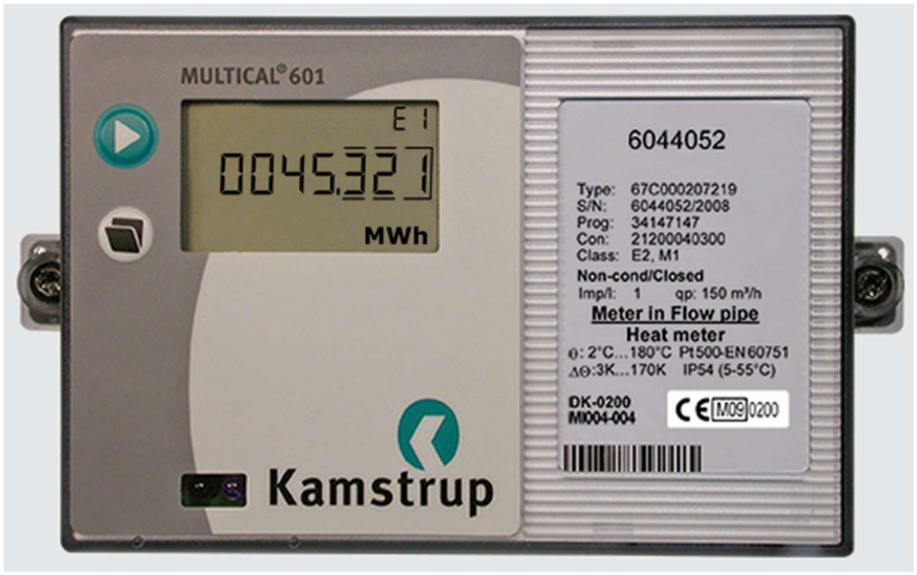


Figure 17. Kamstrup energy meter measuring district heating consumption.

Electricity data is also collected from the two test buildings. Apartment-level electricity consumption is measured by Brunata using sub-meters installed in each apartment. The data is then transferred daily to Neogrid via a Brunata API for data exchange. Electricity from the main meter, including both production and consumption, is collected hourly by the local DSO. This data is subsequently made

available to Neogrid through an API connected to the Danish DataHub, which provides access to data from all billing meters in Denmark.
Status: Done

Model data in Neogrid building setup and start API

Description: After collecting data from test buildings and reference buildings, the modelling of the buildings was initiated in the Neogrid building setup. This means the buildings energy related installations are modelled and configurated in a standardized way on Neogrid's platform. Thereby Neogrids tool suite of analysis and control algorithms can easily be applied. As a result of this all data is now available to external partners through Neogrid's API.
Status: Done.

Technique room upgrade

Technique room upgrade is planned according to ***Figure 18***. Two scenarios are specified.

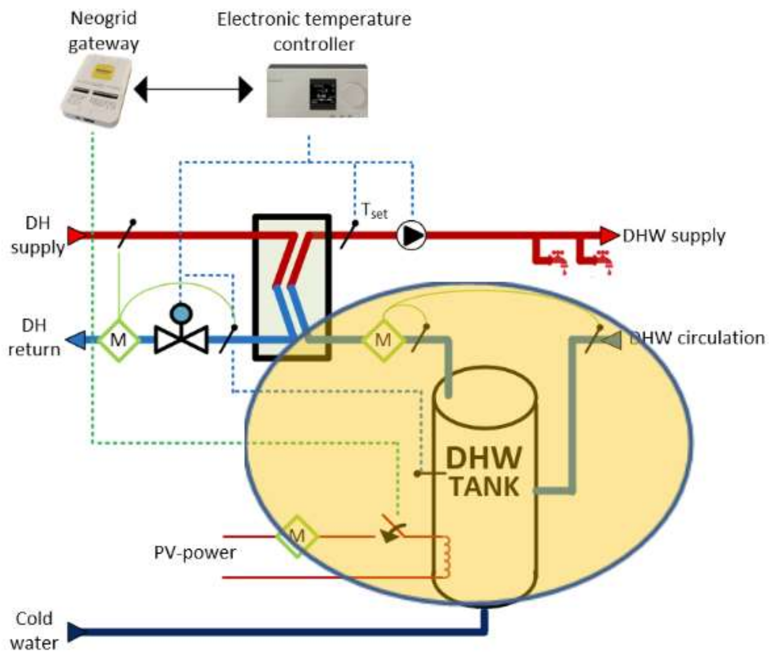


Figure 18. DHW tank with electric heater installed in technique room.

Scenario 1 A DHW tank to use surplus PV power is installed where activation of the electrical heater is externally controlled and electricity power and energy flow in/out of tank is measured.

Scenario 2, shown in ***Figure 19***, exports energy to the district heating system. This upgrade installs an accumulation tank with electric heater in series with district heating return and thereby we can export energy to the district heating grid.

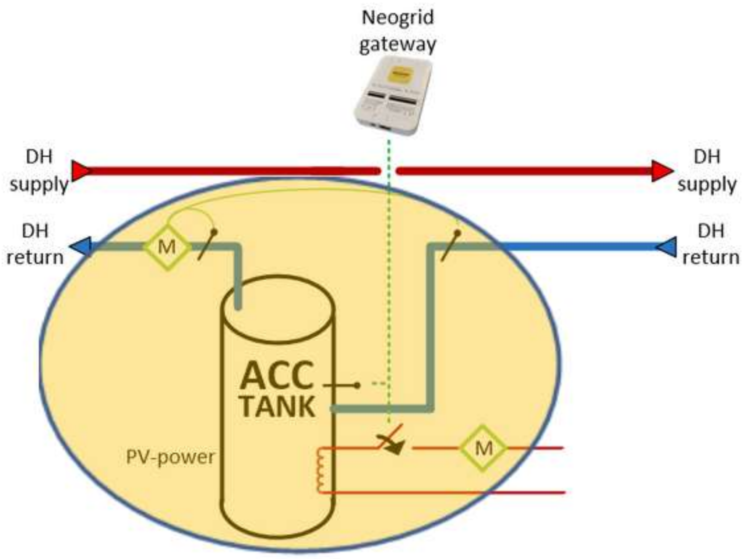


Figure 19. ACC tank with controllable heater installed in DH return.

Status: Specified, approval from housing association pending.

<u>Apartment upgrade</u>

Description In the apartments of the test buildings 2 types of upgrades are planned as shown in ***Figure 20***. It includes a humidity/temperature lorawan Elsys sensor and a lorawan mClimate radiator thermostat, which will be installed in the living room.

Figure 20. Elsys Ti/RH sensor and radiator thermostat.

Status: Specified, approval from housing association pending.

Tenant app

Description A tenant app will be developed and rolled out in the test buildings with features aimed at providing transparency, engaging tenants, and promoting efficient energy use. Here's a summary of the app's features:

1. Apartment Data: Displays electricity and heating consumption, enabling tenants to monitor their usage.
2. Price Information & Flexibility: Provides information on electricity pricing and flexibility opportunities, helping tenants optimize their energy consumption.
3. Nudging for Efficient Behaviour: Encourages tenants to adjust their consumption during periods of low prices and tariffs, while supporting co-sector coupling (e.g., between district heating and electricity).

Status: Partly specified, prototype under development.

2.3.3 Simulation Study and Results

There are two use cases defined with the Danish PS:

Table 8. Danish PS use cases.

ID	Use Case	Description
UC-DK#1	Electricity Optimisation on Building Level	Optimised operation of inverter, battery, and possibly consuming assets, to minimize cost of electricity.
UC-DK#2	Optimisation of District Heating Consumption and Production	Optimised operation of district heating assets, including conversion of surplus electricity to energy for district heating network.

The main activities in the near future to support the use cases are:

- Finalize retrofit installations and setup data collection
- Setup DSO interaction to impede grid bottlenecks
- Include TSO and ancillary services
- Create relevant baselines
- Setup experiments i.e. to detect and estimate flexibility

As an example of ongoing analysis an energy signature of one of the test locations is shown in ***Figure 21***.

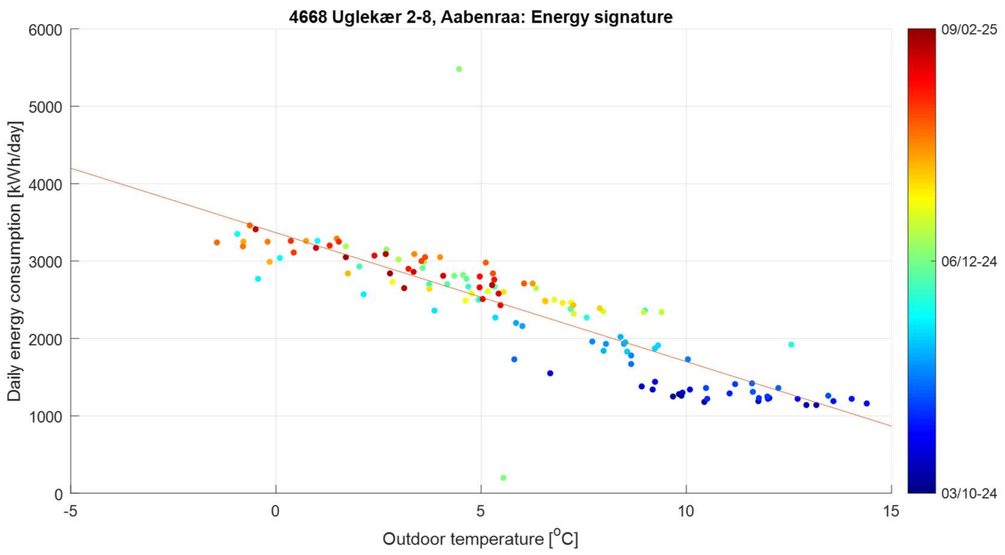
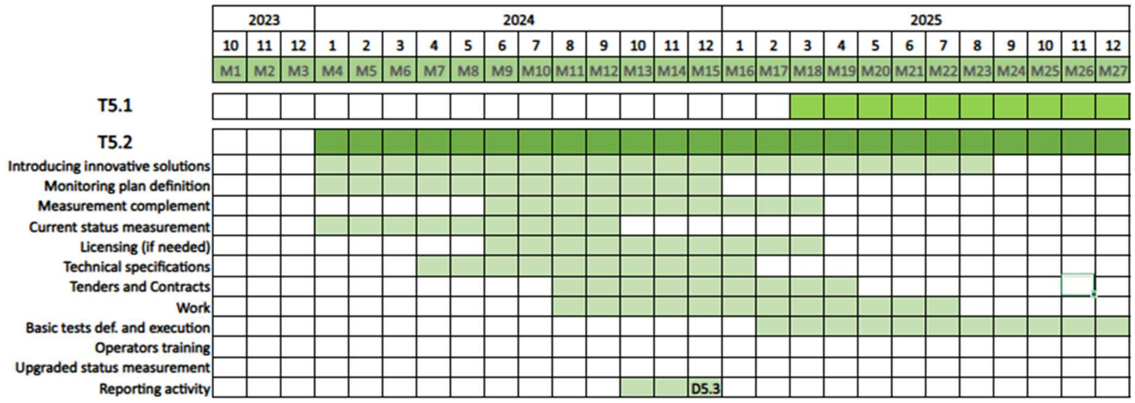


Figure 21. Energy signature for test location.

2.3.4 Future Steps and Planned Timeline

The future steps are lined up in previous section and the planned timeline is shown in ***Figure 22***.



	2023			2024													2025											
	10	11	12	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3	4	5	6	7	8	9	10	11	12
	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15	M16	M17	M18	M19	M20	M21	M22	M23	M24	M25	M26	M27
T5.1																		🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
T5.2				🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Introducing innovative solutions				🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩				
Monitoring plan definition				🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩												
Measurement complement									🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩									
Current status measurement				🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩															
Licensing (if needed)									🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩									
Technical specifications							🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩											
Tenders and Contracts											🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩								
Work											🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩					
Basic tests def. and execution																	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Operators training																											
Upgraded status measurement																											
Reporting activity													🟩	🟩	D5.3												

Figure 22. Implementation schedule of the Danish pilot site.

