CHESTER

Compressed Heat Energy Storage for Energy from Renewable sources

Smart Energy Management System for efficient operation of the CHEST system

PROJECT	**CHESTER**
PROJECT NO.	**764042**
DELIVERABLE NO.	**D4.6**
DOCUMENT VERSION	**1.0**
DOCUMENT PREPARATION DATE	**20/08/2021**
RESPONSIBILE PARTNER	**AIGUASOL**
DISSEMINATION LEVEL	**Public**

Type of Deliverable		
R	Document, Report	**x**
DEM	Demonstrator, pilot, prototype	
DEC	Websites, patent fillings, videos, etc.	
OTHER		
ETHICS	Ethics requirements	
ORDP	Open Research Data Pilot	

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 764042.

EC Grant Agreement	No.764042
Project Acronym	CHESTER
Project Title	Compressed Heat Energy Storage for Energy from Renewable sources
Programme	HORIZON 2020
Start Date of Project	01-04-2018
Duration	48 Months

Financial/Administrative Coordinator	
Project Coordinator Organization Name	TECNALIA
Address	Parque Tecnológico de Bizkaia C/Astondo, Edificio 700 (Spain)
Phone Numbers	+34 946 430 850
E-mail Address	nora.fernandez@tecnalia.com
Project web-site	www.chester-project.eu

Version Management			
Filename		D4.6: Smart Energy Management System for efficient operation of the CHEST system	
Author(s)		D.Vaz/A. Carrera	
Reviewed by		S. Stark/D. Bestenlehner	
Approved by		M. Epelde	
Revision No.	**Date**	**Author**	**Modification description**
RV 1	20-08-2021	D.Vaz/A. Carrera	First version of the document
RV 2	16-09-2021	S. Stark D. Bestenlehner	Review of deliverable
RV 3	26-10-2021	D.Vaz/A.Carrera	Updated with review inputs
RV 4			

Contents

List of Figures

List of Tables

Glossary, abbreviations and acronyms

ANN	Artificial Neural Network
API	Application Programming Interface
ARIMA	Auto-regressive Integrated Moving Average
DB	Database
CF	Capacity Factor
CHEST	Compressed Heat Energy Storage
CHP	Combined Heat and Power system
COP	Coefficient of Performance
DAM	Day Ahead Market
DH	District Heating (network)
DSO	Distribution System Operator
FTP	File Transfer Protocol
GUI	Graphical User Interface
GWP	Global Warming Potential
HX	Heat eXchanger
HP	Heat Pump
HTTP	Hyper-text Transfer Protocol
KNN	K- Nearest Neighbour
KPI	Key Performance Indicator
MPC	Model Predictive Control
NBP	Normal Boiling Point
NEMO	Nominated Electricity Market Operator
ODP	Ozone Depletion Potential
ORC	Organic Rankine Cycle
P2P	Power to Power ratio
PCM	Phase Change Material

PHS	Pumped Hydro Storage
PV	Photovoltaic
REST	Representational State Transfer
RR	Restoration Reserve
RTU	Remote Terminal Unit
SCADA	Supervisory Control And Data Acquisition
SEMS	Smart Energy Management System
SoC	State of Charge
ST	Solar Thermal
TSDB	Time Series Database

1. Introduction

1.1. Executive Summary

This deliverable presents the architecture, composition and data communications processes of the automatized Smart Energy Management System (SEMS) developed for the virtual CHEST system of Aalborg and Ispaster case studies.

SEMS consists of several independent modules:

- **Execution module**, which is the module responsible for the integration and execution of the several SEMS' components;
- **Forecasting module**, that generates forecasts of some key-variables, based on forecasting algorithms and analytical calculus;
- **CHEST model**, responsible for simulating the behaviour of the virtual CHEST systems; a dedicated API to request data from the model was also developed;
- **Optimizer**, that, working iteratively with the CHEST model, returns the optimal operation strategy for the CHEST system;
- **DB module**, a multi-level database that stores all data collected and generated in the overall SEMS operation;
- **KPI calculator**, that calculates CHEST KPIs, based on the data generated by the CHEST model
- **Visualization platform**, that consists of SEMS' user interface for data visualization.

Even though all modules are isolated from each other, all of them (with the exception of the optimizer that works directly with the CHEST model), communicate with the DB module. The DB module was placed at the centre of the architecture to assure all data is properly retained and to keep track of the data flow from and to all modules.

SEMS relies on archive data that were manually sent to the DB module, but also on data that is regularly requested to external providers via API. Namely, the electricity prices of the day-ahead market of Spain, provided by the DSO that operates in Spain, and a set of weather conditions from Aalborg and Ispaster, from a weather data provider, Meteoblue.

Due to data availability and data coherence issues, SEMS was developed and tested with archive energy demand and electricity prices data. However, in order to properly evaluate its operation and impact, the smart system operates dynamically, retrieving and treating the old datapoints as new data.

The architecture based in microservices executed in Docker containers was tested and shows a reliable and efficient performance. The configuration adopted has proven to be robust and offers the additional advantage of having a high scalability and adaptability potential.

1.2. Purpose and Scope

This deliverable presents the work carried out on Task 4.4: Development of control algorithms and Smart Energy Management System (SEMS). SEMS is responsible for the efficient operation

of the CHEST system. It computes the optimal operation strategy for the CHEST system, at each moment, through an optimization module. The optimization process consists of a machine learning procedure that relies on the CHEST model developed in Task 4.2, on predictive algorithms and monitored data from the energy system CHEST is integrated into (electrical grid and/or district heating network). SEMS also calculates KPIs defined in T4.3. All results can be visualized by users through an interface that allows to visualize the operation of the CHEST system, also developed under the scope of T4.4. All modules and tools that compose the SEMS platform are described in this document, as well as the integration processes that connect them into a functional platform.

1.3. Structure of the document

The present document is structured as follows:

- Chapter 2 gives a brief description of SEMS to the reader and the context and conditions in which it is applied.
- Chapter 3 describes the overall digital architecture of SEMS.
- Chapter 4 explains the data communication processes between different modules and its integration in SEMS.
- Chapter 5 describes each module that composes the infrastructure.
- Chapter 6 presents the tests performed to verify the correct functioning of the overall infrastructure described in the previous sections.

1.4. Relations with other deliverables

SEMS development involves the application of tools, information and knowledge developed and acquired in previous tasks and described in the corresponding deliverables. Thus, D4.6. and Task 4.4 are closely related with several other Tasks and Deliverables:

- D2.1, written in the scope of Task 2.1, gives detailed information about the case studies (pilots), the energy systems, energy demands and boundaries of all project pilots and indicates the two of them that are the focus of further analysis and optimization in Task 4.4. The information about the case studies is paramount for the overall conception of SEMS, for the development of the optimization algorithms and the formulation of the forecast methods used by SEMS.
- D4.1 reports the work developed in Task 4.1: it presents the structure of the monitoring and control platform developed for the project pilot cases and describes data forecasting processes deployed for variable prediction that were later refined in Task 4.4 and applied to SEMS. The data collected by the system detailed in D4.1. is essential for SEMS functioning, so the monitoring platform it describes was integrated in SEMS. Furthermore, the user interface platform created to meet the purposes of Task 4.1 was also integrated in SEMS and further developed to work as SEMS visualization platform.
- D4.2 describes the advanced dynamic model of the CHEST system, developed in Task 4.2, one of the main components that integrates SEMS. D4.2 also states the format of market integration of the CHEST system, defining thus the variables that have to be considered in SEMS management process.

- D4.4 describes the analysis of the control strategies for the CHEST system connected to the energy networks, setting the ground for optimization strategies applied in SEMS.
- D4.5 defines KPIs as well as the integration of the CHEST system into the energy system of each case study and their corresponding business case.

2. Overview

This section offers an introductory explanation of SEMS' basic principles of operation, and presents the case studies SEMS was developed for and tested with.

2.1. Brief description of SEMS

SEMS computes the optimized operating conditions of a virtual CHEST system, in real-time, based on predictive algorithms. SEMS can be categorized as a Model Predictive Control (MPC) tool. MPC algorithms firstly started being employed by the chemical industry in the decade of 1980, but in the last years, as their application became more common, MPCs spread to other areas, in particular, to power systems management applications.