3 DEVELOPMENT OF APIS

The development of APIs on all seven PS is crucial to enable smooth data collection locally, so as EVELIXIA's platform southbound APIs and connectors may be able to capitalize on and establish secure and reliable bidirectional communication channels. All PS will work on the development and adaptation of the necessary field platform API connectors.

3.1 Architecture of PS

The following subsections present the architecture of each PS and the connection to EVELIXIA's platform. These architectures are designed to ensure seamless data exchange and interoperability.

3.1.1 Austrian PS Architecture

The Austrian PS architecture, shown in ***Figure 23***, manages the connectivity of four pilot specific APIs, ensures a continuous, anonymized and structured data recording and provides a secure access to the EVELIXIA southbound API connector with documented GET and PUT commands.

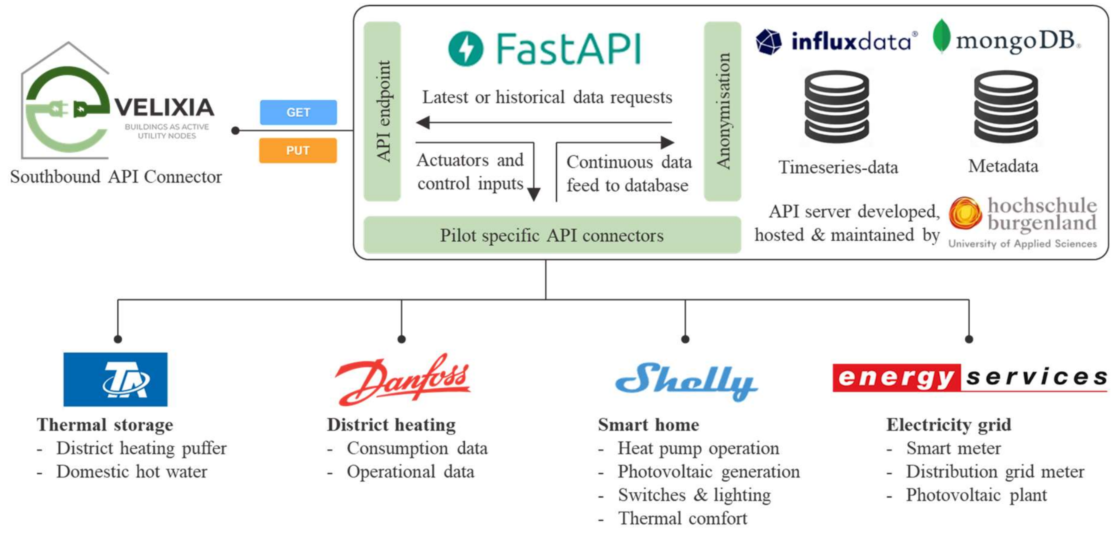


Figure 23. Austrian PS architecture.

3.1.2 Romanian PS Architecture

The Romanian PS architecture is composed of three main components: the sensorial network, the measurement infrastructure and the hosted local database with an API endpoint provided to facilitate communication with the EVELIXIA platform. The architecture of the Romanian PS is presented in ***Figure 24***.

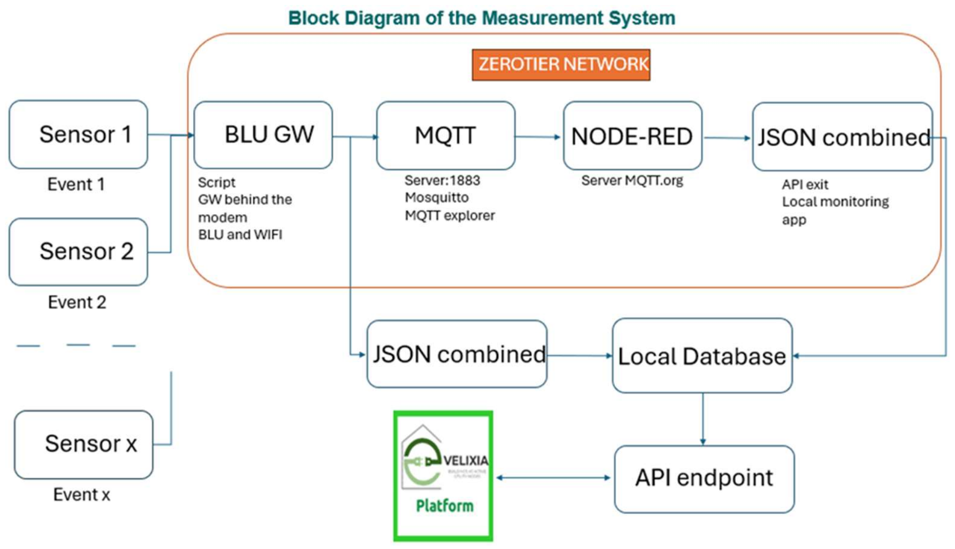


Figure 24. Romanian PS architecture.

3.1.3 French PS Architecture

The French PS architecture is presented in ***Figure 25***. The hypervisor platform will ensure seamless data exchange and interoperability. The French PS can operate as a standalone, since the control modules and the micro services are ensured and maintained within the PS, as well as the storage of data and results in the database. The architecture will ensure interoperability with the EVELXIA platform.

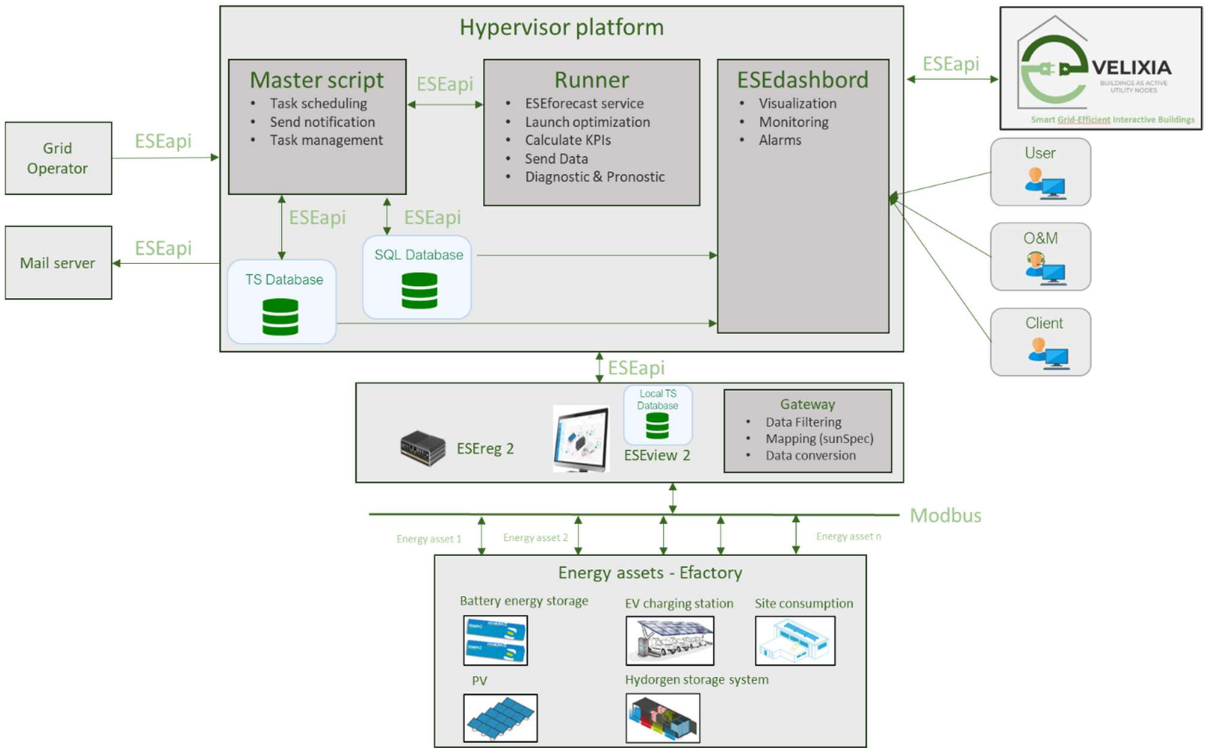


Figure 25. French PS architecture.

3.1.4 Danish PS Architecture

The Danish PS utilizes Neogrid's PreHEAT cloud energy management solution, an integrated platform for real-time data collection, data-driven control, and performance analysis. Its communication system is based upon a gateway, which is capable of communicating with the local BMS system and IoT sensors, as well as Neogrid's cloud servers. The structure of the setup is presented in ***Figure 26***.

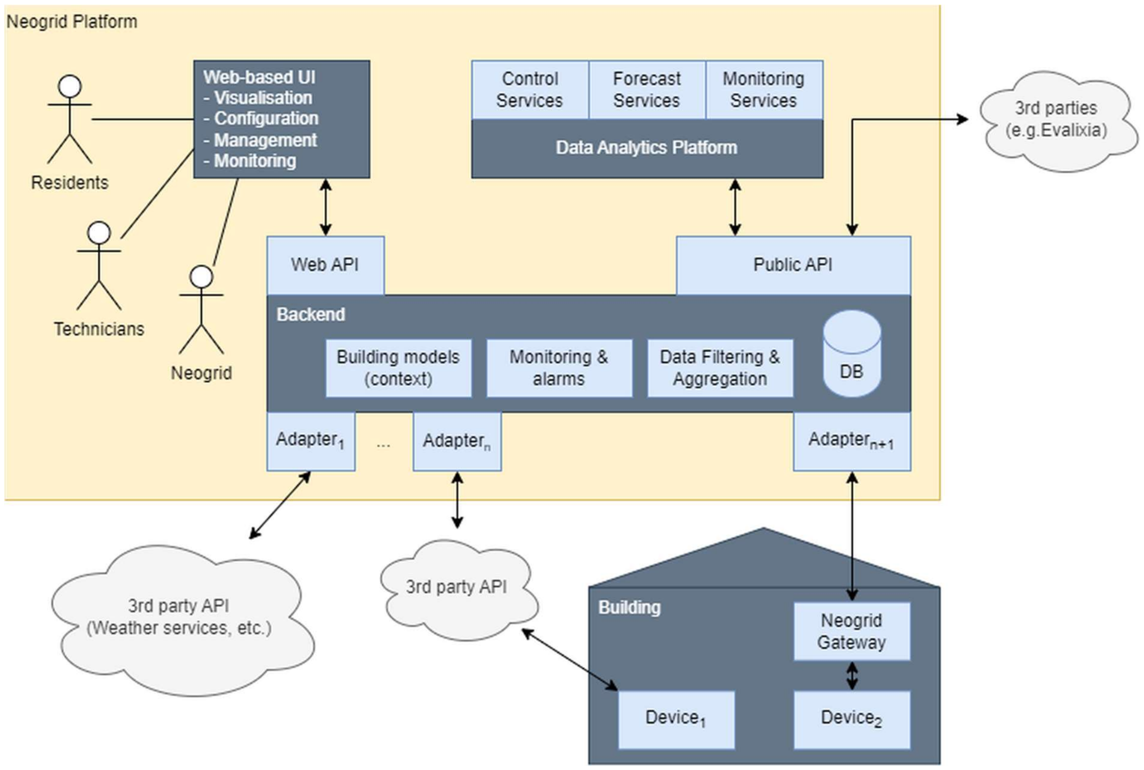


Figure 26. Danish PS architecture.

The gateway supports reading of data and writing of setpoints to BMS systems based upon BACnet and Modbus (including their IP versions). It also supports direct communication with Danfoss ECL-series of heating controllers (110, 210, 310) and Wireless IoT sensors supporting Wireless Mbus. It also connects to a LoraWAN gateway allowing the integration of LoRa-based devices.