MPCs are smart control mechanisms that optimize processes based on information generated by predictive models. Figure 1 presents a scheme of the general information flow of an MPC. MPCs are composed of two main pieces: a dynamic model and an optimizer. The model uses input data to better represent complex processes and iteratively calculate emulated results of the process with optimized control parameters. The optimized parameters are generated by the optimizer respecting some given constraints and aiming for minimizing a cost function.

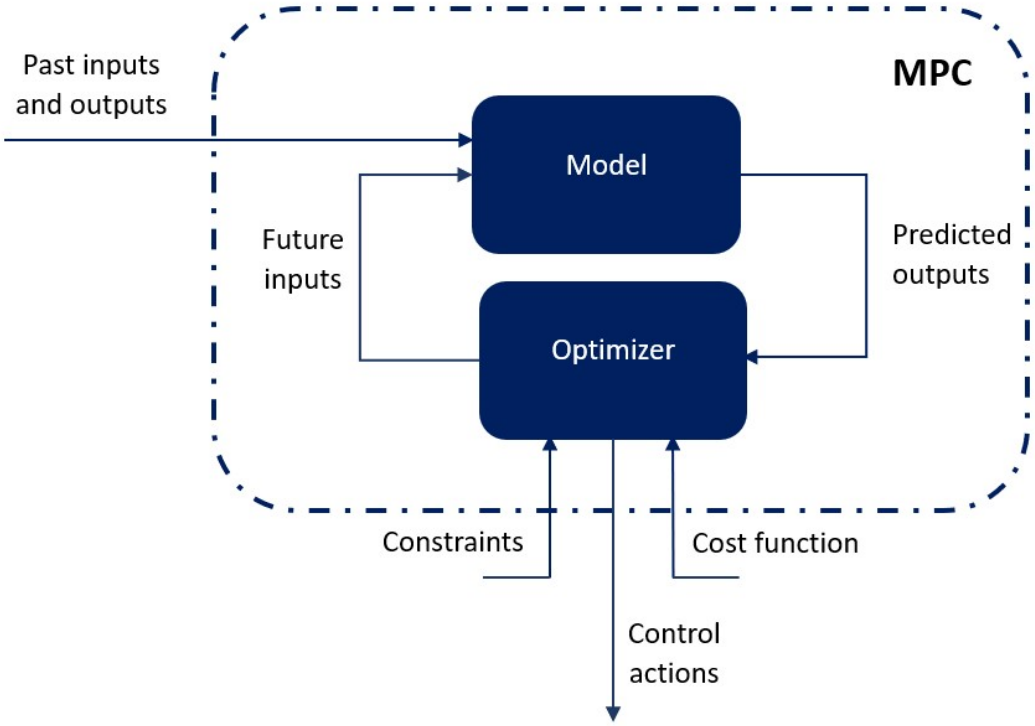

Figure 1: MPC graphical sketch

The parameters generated by the optimizer are sent by the MPC to the control system, not represented in the figure above, that automatically implements them.

SEMS will provide the operation strategy of the CHEST system at each moment. Thus, in its particular case, a layer of artificial intelligence capacities is also added to the predictive control system. The input data that feeds the model includes the forecasted values of energy demand and electricity prices that will influence the model's results and condition the decisions made by the optimizer. The forecasting process parses data and learns from that data to generate more precise predictions. Ultimately, these machine learning methods are fundamental to find optimal control parameters based on empirical data.

The development of each of the elements described, from the type of input data to the formulation of the optimizer, depends on the MPC's application object and its surrounding conditions. The case studies in which SEMS development was based on, and their relevant characteristics are described in the next section.

2.2. Case studies

In the scope of WP2, two of the seven pilots of the project were selected to be used as case studies for WP4's tasks for further analysis and optimization: Aalborg, in Denmark, and Ispaster, in Spain. SEMS and its optimization strategies were designed taking into account the characteristics of these two case studies and their energy systems. Detailed information about the pilots can be found in D2.1.

The selected pilots have in common the availability of real, monitored data, either if collected in real-time, as is the case of Ispaster, or stored from previous years (see D4.1 for a more detailed description of the available information and completeness), as it happens for Aalborg. These data are of utmost importance for the system optimization aimed at in the context of T4.4.