The gateway interacts with a cloud system operated by Neogrid (via a secure MQTT connection with mutual authentication 1.2 over TCP/IP). The cloud then provides a web-based API supporting the following:

- Management of the installation
- Online visualization of time series of signals and usage analyses
- Tagging of signals for adequate modelling and monitoring
- Management of user access to the data

Authorised users can access data on the cloud via an open web API based on JSON, which is documented here https://neogrid-technologies.gitlab.io/neogrid-api/. This API allows the following:

- Reading time series of measurements
- Reading of local weather data
- Sending setpoints to the controller (forwarded by the gateway to the BMS system, or regulated by the gateway itself via an on-board tuneable PID controller)

The system has been used for control in buildings around Denmark for more than 2 years. Currently, it operates data acquisition in over 1000 buildings in Denmark, and online control of space heating and domestic hot water in more than 600 of them. These buildings are mostly large apartment building blocks and single-family houses, with a limited number of schools, institution buildings and offices. In the vast majority of residential complex locations, the online control is operated commercially. The gateway and communication protocol support the collection of a large number of devices. Stable data collection has been demonstrated for over five years in a building with 1,100 measurement points recorded every minute via a single gateway connected to a BMS system.

Moreover, a toolbox for Matlab and Python has been developed, which allows fast usage of the system without extensive prior knowledge of web API management.

Neogrid's control of the heating system is typically based on operating the mixing loop and domestic hot water exchangers at central level and using feedback from the apartments in the building. This control is computed in the cloud. The corresponding setup is illustrated in ***Figure 27***.

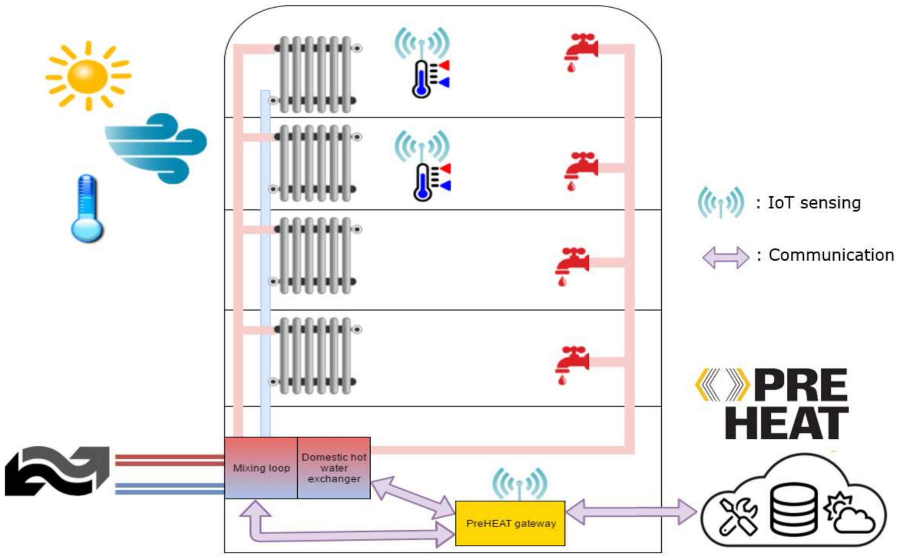
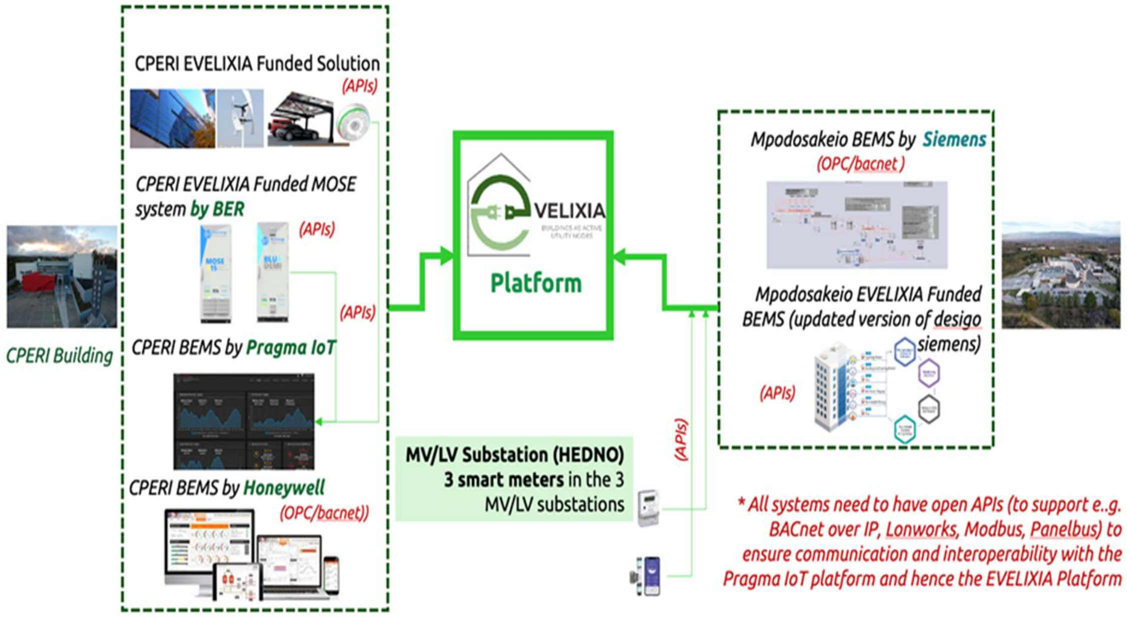


Figure 27. Structure of the setup in a building with district heating.

3.1.5 Greek PS Architecture

The architecture of the Greek Pilot Site (GR-PS) comprises a composite structure that integrates two distinct building entities with EVELIXIA platform, as depicted in ***Figure 28***. Each building -CPERI office building & Bodosakeio hospital- operates under its own monitoring and control system respectively. Both systems meet the necessary specifications and possess the required licenses (incl. free APIs) to ensure seamless connectivity with the EVELIXIA platform.



CPERI EVELIXIA Funded Solution
(APIs)
CPERI EVELIXIA Funded MOSE system by BER
(APIs)
(APIs)
CPERI BEMS by Pragma IoT
CPERI BEMS by Honeywell
(OPC/bacnet))
CPERI Building
EVELIXIA
Platform
MV/LV Substation (HEDNO)
3 smart meters in the 3 MV/LV substations
(APIs)
Mpodosakeio BEMS by Siemens
(OPC/bacnet)
Mpodosakeio EVELIXIA Funded BEMS (updated version of desigo siemens)
(APIs)
* All systems need to have open APIs (to support e.g. BACnet over IP, Lonworks, Modbus, Panelbus) to ensure communication and interoperability with the Pragma IoT platform and hence the EVELIXIA Platform

Figure 28. Greek PS architecture.

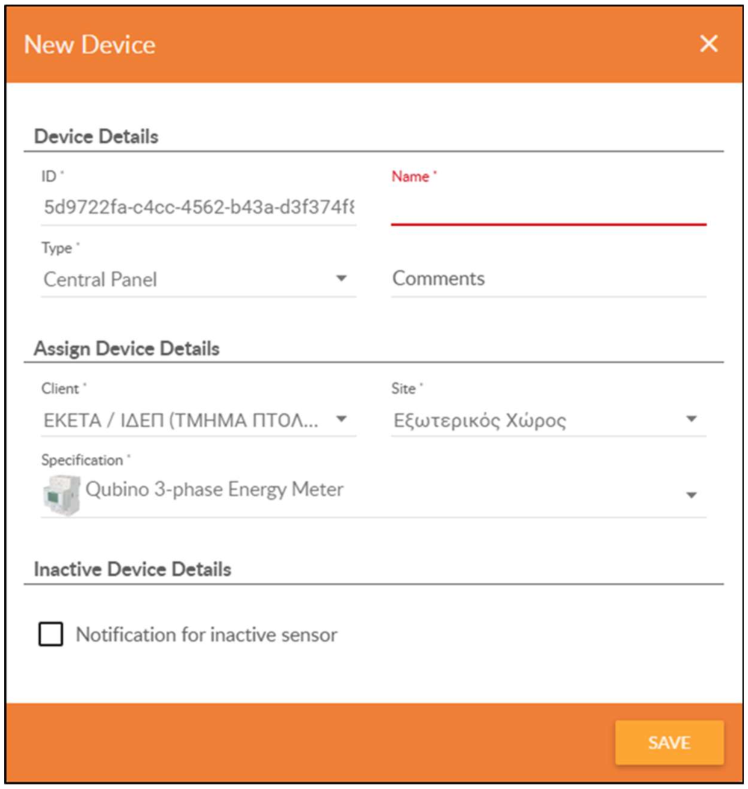
3.1.5.1 CPERI office building

The local data management and integration with EVELIXIA platform will be performed through the already installed PRAGMA IoT platform, which will be advanced to incorporate a dedicated device connector application, developed and deployed by CERTH.

PRAGMA IoT platform[1] implements an innovative IoT framework able to collect, analyse, compare, and present information collected from various nodes and sites. Its core features include the support of seamless data collection and aggregation, configuration according to user needs and preferences, user authentication, data-driven services, and automated analytics information flows. The latter is achieved through the use of dynamic dashboards and widgets.

The platform consists of the backend and the frontend (web-based user interface) part. The former provides a web services API for administration, user and asset management, the insertion of new data, and the retrieval of existing stored data within a specified time period.

After registering a user and adding a site/building, the addition of new devices to the system is critical to the process, as shown in Figure 29**,** since a device must be assigned to a specific user and site. PRAGMA IoT platform supports various device types (i.e., 3-phase energy meters, smart plugs, motion sensors, door/window sensors, indoor temperature & humidity sensors, CO_2 sensors, weather stations, etc.)


New Device

Device Details

ID *: 5d9722fa-c4cc-4562-b43a-d3f374f8

Name *:

Type *: Central Panel

Comments

Assign Device Details

Client *: ΕΚΕΤΑ / ΙΔΕΠ (ΤΜΗΜΑ ΠΤΟΛ...

Site *: Εξωτερικός Χώρος

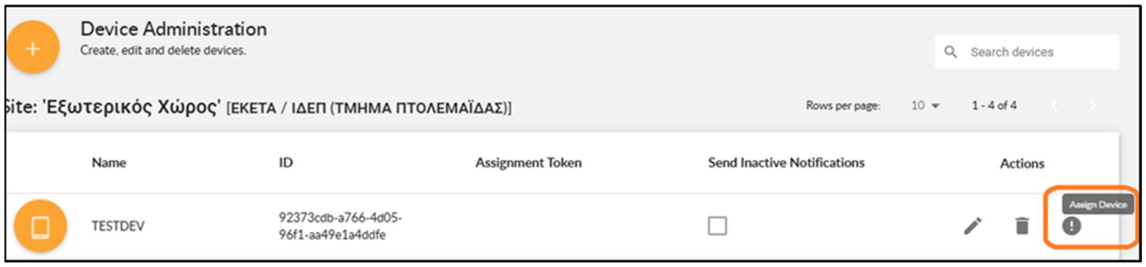
Specification *: Qubino 3-phase Energy Meter

Inactive Device Details

☐ Notification for inactive sensor

Figure 29: Addition of a new device on PRAGMA IoT platform

Following the addition of a new device (e.g. the planned installation of new smart sensor and control devices at the CPERI building), the assignment of a token to the device has to be made in order to be used later in data retrieval requests from the device. The token is a UUID (Universally Unique Identifier) string that is generated automatically by the PRAGMA IoT platform after pressing on the respective button on the User Interface (***Figure 30*** and ***Figure 31)***.


Device Administration

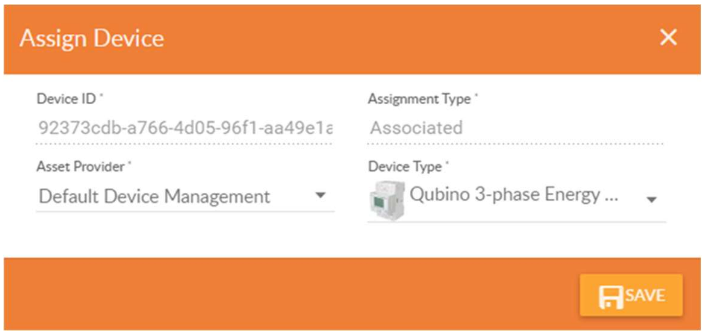
Create, edit and delete devices.

ite: 'Εξωτερικός Χώρος' [ΕΚΕΤΑ / ΙΔΕΠ (ΤΜΗΜΑ ΠΤΟΛΕΜΑΪΔΑΣ)]

Rows per page: 10 1 - 4 of 4

Name	ID	Assignment Token	Send Inactive Notifications	Actions
TESTDEV	92373cdb-a766-4d05-96f1-aa49e1a4ddfe		☐	

***Figure 30*. Assigning a token to a new device.**


Assign Device

Device ID *
92373cdb-a766-4d05-96f1-aa49e1a

Assignment Type *
Associated

Asset Provider *
Default Device Management

Device Type *
Qubino 3-phase Energy ...

Figure 31. Device assignment settings.

CERTH has developed a Device Connector module in Python programming language within the project, in order to facilitate the integration of devices with the PRAGMA IoT platform. The Device Connector allows the communication modules of devices to send their data to the PRAGMA IoT platform without being aware of the PRAGMA IoT platform's setup. This is because the Device Connector (i) transforms any data format sent by a device to the format that is required by the PRAGMA IoT platform, and (ii) makes the matching of device IDs.

The Device Connector supports different protocols, such as MQTT (Message Queuing Telemetry Transport) and Modbus TCP/IP to interact with devices. After making the proper transformations to the data format, it forwards the data (e.g. device measurements) to the PRAGMA IoT platform by using the platform's HTTP REST API (Application Programming Interface). Additionally, it is capable of sending commands to control devices by receiving them through specific MQTT topics. The Device Connector has been deployed at CERTH and is accompanied by a Mosquitto MQTT broker that has been deployed at the same location.

Implemented measures for security and data privacy:

PRAGMA IoT platform

- Use of HTTPS (Hypertext transfer protocol secure) protocol.
- Implementation of user authentication.
- Implementation of a token-based authentication mechanism: A pre-created token has to be provided as an authorization header each time a request is made by the caller.

Device connector

- Connection of clients to the MQTT broker by providing the correct username and password.

- Encrypted connections with the MQTT broker via TLS (Transport Layer Security), implemented through a provided certificate. The use of TLS ensures the confidentiality and integrity of information, preventing leakage and tampering. The one-way authentication approach has been adopted, i.e., the client authenticates the MQTT broker (server) when connecting.

[1] https://cperi.pragma-iot.com

3.1.5.2 Bodosakeio Hospital

The current BEMS is currently updated within the EVELIXIA framework with the latest version of Desigo CC (SIEMENS). Desigo CC is an open platform by design and supports a variety of standard protocols and interfaces for field network integrations. Furthermore, Desigo CC provides data to external applications and services. The evolution of Desigo CC is not only limited to Siemens developments as its Ecosystem developers provide fast, dynamic and creative enhancements and can exchange ideas, products and services based on customer or user demands.

Standard Protocols

Desigo CC communicates with field network devices to monitor and command information by using the following standard protocols:

- BACnet and BACnet/SC
- OPC DA (Data Access) and OPC UA (Unified Architecture)
- Modbus TCP
- SNMP
- KNX and KNX Secure
- S7, S7 PLUS and S7 secure communication
- IEC 61850

Subsystem Integration with SORIS

SORIS stands for Southbound Open RESTful Integration Service. It provides an open integration framework for Desigo CC that is easy to work with interoperability between computer systems on the internet: Developers can use the SORIS SDK to create SORIS adapters in Java or C# that map to the foreign system's protocol or interface. For security reasons, stage 1 adapters should be deployed locally on the

Desigo CC server or FEPs. They can be deployed on other Windows, Linux or embedded device hosts inside the intranet via secure https communication and a VPN.

OPC Server

Desigo CC allows Enterprise applications or other management systems to access real time values from integrated subsystems via OPC. The Desigo CC OPC Server supports the OPC DA (Data Access) specification.

BACnet Server

Desigo CC allows 3rd party applications or other management systems to access real time values from integrated subsystems via BACnet. The BACnet Server complements the northbound interfaces of Desigo CC, and it is used to expose Desigo CC data, over a BACnet/IP network.

The BACnet Server acts as a gateway for non-BACnet network devices integrated to Desigo CC, to be exposed over a BACnet IP network, as BACnet objects.

SNMP Agent

Desigo CC allows third-party Network Management Systems (NMS) to access real time values and statuses from integrated subsystems via SNMP. Desigo CC exposes data to an SNMP network.

The following main features are supported:

- SNMP protocol V2, and V3
- SNMP traps
- SNMP get

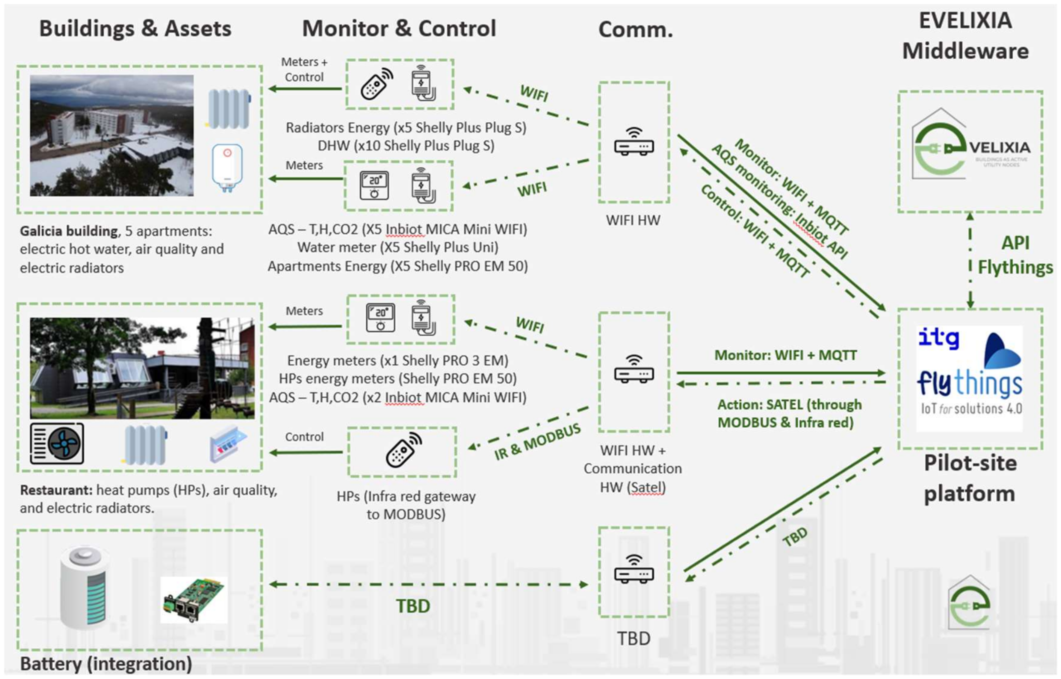
Certifications and Approvals

Desigo CC has been tested against a wide range of domain- and country-specific norms and standards, including:

- BACnet Revision 1.18, certified by BACnet Testing Laboratory as BACnet CrossDomain Advanced Workstation Software (B-XAWS)
- AMEV recommendation BACnet 2017 compliant with Management Operation Unit
- (MOU-B) profile
- IT security compliant with the ISA-99/IEC 62443-4-2 Security Level 2 (SL2)
- OPC DA V2.05a and V3.0 Server, certified by the OPC Foundation certification program

3.1.6 Spanish PS Architecture

The architecture of the Spanish PS is shown in ***Figure 32***.



Buildings & Assets
Monitor & Control
Comm.
EVELIXIA Middleware
Meters + Control
WIFI
Radiators Energy (x5 Shelly Plus Plug S)
DHW (x10 Shelly Plus Plug S)
Meters
WIFI
WIFI HW
Monitor: WIFI + MQTT
AQS monitoring: Inbiot API
Control: WIFI + MQTT
API Flythings
Galicia building, 5 apartments: electric hot water, air quality and electric radiators
AQS – T,H,CO2 (X5 Inbiot MICA Mini WIFI)
Water meter (X5 Shelly Plus Uni)
Apartments Energy (X5 Shelly PRO EM 50)
Meters
WIFI
Energy meters (x1 Shelly PRO 3 EM)
HPs energy meters (Shelly PRO EM 50)
AQS – T,H,CO2 (x2 Inbiot MICA Mini WIFI)
Monitor: WIFI + MQTT
Action: SATEL (through MODBUS & Infra red)
Control
IR & MODBUS
WIFI HW + Communication HW (Satel)
Restaurant: heat pumps (HPs), air quality, and electric radiators.
HPs (Infra red gateway to MODBUS)
Pilot-site platform
TBD
TBD
TBD
Battery (integration)

Figure 32. Spanish PS architecture.

Within the apartments, air quality sensors (Inbiot) will be monitored through a specific API that will be accessible in "*FlyThings*". Also, the whole apartment energy consumption (Shelly), space heating electric energy consumption (for radiators, Shelly) and electric DHW tanks energy consumption (for DHW tanks, Shelly) will be monitored within the apartments through WIFI and a web server for configuration and data visualization inside ITG's IoT platform "*FlyThings*". They will have control possibilities via MQTT and WIFI, except for the whole apartment energy consumption device, which won't allow control. Other additional water meters will be installed (Shelly, one for each water tanks system) without possibility of actuation but with the same monitoring process.

Within the restaurant, IoT devices will control and monitor the energy consumption (Shelly) through WIFI and a web server for configuration and data visualization inside "*FlyThings*". The reversible air-to-air heat pumps (Infrared gateway to MODBUS) will allow actuation with SATEL equipment via "*FlyThings*" whereas their monitoring will be done with Shelly equipment through WIFI and MQTT commands inside "*FlyThings*". Two air quality sensors (Inbiot) will be monitored through a specific API that will be accessible in "*FlyThings*".

The additional external battery ESS will be repaired to be monitored and controlled via "*FlyThings*". The communication between the battery ESS and "*FlyThings*" is still to be decided though.

ITG's IoT Platform has already an access for querying deployed assets through an API. A component to allow actuation on assets will be developed and included for the EVELIXIA platform to access datasets and actuate over assets through the same "*Flythings*" API.

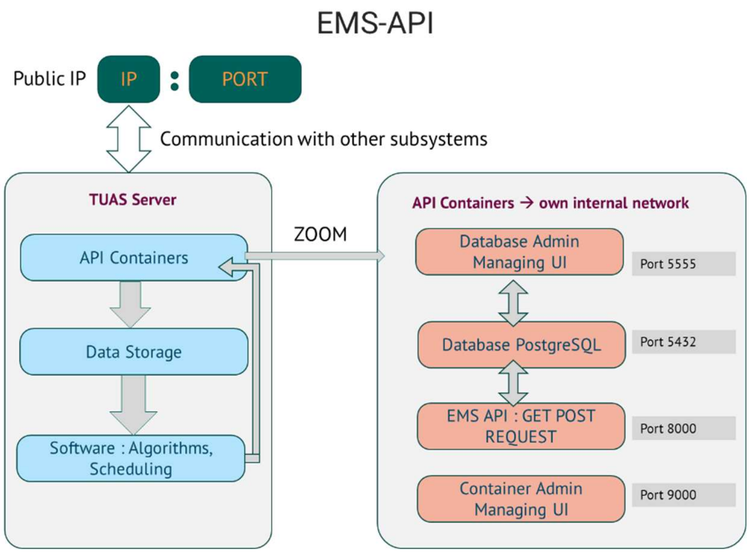
3.1.7 Finnish PS Architecture

3.1.7.1 PS Software Architecture

The Finnish PS system's architecture is based on FastAPI as the backend framework and is deployed on a TUAS server using Docker containers for modularity, scalability, and efficient resource management.