Aside that, the two selected case studies are very different in terms of scale and context. This distinction between the pilots allows to explore SEMS implementation and its optimization potential applied to two different contexts and work modes of the CHEST system.

Aalborg is a large-scale pilot with the DH network of the city at its kernel. The DH network, that is responsible for tackling a heat demand of 2000 GWh/year, takes up heat from several excess heat sources which can be beneficial for the integration of the CHEST system. Aalborg's case study explores the integration of a CHEST system that aims for obtaining revenues by offering grid services, operating as an electrical storage device for the electrical grid, coupled to the DH system. In this case study, the market prices highly influence the external controlling signals set by SEMS.

Ispaster case study, on the other hand, consists of an (almost) isolated micro-grid of heat and electricity. Ispaster aims for obtaining integral energy independence and resilience. This case study allows to explore CHEST application to small island systems, as an electrical storage system, exploring its potential as replacement of the electrical batteries that integrate the energy network of the small town, necessary to get a relevant share of renewables. In this case, CHEST integration does not consider the connection to an external energy market. SEMS' control signals are based on the mismatch between the energy demand profile and energy generation profiles.

3. SEMS Architecture

In this section SEMS architecture and information flow is described, and the modules that compose the overall system and their integration and deployment methods are explained.

3.1. Overview of SEMS architecture

SEMS is composed of a set of independent modules and tools integrated in a global infrastructure. Figure 2 shows a simplified schematization of SEMS' architecture with its general components and the information flow.

The dark blue shapes of the image represent the modules purposely created for CHESTER. It includes the CHEST model and the Optimization module, that correspond to the essential components of an MPC, as explained in the previous section; and the forecasting module.

The orange rectangles represent the modules that were developed and/or adapted from existing tools. The DB module, that consists of SEMS' database, can be found in this category. It is placed at the centre of SEMS architecture. Thus, all information used or generated by the other modules is centralized, allowing to oversee all data traffic and quickly identify data-flow-related issue. This architecture option also enables to increase communication efficiency by avoiding concatenated delays as could occur in a chain communication configuration.

The visualization platform was also created based on an existing platform. Even though the visualization platform is not an intrinsic part of SEMS, it is crucial to explore and analyse SEMS results and the benefits its implementation leads to, since it is the means for CHEST users to consult all information related with its operation. SEMS visualization platform provides users with indicators of CHEST's performance at each moment, such as the energy balance of the system, charging/discharging mode and operation costs, as well as with the resulting energy, economic and environmental benefits generated by the CHEST system. It is though this component that CHEST's performance can be analysed and that potential of heat-electrical storage integration of the CHEST depending on site conditions can be assessed.

The light blue shape corresponds to existing platforms that were merely integrated in the infrastructure. It includes data streamed from external data providers and data collected by the monitoring systems implemented in the pilot sites that measure and record metrics on the operation conditions of the existing assets. It shall be noted that the information received by the DB module from these systems is not necessarily in real-time.

At last, the greyish shapes represent components of the system, that would integrate a real system but are not integrated in the developed infrastructure, given the virtual character of the developed CHEST system. Since there is no physical CHEST facility, the control module that would deploy the decisions made by SEMS was not developed.

In a physical CHEST, SEMS would be implemented into the control system, the optimized operation strategy generated would be directly applied to the CHEST system and the resulting operation conditions would be read by the monitoring system and displayed in the visualization platform along with the conditions calculated by CHEST digital twin, the CHEST model. As this is not the case, the visualization platform will only exhibit the conditions of the emulated CHEST system. The control parameters that SEMS would implement in a physical CHEST are stored in the DB module and plotted in the visualization platform.

SEMS was developed resorting to a variety of tools upon which its modules were built and the communication processes between them were configured. Its architecture is mainly based on Dockerized services which guarantees the automatization of a complex but robust, reliable and scalable structure. Most of the modules and communication processes were developed in Python, an interpreted, general-purpose, object-oriented, high-level programming language with dynamic semantics and a procedural paradigm.