The system enables remote data collection from sensors that are not directly connected to the main network. Instead, edge nodes poll sensor data via Modbus (RTU/TCP) and transmit it securely using MQTT to a central broker hosted on the TUAS server, see **Edge Computers 3.1.7.2**. From there, the FastAPI application processes and stores the data in a TimeScale database, while additional forecasting and optimization modules analyse the collected information for advanced insights. To ensure smooth communication between services, the entire backend is containerized using Docker, with separate networks facilitating secure internal and external communications. The system is designed to be scalable, secure, and

efficient, making it ideal for automation, predictive maintenance, and real-time monitoring applications.
The following diagrams, as shown in ***Figure 33*** illustrate the detailed components, data flow, and networking infrastructure of this system:



EMS-API
Public IP
IP
PORT
Communication with other subsystems
TUAS Server
API Containers
Data Storage
Software : Algorithms, Scheduling
ZOOM
API Containers → own internal network
Database Admin Managing UI
Port 5555
Database PostgreSQL
Port 5432
EMS API : GET POST REQUEST
Port 8000
Container Admin Managing UI
Port 9000

Figure 33. Finnish EMS-API architecture.

3.1.7.2 Edge computers

The edge computing system serves as an intermediary between the EMS API (FastAPI backend) and data-acquiring devices, ensuring reliable and efficient data transmission. The edge computers, typically Raspberry Pi devices, are directly connected to Modbus-based sensors via RTU (serial) or TCP (network-based communication). Their primary functions include:

- *Polling Data from Modbus Devices* – A dedicated container continuously reads sensor data.
- *Local Storage* – The acquired data is stored on the device to prevent data loss in case of network interruptions.
- *Data Packaging and Transmission* – A second container groups sensor data into structured messages and sends them to the EMS API via MQTT, ensuring efficient communication. The edge computing setup is containerized using Docker, enabling flexibility and easy deployment. In the provided setup, two Docker containers run on each Raspberry Pi:
- Modbus Polling Container – Collects sensor data.

- Message Grouping & Transmission Container – Prepares and sends data via MQTT.

Edge devices architecture is depicted in ***Figure 34***.

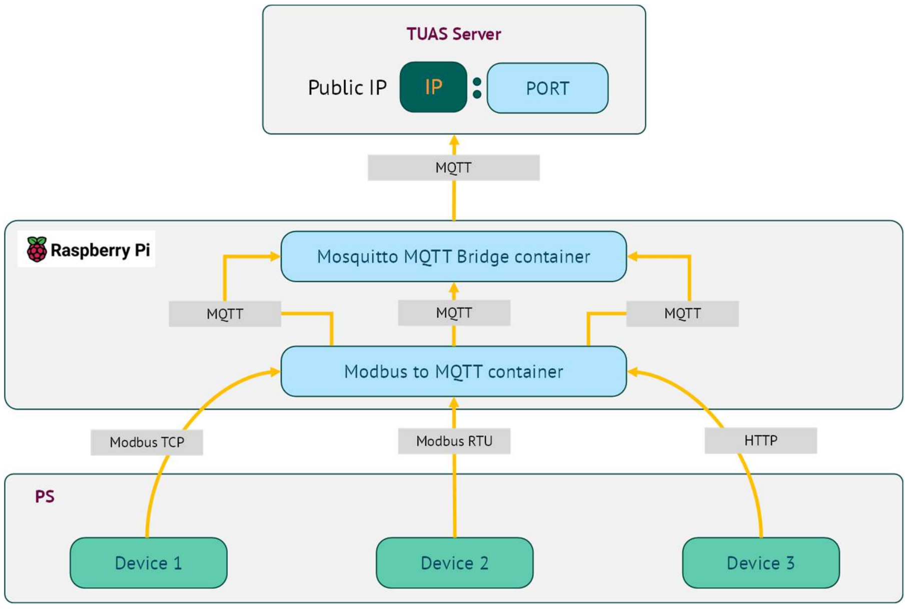


Figure 34. Edge devices architecture.

3.1.7.3 Software capabilities

The software developed is designed to monitor, evaluate, optimize, and control energy flow within microgrids. Its architecture operates continuously, managing multiple processes simultaneously. These processes are categorized into two main areas:

- **Data Services** – Handling data acquisition, monitoring, and management.
- **Control Services** – Processing data for forecasting, optimization, modelling, and scheduling.

Data Services

These services ensure smooth and reliable data collection, transmission, and storage, forming the backbone of the EMS. They allow real-time access to energy data from various sources, enabling better decision-making and system optimization.

Acquisition

The EMS is built to gather data from multiple sources, including energy meters, sensors, converters and web APIs. This data includes real-time energy consumption, environmental conditions (such as solar irradiance and temperature), and external information like electricity prices and weather forecasts. Data sources outside of the operated system include:

- *Electricity Prices* – Retrieved from the ENTSO-E Transparency Platform to analyse cost fluctuations.
- *Weather Forecasts* – Provided by Met Norway API to anticipate climate conditions affecting energy generation.
- *Solar PV Forecasts* – Sourced from the Finnish Meteorological Institute (FMI) to predict solar energy production.

Monitoring

The EMS continuously tracks energy consumption and generation across various sites, including pilot locations and remote installations. For remote monitoring, low-cost monitoring boxes were developed by TUAS to connect devices to the EMS API and transmit real-time data (**See Edge Computers 3.1.7.2**). These boxes, enclosed in electrical cases, require minimal local infrastructure—only power and internet connectivity.

The monitoring process follows two key steps, using industry-standard protocols:

Data Querying – EMS communicates with energy meters, inverters, and weather stations via Modbus TCP/RTU, HTTP, Dupline (Work in progress), widely used protocols for industrial automation. This ensures compatibility with a broad range of devices.

Data Transmission – Once collected, the data is converted into MQTT messages, a lightweight IoT protocol for fast communication. These messages are sent to the EMS server, where an MQTT broker processes and stores them in the database. In some cases, devices with built-in MQTT support can transmit data directly.

This real-time monitoring capability helps detect system irregularities, optimize energy use, and support decision-making for energy efficiency improvements.

Management

All data collected via the EMS API is stored in two synchronized databases:

- Main Database – Stores incoming data from all connected devices.
- Replica Database – Mirrors the main database to enhance security and improve API performance, ensuring smooth user access without overloading the main system.

Data access is controlled through three user levels:

- *Public* – Limited access to general information.
- *Building Owners* – Can monitor and manage their specific sites.
- Researchers – Authorized to analyse and extract data for advanced studies.

EMS also manages device interactions through dedicated tables, tracking operations for various components like heat pumps, renewable energy sources, and storage systems. Commands sent to devices are logged, including system responses and final execution status, ensuring supervision, transparency and control.

In schematic ***Figure 35***, it is shown how the system of Finnish PS gathers data from the local energy assets at the pilot site, from where it is sent to the TUAS EMS using MQTT. TUAS EMS will interface with the EVELIXIA southbound API, to provide the PS data and controllability to the upper system. The measurements are sent to the TUAS EMS nearly real time, but also saved locally, in case of system outages or malfunctions. Data gathering is not reliant on the EVELIXIA platform and is able to operate independently.

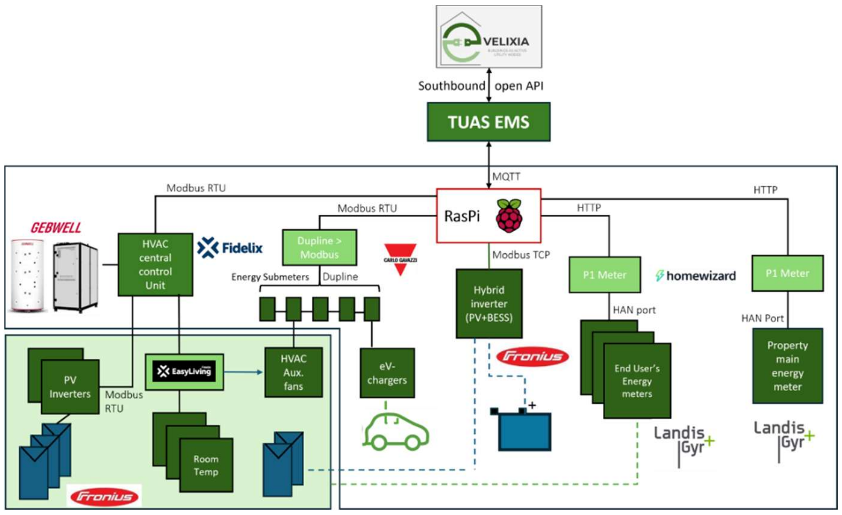


Figure 35. Finnish PS implementation.

Control Services

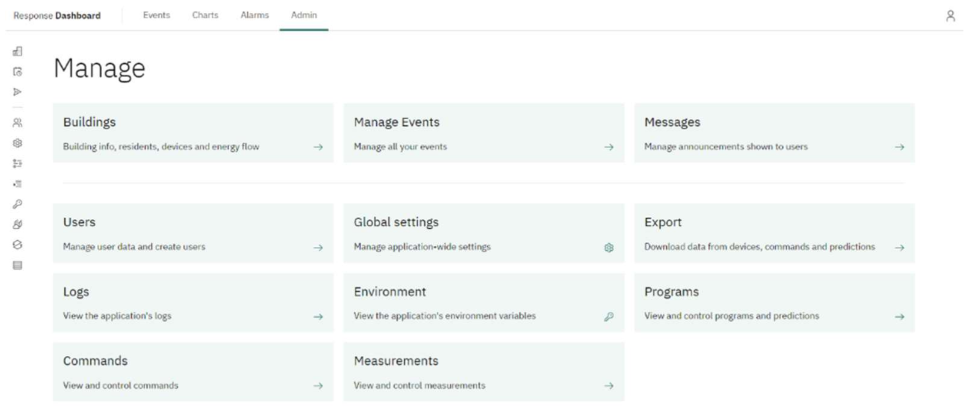
Control services process stored data to optimize energy management. These services enable:

- **Forecasting –** Predicting future energy consumption and renewable energy production.
- **Optimization –** Adjusting system operations to minimize costs and maximize efficiency.

- **Scheduling** – Automating energy distribution based on demand and pricing trends.

3.1.7.4 TUAS EMS UI

The EMS integrates a User interface (UI) key feature (Work in progress), enabling users to interact with integrated systems seamlessly. The landing page provides an overview of energy flows in buildings. Users can visualize configured systems, including energy meters, storage units, and renewable sources. Additionally, the system displays weather and irradiation forecasts from the Finnish Meteorological Institute (FMI), aiding in predicting renewable energy production. The EMS Interface supports real-time data visualization (up to one-minute resolution) for any sensor publishing data to its database. This feature helps owners track building events within a unified interface. Building owners and administrators (mainly Turku UAS developers) can configure new buildings and sensors via the UI, as shown in ***Figure 36***. Authorized users can edit, add, or delete buildings, devices, and sensors, as well as initiate programs like predictions and system optimization. ***Figure 37*** showcases available data, including next-day weather condition. With energy monitoring at resolutions as high as 10 seconds, the interface detects system abnormalities. For example, it identifies unexpected power peaks, enabling owners to take corrective action as needed.



Manage

Buildings
Building info, residents, devices and energy flow

Manage Events
Manage all your events

Messages
Manage announcements shown to users

Users
Manage user data and create users

Global settings
Manage application-wide settings

Export
Download data from devices, commands and predictions

Logs
View the application's logs

Environment
View the application's environment variables

Programs
View and control programs and predictions

Commands
View and control commands

Measurements
View and control measurements

Figure 36. EMS UI. Owner/administrator tab.

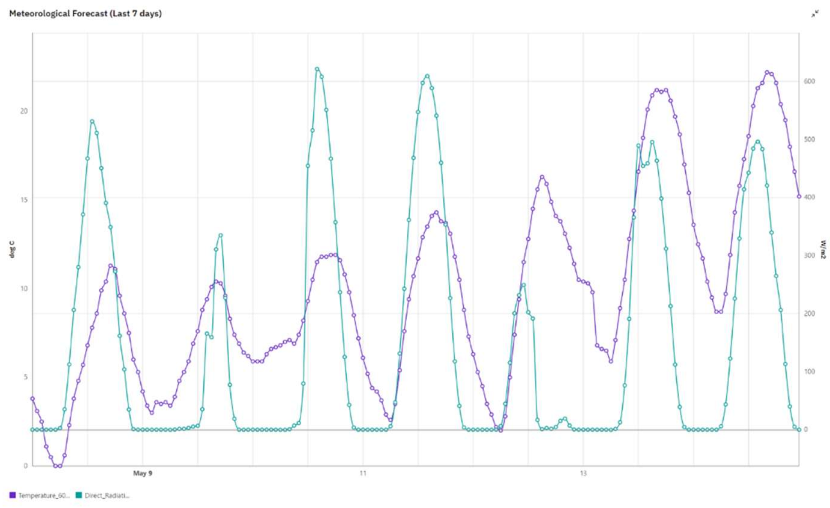


Figure 37. EMS UI. Weather forecast.

3.2 Data Model and API Development

This section will specify the data models and formats to ensure the appropriate quality and facilitate seamless interoperability and the respective API development on each PS.

3.2.1 Austrian PS Data Model and API Development

The Austrian PS API platform hosts two databases, one for the efficient timeseries-data collection (*InfluxDB*) and one for the additional metadata (*MongoDB*), compare ***Figure 23***. The metadata is structured on a data model, show in ***Figure 38***.

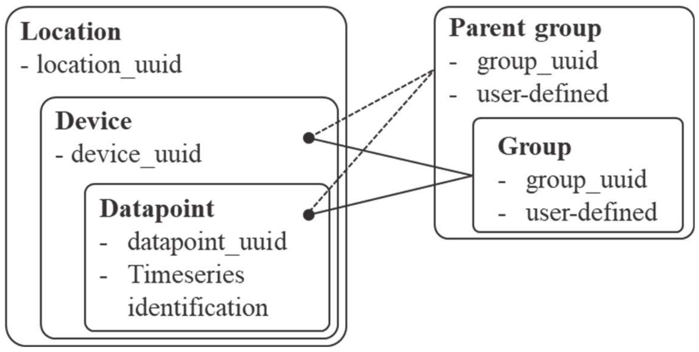


Figure 38. Austrian PS data model.

In its core, the lowest assignment level is called **datapoint** which is referenced with an individual uuid (Universal Unique Identifier) used to address the values over time in the timeseries database. Each datapoint originates from a **device** which is the hardware collecting or providing a single or multiple datapoints. The devices are installed at specific **location**, which are the buildings and energy assets participating in the Austrian PS. To allow for individual grouping and organization of devices and datapoints a user-defined **group** category is additionally defined. The groups can also be hierarchical structured into parent and child relationships. Devices, locations and groups are likewise identified with specific uuids automatically generated on creation. Addressing the platform by its API endpoint is based on the different uuids as shown in the example below with the HTTP GET command “status” for the latest value of a datapoint identified with its datapoint_uuid, in this case a temperature sensor value.

```
curl -X 'GET' \
  'https://api.hochschule-burgenland.org/datapoints/DATAPOINT_UUID/status' \
  -H 'accept: application/json' \
  -H 'x-api-key: YOURAPIKEY'
```

The successful response is given in JSON format and may contain the following structure.

```
{
  "time": "2025-01-29T12:32:30.087312+00:00",
  "value": 20.4,
  "unit": "°C"
}
```

Additional metadata for datapoints, device, locations and groups, like plain names, units, descriptions, hardware modules, origin and parent groups can be accessed by specific additional commands. The comprehensive documentation of the API endpoints with all available GET, PUT, POST and DELETE commands was provided to the relevant EVELIXIA project partners.

3.2.2 Romanian PS Data Model and API Development

The data model in the context of the Romanian PS will be a custom defined one, as the data is being pooled from multiple systems some amount of message pre-processing is being done before the data is being inserted into the local database for ease of collecting data. The physical parameters will be stored as columns in one or multiple tables as necessary.

The REST API endpoint is being developed on the OpenAPI Specification template which will provide JSON messages as responses to queries.

A standard payload, specifically from the already installed sensors, may contain the following:

```
{
“Timestamp”:17326277049,
“illuminance_1”:146,
“illuminance_2”:165,
“proximity_1”:0,
“proximity_2”:0,
“temperature_1”:22.2,
“temperature_2”:23.6
...
}
```

Where each message part represents one physical reading from one of the sensors as part of the Senzors_data message.

3.2.3 French PS Data Model and API Development

The data model to store the measurements from the energy assets will follow the Sun Spec Information Model Reference [3]. The measurements are first stored locally into a Time-Series database, a NoSQL database that is optimized for Time-Series data with high write and query performance. The database technology is ideal for monitoring, IoT, and real-time analytics.

The data will be structured into:

- **Measurements:** name of the data collection
- **Tags:** indexed metadata, description of the data stored in measurements
- **Field:** non indexed data with actual data value

Figure 39 represents the data storage model. The SignalName is based on SupSpec information model reference, while the SignalTypes are custom defined by the demonstrator partner.

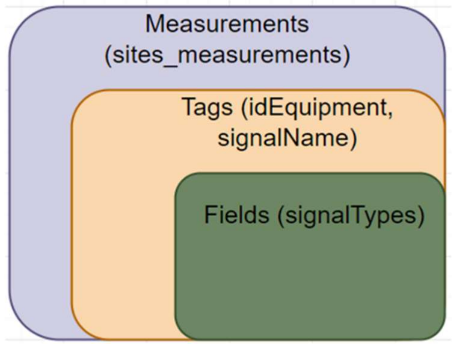

Measurements (sites_measurements)
Tags (idEquipment, signalName)
Fields (signalTypes)

Figure 39. French PS data model.

Table 9 represents an example of the battery data model from SunSpec information model reference.

Table 9. SunSpec information model.

Description	SignalName	SignalType
Active Power	W	ACTIVE_POWER
Reactive Power	Var	REACTIVE_POWER
DC Current	A	DC_CURRENT
DC Voltage	V	DC_VOLTAGE
SOC	SoC	SOC_SOC
SOH	SoH	SOH_SOH

As to API development, The API facilitates real-time data exchange for equipment measurements in storage plants, using a REST architecture developed with Django. Currently, it operates internally with private authentication, offering JSON responses based on the Sunspec data model. For instance, querying a site's equipment measurements returns structured data, such as:

```
{
 "site": {
  "idSite": "10",
  "values": [
   {"time": "2024-11-30T23:00:00Z",
    "idEquipment": "5",
    "signalName": "W",
    "signalType": "ACTIVE_POWER",
    "value": 0.0
   },
  ...
  ]
 }
}
```

3.2.4 Danish PS Data Model and API Development

The Danish PS uses a public API, fully documented in https://neogrid-technologies.gitlab.io/neogrid-api/.

Some of the relevant steps and features using the API are:

- Gaining access to the API (key)
- Authorisation
- API methods
- Get all locations
- Get location with building model
- Get devices for a location
- Get measurements
- Get measurements using the building model
- Get earliest/latest measurements using the building model
- Get weather forecast
- Send a setpoint schedule for control
- Get setpoint schedule
- Get indoor temperature setpoint
- Get price components
- Get price data
- Get comfort profile setpoints
- Status of the services

In order to use the API within Evelixia you should get an API key and need to know which location ids you should interact with. This can be supplied by Neogrid. On the same time, you will also have access to Neogrid's app where all data can be visualized, and analysis can be performed. Then it's straightforward to load relevant data for a building.

The API is two ways, so sending schedules and adjust settings is also possible. It depends on what is supported by the building and what your role is.

3.2.5 Greek PS Data Model and API Development

3.2.5.1 PRAGMA IoT

The custom data models are utilised by the PRAGMA IoT platform. The data models are represented and stored in JSON format. Specific devices can send their measurement data to the Device Connector instead of sending them directly to the PRAGMAT IoT platform. In this case, the Device Connector will transform the payload sent by the device to the appropriate JSON format that is required by the PRAGMA IoT platform. An example is shown in the next two JSON structures, which employ a 3-phase meter (Modicon M262), installed at different locations at the Greek PS by HEDNO. The first one is the power meter device raw data JSON, while the second one is the measurements JSON format that is acceptable by the PRAGMA IoT platform.

```
{
 "Measurements_from_T300": {
  "ModiconM262": "TM262L10MESE8T",
  "timestamp": "2025-01-14T16:30:15"
 },
 "Values": {
   "Analog_Measurements": {
     "Voltage_1": 238.38,
     "Voltage_2": 236.70,
     "Voltage_3": 238.25,
     "I1": 10.20,
     "I2": 11.428,
     "I3": 12.64,
     "Frequency": 49.980,
     "Real_Power_P": 5968.503,
     "Reactive_Power_Q": -5427.585,
     "Apparent_Power_S": 8067.323,
     "Power_Factor": 0.73983
   }
 }
}
```

```
{
  "eventDate": "2025-01-14T16:30:15+0000",
  "updateState": true,
  "measurements": {
    "VL1": 238.38,
    "VL2": 236.7,
    "VL3": 238.25,
    "AL1": 10.2,
    "AL2": 11.428,
    "AL3": 12.64,
    "PTOTAL": 5968.503,
    "Frequency": 49.98,
    "Reactive_Power_Q": -5427.585,
    "Apparent_Power_S": 8067.323,
    "Power_Factor": 0.73983
  }
}
```

PRAGMA IoT HTTP REST API specifications

The insertion of new measurement data of a particular device to the PRAGMA IoT platform is performed via two discrete methods.

Method 1: HTTP POST requests

The query path is:

https://baseURL/pragma/api/assignments/{device-assignment-token}/measurements?tenantAuthToken=xxxxxx

The following three headers shall be included in the request:

```
"Content-Type": "application/json; charset=utf-8"

"Access-Control-Allow-Origin": "*"

"Authorization": <token>
```

The JSON format of the body to be sent must have a structure similar to the following.

```
{
  "eventDate": "2025-02-05T12:00:00+0000",
  "updateState": true,
  "measurements": {
    "measurementName1": 1,
    "measurementName2": 1
  }
}
```

Method 2: MQTT protocol messages (publish to MQTT topic)

The JSON format of the MQTT message to be sent must have a structure similar to the following.

```
{
  "hardwareId": "xxxx",
  "type": "DeviceMeasurements",
  "request": {
    "measurements": {
      "measurementName1": 1,
      "measurementName2": 1
    },
    "updateState": true,
    "eventDate": "2025-02-05T12:00:00+0000"
  }
}
```

The Device Connector utilises the first method for interconnection with the PRAGMA IoT platform.

The retrieval of all measurement data stored in the PRAGMA IoT platform is achieved by performing HTTP GET requests on the device level, each with relative required input parameters, as shown in ***Table 10***.

Table 10. Data retrieving Query from the PRAGMA IoT platform.

Property	Description
Request type	HTTP GET
Headers	Authorization: “<authentication token>” X-Sitewhere-Tenant: “<user token>”
Path structure	https://baseURL/pragma/api/assignments/{device-assignment-token}/measurements/series?page=1&pageSize=30000&startDate={date-time}&endDate={date-time}&measurementIds={metric name}
Parameters	device assignment token: the device assignment token string startDate: start date time of measurements in UTC, e.g. 2025-01-30T00:00:00.000Z endDate: end date time of measurements in UTC, e.g. 2025-01-31T23:59:59.000Z measurementIds: the measurement of interest, e.g. “power”. Note: Devices can provide more than one measurement. This query parameter can be omitted in order to get the data for all supported measurements.