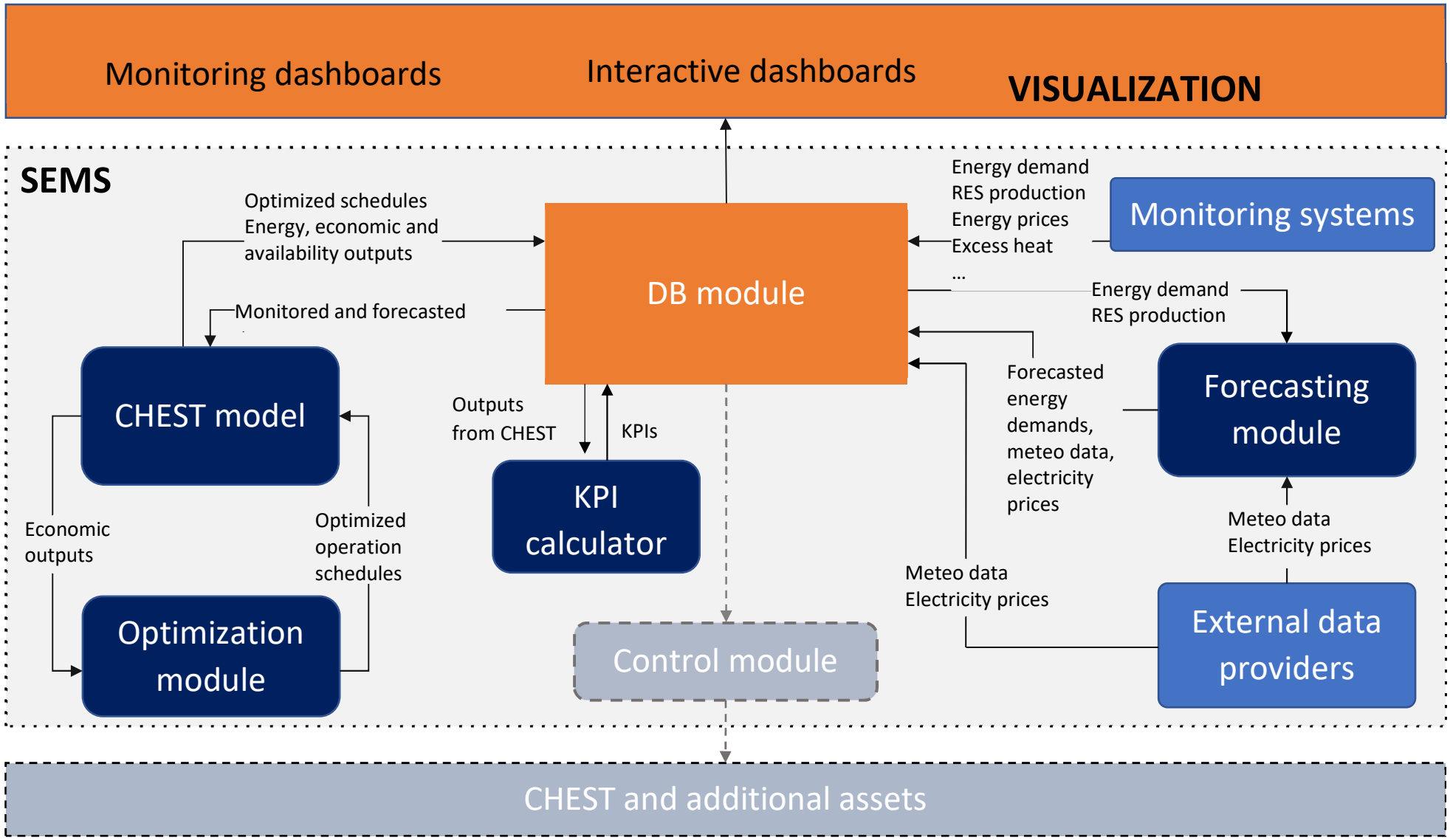

Figure 2: SEMS architecture and data flow schematization

4. Tools and modules integration

This chapter presents the integration of the several tools and modules developed for SEMS into an integrated structure, and the communication processes between those independent elements.