The received response is a JSON array containing the time series data from the PRAGMA IoT platform. As shown below, each entry consists of the datetime and the corresponding value.

```
[
  {
    "measurementId": "power",
    "entries": [
      {"value": 581.4,
        "measurementDate": "2024-01-30T00:01:40.861+0000"
      },
      {"value": 585.7,
        "measurementDate": "2024-01-30T00:16:40.850+0000"
      },
      {"value": 582.3,
        "measurementDate": "2024-01-30T00:31:40.925+0000"
      },
      {"value": 581.1,
        "measurementDate": "2024-01-30T00:46:40.896+0000"
      },
      {"value": 581.3,
        "measurementDate": "2024-01-30T01:01:40.923+0000"
      },
      {"value": 583.2,
        "measurementDate": "2024-01-30T01:16:40.926+0000"
      },
      { "value": 582.2,
        "measurementDate": "2024-01-30T01:31:40.930+0000"
      }
    ]
  }
]
```

The PRAGMA IoT platform allows getting devices' configuration and installation information, including metadata, by making a GET HTTP request to:
https://baseURL/pragma/api/devices/?site={site-uuid}&page=1&pageSize=1000&includeDeleted=false&excludeAssigned=false&includeSpecification=true&includeAssignment=true

A JSON with all the information per each available device at the given site is returned as a response. For each device, various properties are returned, such as the ID, name, type, associated building name and ID, specs/measures populated by the device including their measurement units, and others.

3.2.5.2 Desigo CC

Web Services with NORIS

NORIS stands for Northbound Open RESTful Integration Service. Desigo CC allows external applications to read and write real-time data as well as access events or historical values, by using the provided REST (Representational State Transfer) web service interfaces, for example NORIS. Web Services can be used for applications such as Enterprise Software, Energy Management services, or Facility Management systems. EVELIXIA platform is expected to utilise the above web services to retrieve the data from GR-PS.

3.2.6 Spanish PS Data Model and API Development

The sensor data and asset status will be stored in ITG's IoT platform database, "*Flythings*". The EVELIXIA platform will be able to access the PS's data using the "*Flythings*" API. The API component for querying deployed assets is already available. The API component for managing assets to be deployed will be developed in 2025.

"*Flythings*" is a platform with its own optimised data model inspired by the Sensor Observation Service (SOS) of the OGC. The platform is optimised for dynamic scenarios, being able to handle both, simple sensors and complex devices with multiple inputs and outputs. Observations from sensors and complex devices are stored in time-series databases, reducing performance and scalability issues associated to traditional structured databases.

The platform allows to map uniformly and effortlessly multiple components such as sensors, actuators and communication modules within the same entity. The platform adopts multi-format compatibility, counting on JSON as the main format used, improving overall performance and reducing the weight of data transmissions. The platform does not need the use of geographic datasets, including the mobile devices and those without a fixed location within the range of use.

Regarding the API:

- It is based on RESTful principles, allowing an easy user-platform interaction. These principles guided the design of operations for device creation, observation publishing, and historical data querying. These designs and principles ensure an efficient use of standard HTTP resources like GET, POST, PUT, and DELETE.

- Support from WebSockets enable continuous real-time transmission of data from sensors. Additionally, integration with low-capacity devices is possible due to the MQTT protocol implementation.
- The API adopts OAuth 2.0 as an authentication mechanism, complemented by support for JWT (JSON Web Tokens). This ensures that only authorized users and devices can interact with the platform, in this case, the EVELIXIA platform. All communication is protected via HTTPS, ensuring data integrity and confidentiality.

The new feature developed for the API will also enable actuation on the devices that are meant to undergo control actions.

The API is accessible through the following URL:

https://api.flythings.io/apidocs/index.html, where the main elements of the data model with its new adaptations can be observed, as well as the data integration requests offered by the system.

3.2.7 Finnish PS Data Model and API Development

3.2.7.1 Finnish data models

In TUAS EMS, data access is controlled through three user levels:

- Public – Limited access to general information.
- Building Owners – Can create, monitor and manage their specific sites.
- Researchers/Developers – Authorized to analyse and extract data for advanced studies, add pilot sites and in general full configuration of the EMS itself.

The relational database used (TimeScaleDB) was built to allow the addition of unlimited users, devices and sensors and commands for executing special tasks. This database is composed as follows:

1. Users
2. Sensors
3. Measurements
4. Programs
5. Predictions
6. Devices

7. Heat Pump
8. RES
9. Converter
10. Storage
11. Commands
12. CommandsResponse
13. CommandsResults
14. APschedulerJobs

EMS also manages device interactions through dedicated tables, tracking operations for various components like heat pumps, renewable energy sources, and storage systems. Commands sent to devices are logged, including system responses and final execution status, ensuring transparency and control. The relational database used (TimeScaleDB) was built to allow the addition of unlimited users, devices and sensors and commands for executing special tasks. This database is composed as follows:

Figure 40 shows the MAIN TABLES.

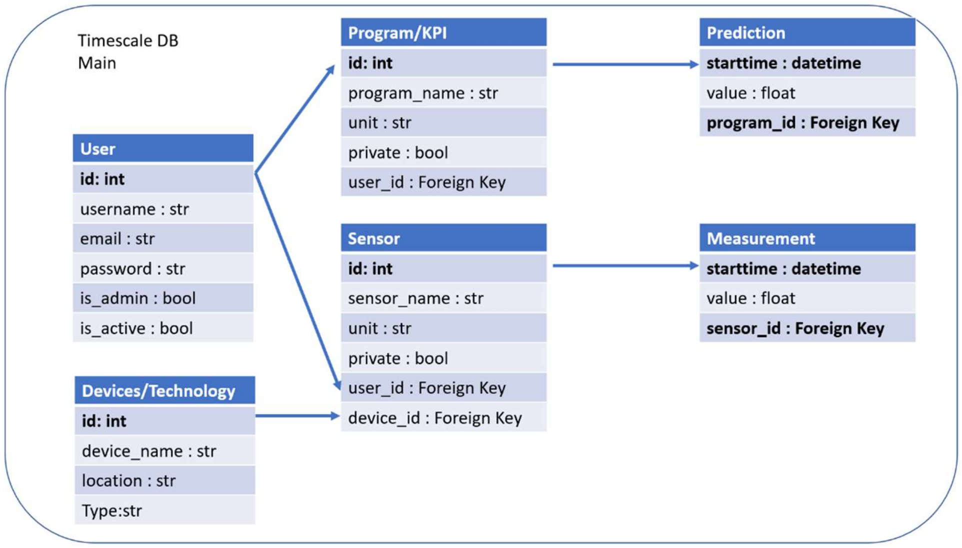


Figure 40. EMS Main DB description.

Figure 41 shows the DEVICES.

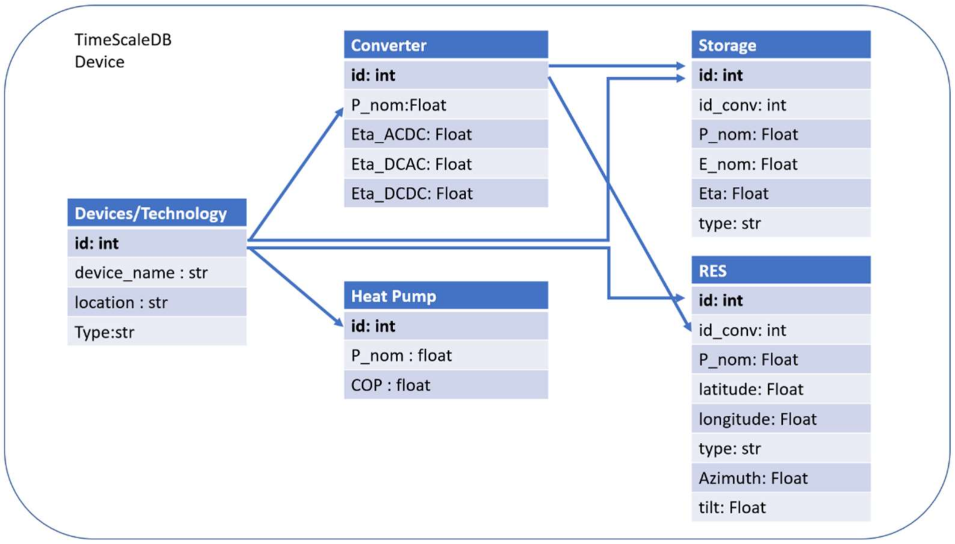


Figure 41. EMS Devices/Technology DB description.

Figure 42 shows the COMMANDS.

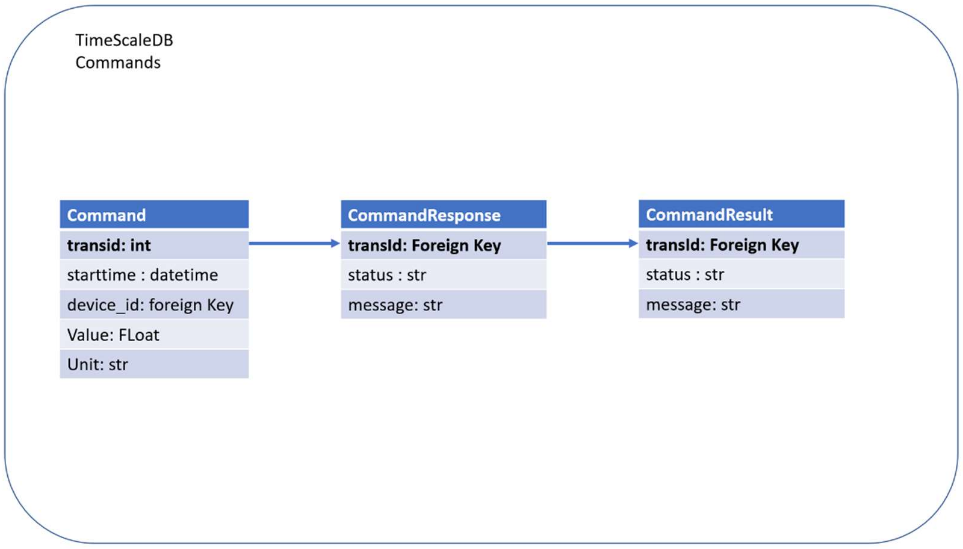


Figure 42. EMS Commands DB description.

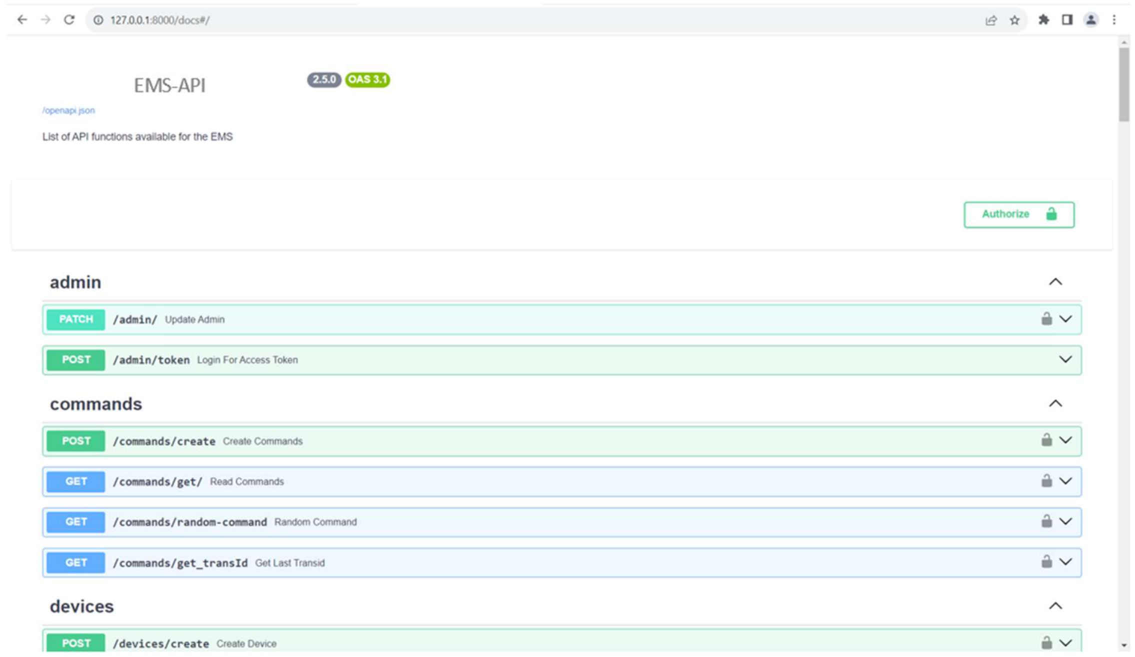
The attributes forming the primary key are highlighted in bold and the arrows represent the relationships between each database.

3.2.7.2 Finnish EMS-API

In case of EMS-API, as shown in ***Figure 43***, the followings are the different functions:

- Data acquisition system from various sensors (Implemented and working)
- Data acquisition from 3rd parts: electricity prices, weather forecast (Implemented and working)
- Electrical system optimization (Implemented -Tested, work in progress)
- Sending commands to assets for controlling (BESS) (Implemented and tested at lab conditions, pending for implementation in Pilot Site)

The main API function panel is accessible through the following link:
http://193.166.139.12/api/docs
All endpoints have a prefix http://193.166.139.12/api/
The server running the API has not important CPU computation capabilities, therefore, heavy request (10k+ datapoints) might take 10-20 seconds before being returned.


127.0.0.1:8000/docs#/

EMS-API 2.5.0 OAS 3.1
/openapi.json
List of API functions available for the EMS

Authorize

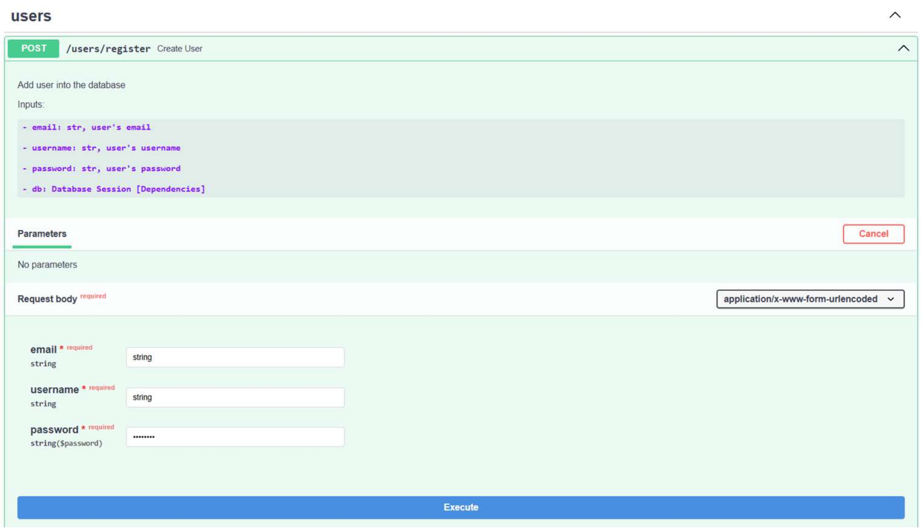
admin
PATCH /admin/ Update Admin
POST /admin/token Login For Access Token

commands
POST /commands/create Create Commands
GET /commands/get/ Read Commands
GET /commands/random-command Random Command
GET /commands/get_transId Get Last Transid

devices
POST /devices/create Create Device

Figure 43. **Fast API doc Web GUI.**

This webpage lists all the endpoints with this API. As an example, the first endpoint to create a user will be:
http://193.166.139.12/api/docs#/users/create_user_users_register_post
Through the main webpage, we can use and test every function of the API.
Figure 44 shows an example when creating a user.


users

POST /users/register Create User

Add user into the database

Inputs:

```
- email: str, user's email
- username: str, user's username
- password: str, user's password
- db: Database Session [Dependencies]
```

Parameters

No parameters

Request body required — application/x-www-form-urlencoded

email * required `string`	string
username * required `string`	string
password * required `string($password)`	••••••••

Figure 44. **Creating a user with Swagger UI.**

After filling the requested body + executed, check if the response has been successful.

The Methods:

- POST Method is to add data in DB (Rights needed)
- GET Method is to read data from DB
- PATCH Method is to update data in DB (Just Admins)
- DELETE Method is to delete data from DB (Just Admins)

Interacting with the API

At TUAS, Python is the recommended programming language for interacting with the API. To send HTTP requests, its id first needed to install the requests module. If your **pip** is being used, install it with the command:

```
>>> pip install requests
```

Making API Requests

To use the API, we need to identify the URL where the API functionality is accessible. Then, send an appropriate request body using a Python dictionary. This ensures that the API receives the correct data format for processing.

Authentication

Certain API actions require authentication. Before making such requests, we must first obtain an authentication token by calling the authentication endpoint. ***Figure 45*** provides an example function to obtain a token.

```
TOKEN_URL = f"{BASE_URL}/users/token"
user_data = {
   "username": username,
   "password": password
}
r_token = requests.post(TOKEN_URL, data = user_data)
token = r_token.json()['access_token']
r_headers = {"Authorization": f'Bearer {token}'}
```

Figure 45. **Example function to get tokens.**

Once you have the token, include it in the request headers for subsequent API calls. For example, when creating a new device in the system, the token must be passed in the headers. ***Figure 46*** demonstrates an example function to create a program in the database using the API.

```
# 1) Register the Program and add it in the Database (Must be done only once per
program_name)
url_program = f"{BASE_URL}/programs/register"

data_program = {
   "program_name": "Test_program",  # To be changed
   "unit": "something",
   "private": True,  # Boolean, can be False also. This defines who accesses data from
this program.
}

prgm_post = requests.post(url = url_program, params = data_program, headers =
r_headers)
```

Figure 46. **Example function to create a program in the database.**

Handling API Responses

The response content for each request can be accessed in two formats. If you need a string output, use .text. If you need the response as a Python dictionary, use .json(). These methods allow you to process the returned data efficiently.

```
# Get Data from predictions
url_get_predictions = f"{BASE_URL}/predictions/get/"
get_data = {
   "download": "no",
   "starttime": "2023-09-01 00:00:00",
   "endtime": "2023-09-02 00:00:00",
   "program": ["Spot_price",  data_program["program_name"] ]# List of
program_name, to be adapted
}

r_get_pred = requests.get(url=url_get_predictions, params=get_data,
headers=r_headers)

print(r_get_pred.json())

# Get RESPONSE Format {'program_name_1': {'starttime': ['2023-01-
01T00:00:00+00:00', ...],
#                                  'unit': 'program_name_1_unit',
#                                  'value': [87, ...,]},
#                   'program_name_2': {'starttime': ['2023-01-02T00:00:00+00:00', ...],
#                                  'unit': 'str',
#                                  'value': [789.598, ...],}}
```

Figure 47. **Example Function to get predictions from different programs.**

For instance, ***Figure 47*** illustrates an example function that retrieves predictions from the API endpoint /predictions/get. The response from this GET request, formatted as JSON, is shown in ***Figure 48*** (r_get_pred.json()).

```
{'Test_program_3': {'starttime': ['2023-09-01T07:48:00+00:00', '2023-09-
01T07:50:00+00:00', '2023-09-01T08:01:27.774400+00:00'],
                'unit': 'something',
                'value': [184.0, 184.0, 127.0]},
 'Test_program_7': {'starttime': ['2023-09-01T07:39:00+00:00', '2023-09-
01T07:50:00+00:00'],
                'unit': 'something',
                'value': [187.0, 154.0]}}
```

Figure 48. **r_get_pred.json() response from GET Requests.**

3.3 APIs Integration and Testing

The integration and testing of APIs are critical steps to ensure seamless data exchange and interoperability across all PS. Initial testing has already begun at two PS, the Danish and Austrian PS, focusing on validating the reliability and performance of the developed APIs in real-world scenarios. These tests involve verifying secure data transmission, assessing bidirectional communication functionality, and ensuring compatibility with existing systems. The insights and results from these preliminary tests will guide further refinements and optimizations.

The remaining PS are scheduled to begin their integration and testing phases following the completion of these initial evaluations. This phased approach ensures a streamlined deployment process, leveraging the lessons learned from early testing to address potential challenges proactively. Additionally, it allows the adaptation and tailoring API implementations to the unique requirements of each PS, ensuring a reliable API framework. By iteratively addressing challenges and incorporating lessons learned, the project aims to progressively enhance the APIs' effectiveness and scalability across all PS.

4 CONCLUSIONS

This deliverable marks a significant milestone in advancing EVELIXIA's goals by laying the foundation for secure, scalable, and interoperable systems across all PS. The development of APIs and hypervisor solutions has successfully addressed the immediate needs of local data collection and bidirectional communication, demonstrating alignment with EVELIXIA's roadmaps. These outcomes contribute directly to EVELIXIA's goals of enabling robust communication channels and fostering cross-platform integration.

A critical analysis of the results highlights the effectiveness of the methodologies employed, particularly the phases testing approach at PS. Early testing has provided valuable insights into system performance, reliability, and potential challenges, ensuring that subsequent implementations will benefit from these learnings. Although significant strides have been made, the ultimate objective of achieving seamless integration across all PS requires sustained effort, particularly in addressing site-specific requirements and maintaining interoperability.

The next steps involve extending integration and testing efforts to the remaining PS (French, Romanian, Spanish, and Finnish) and refining APIs based on feedback and results from initial implementations. Interactions with other WP and stakeholders will be essential to ensure coherence and alignment with the broader EVELIXIA framework. This includes advancing optimization algorithms for the three specific PS, enhancing interoperability with the middleware layer, and incorporating emerging insights to improve system scalability and resilience. By maintaining a proactive and adaptive approach, the project is well-positioned to achieve its long-term objectives and deliver impactful results.

5 REFERENCES

[1] F. Agha Kassab, R. Rodriguez, B. Celik, F. Locment, and M. Sechilariu, ‘A Comprehensive Review of Sizing and Energy Management Strategies for Optimal Planning of Microgrids with PV and Other Renewable Integration’, *Appl. Sci.*, vol. 14, no. 22, Art. no. 22, Jan. 2024, doi: 10.3390/app142210479.

[2] ‘France Announces New FIT Rates For PV Systems Up To 500 KW - News’. Accessed: Jan. 14, 2025. [Online]. Available: https://www.dsneg.com/news/france-announces-new-fit-rates-for-pv-systems-76985776.html

[3] ‘SunSpec Information Model Reference - SunSpec Alliance’. Accessed: Jan. 14, 2025. [Online]. Available: https://sunspec.org/sunspec-information-model-reference/

6 ANNEXES

6.1 Annex 1

In this annex, KPIs results of the five simulations are presented in ***Table 11.***

Table 11. Annual KPIs results of five simulations for E-Factory.

KPIs	Weight self-consumption = 1	Weight self-consumption = 0.75	Weight self-consumption = 0.5	Weight self-consumption = 0.25	Weight self-consumption = 0
Net revenues (k€)	85.47	88.62	143.85	371.43	376.96
Revenues (k€)	331.22	334.19	391.90	666.31	667.130
Grid costs (k€)	245.75	245.56	248.04	294.87	290.16
Savings self-consumption (k€)	168.77	168.47	164.22	133.93	132.94
Grid total energy (MWh)	132.19	132.17	130.35	99.16	97.22
Grid consumption (MWh)	180.57	180.99	187.59	196.49	194.67
Grid injection (MWh)	48.38	48.82	57.25	97.33	97.45
Grid-to-load consumption (MWh)	114.64	114.12	117.38	194.08	192.63
PV-to-load consumption (MWh)	131.85	131.64	138.16	131.49	131.38
PV self-consumption (MWh)	181.03	180.59	172.17	132.08	131.96
Self-consumption rate (%)	78.91	78.72	75.05	57.57	57.52
Self-sufficiency rate (%)	78.91	78.72	75.05	57.57	40.25
Grid dependency (%)	46	45.9	43.78	40.01	40.25
ESS dependency (%)	54	54.1	56.22	59.99	59.75
FC dependency (%)	25.07	25.39	22	0.54	0.54
ELY energy (MWh)	133.8	135.6	117.2	2.8	2.8
ESS nb cycles equivalent	1.22	1.14	1.42	0.05	0.02
PV curtailment (%)	4.95	4.66	5.79	0.34	0.19