4.1. Execution module

Even though, the execution module (as the name infers) is a module, it is presented in this particular section due to its key-role in the integration of several modules into a unified structure. The execution module is the commands operation centre of SEMS. It was implemented in Python and it is responsible for making requests to the different modules and to check if all data is arriving to the DB module with the proper timing. The execution module has been deployed as a Docker containerized microservice with Kubernetes orchestration that uses a cronjob (jobs that follow a specific schedule and failure handling procedure) to execute the microservice every hour.

The execution module microservice operates as follows:

1. Every hour, the module requests the execution of the forecasting module that will collect real data (electricity price, heat demand, weather conditions) up to the current moment from the database module. The forecasting module generates forecasts for the next 24 hours. Once the forecasting module calculation ends, the results are sent to the DB module.
2. Then, the execution module checks if forecast data for the next 24 hours is indeed available in the database, as well as the monitoring data of the energy system and the data from external providers and,
3. When confirmed, sends a request to the model and optimizer API. CHEST model retrieves real and forecasted data from the DB module and iteratively calls the optimizer that, based on the economic outputs from the model, returns a charge/discharge operation strategy for the next 24 hours, until achieving an optimal result. Once the optimization process ends, the results are stored in the DB module.
4. Finally, the execution module requests the execution of the KPI calculator that, based on the output data sent to the DB Module by the CHEST Model, produces KPI results of the CHEST system and sends them to the DB Module.

4.2. Model API

To allow the integration of the CHEST model with the other modules, an HTTP based application programming interface (API) was developed. Almost all the CHEST Model features can be independently accessed via the API: it is possible to retrieve the operation conditions of the CHEST components, economic outputs or availability data, as well as the operation strategy the model requests to the optimizer. The integration of the model with SEMS is prepared to request via the API all outputs generated by the model and the optimizer.

The CHEST model API is a .NETCoreApp (v2.1) application, prepared to deal with an unlimited number of requests from a user that was implemented in C# via Microsoft Visual Studio 2017 environment along with Microsoft.Net Framework 4.8.

The API resorts to the following libraries:

- Microsoft.AspNetCore.App version 2.1.1
- Microsoft.NETCore.App version 2.1.0
- Microsoft.VisualStudio.Web.CodeGeneration.Design version 2.1.1

Even though the CHEST model HTTP API is essentially a RESTful API, in order to provide a more flexible access to clients that cannot adhere to strict REST protocols, the API provides alternative access through various overrides. Thus, the default data exchange format is JSON, but the API request allows different formats to send and receive data, through access with pluggable formatters. This feature enables a higher access flexibility to the model that may be useful for querying outside the SEMS scope or adaptations to the model integration in the overall system.

To protect the access to the CHEST model, an API etoken is required to access the Model API. The API etoken can also track API requests.

Standard HTTP response codes are used for all returned results and errors. Table 1 and Table 2 present the output response codes of successful requests and unsuccessful requests, respectively, and the corresponding description.

Table 1: Successful output codes of the CHEST model API

Code	Description
200	The request completed successfully.
204	The server has completed the request successfully but is not returning content in the body. This is primarily used for storing data points as it is not necessary to return data to the request caller.
301	This may be used in the event that an API call has migrated or should be forwarded to another server.

Table 2: Error output codes of the CHEST model API.

Code	Description
400	The request could not be understood by the server due to incorrect syntax. The client SHOULD NOT repeat the request without modifications.
401	.Indicates that the request requires user authentication information. The client MAY repeat the request with a suitable Authorization header field
404	The requested endpoint or file was not found. This is usually related to the static file endpoint.
405	The requested verb or method was not allowed. Please see the documentation for the endpoint you are attempting to access.

406	The request could not generate a response in the format specified.
408	The request has timed out. This may be due to a timeout fetching data from the underlying storage system or other issues.
413	The results returned from a query may be too large for the server's buffers to handle.
500	An internal error occurred within the model.
501	The requested feature has not been implemented yet. This may appear with formatters or when calling methods that depend on plugins.
503	A temporary overload has occurred. Check with other users/applications that are interacting with the model and determine if you need to reduce requests or scale your system.

In the case an error occurs, the API returns a response that includes a valid HTTP status error code and a content body with error details.

4.3. External Data microservice

SEMS relies on data from external provider, namely, meteorological data and electricity prices. This data is directly used by the CHEST model, on one side. On the other side, up-to-date meteorological data and electricity prices are also necessary for the forecast module to generate 24-hour-ahead forecasts that are fed to the CHEST model.

The external data microservice is a Dockerized service that has been deployed to gather this information from external data providers. The external data microservice has been configured to be executed every day and access the API services of the respective providers to request the aforementioned data and store it in the DB module.

The electricity prices for the Spanish day-ahead and balancing markets are retrieved from REE (Red Eléctrica de España) [1], the DSO in Spain. REE provides an API that offers a REST service to allow third parties to access the prices from the different electricity markets.

The meteorological data is collected from Meteoblue weather API [2]. Meteoblue is a commercial weather data provider that offers up-to-date local weather information for any location on Earth (land or ocean) from coordinates, including weather forecasts.

The following data is requested to Meteoblue weather API:

- ambient air temperature [ºC]
- relative humidity [%]
- global horizontal irradiance [W/m^2]
- wind speed [m/s]
- wind direction [º]
- wind gusts [m/s]
- total precipitation [mm]

From the following locations:

Aalborg:

- latitude: 57.046707 º
- longitude: 9.916667 º

Ispaster

- latitude: 43.362778 º
- longitude: -2.543056 º

4.4. Archive data

SEMS implementation and demonstration also resort to archive data, namely:

- Monitored operation conditions of Aalborg DH network and corresponding demands, from 2016;
- Prices from the Danish electricity market, from 2019;
- Monitored operation conditions of Ispaster energy network and assets, from 2019.

As explained in D4.1, the monitoring system of the DH network of Aalborg is not integrated with SEMS infrastructure due to proprietary and confidentiality issues. Although, for testing and demonstration purposes, data from a 1-year period of monitoring was transferred by FTP (File Transfer Protocol) from the DH central station, stored in a .csv format file and posteriorly sent to the DB module through the database's own REST API. Local data collection from sensors and meters is carried out through a SCADA system.

Since the external controlling signals generated by SEMS are highly dependent on the market prices in the Aalborg case-study, the electricity prices considered for SEMS' operation are the ones that correspond to the same period as of the available monitored data. Thus, electricity prices from 2019 were manually downloaded from the electricity provider's website and, once again, sent to the DB module with the REST API of the timeseries database.

Data from energy mini-grid of Ispaster is collected and sent to the database in real-time. The metrics are read and collected by Remote Terminal Units (RTU), physically connected to the equipment, and sent via Modbus to the central SCADA station. The SCADA station communicates with the database API that converts and loads the received data to the DB module. Since the beginning of the project, Ispaster's DH network has experienced modifications: the demand is now considerably higher and additional boilers were added to the network. The production values and operation conditions of these new elements are not being monitored; thus, the current conditions of the energy network cannot be modelled and the available real-time monitoring values of the system could not be used to deploy and test SEMS. For that reason, SEMS variation for the Ispaster case study was developed based on data of the energy network of Ispaster from 2019. SEMS operation applied to Ispaster case study is not affected by the energy markets, therefore, there was no need to use archive electricity prices data.

For further details about the monitoring systems of both pilots D4.1 shall be consulted.

5. Tools and modules description

The following chapters describe the tools and modules that compose SEMS. Some of SEMS' elements are already thoroughly explained in other deliverables. Hence, the present document describes the modules that were not yet described in other documents, reports applied updates on formulation and configuration, and offers complementary information, but mostly, provides the description of these elements from a data integration and communication flow perspective.

5.1. CHEST model

CHEST model is a complete and advanced dynamic model of the CHEST system and its interaction with the energy system it is integrated into. The model has been developed in TRNSYS [3] an extensible simulation environment for the transient simulation of systems with an open and modular structure. The whole system model is composed of more than 60 TRNSYS components and couples a C++ refrigerant properties database, CoolProp [4]. Due to requirements upon which this tool was developed and the complexity of the overall system, the CHEST model has a high computational cost.

For that reason, a physical server was purposely installed in Aiguasol's offices to host the model and the Virtual Machine was the Dockerised environment where the model is deployed.

The Dockerised environment that hosts the CHEST model is composed by a swarm with several containers. The Docker swarm was sized according to the expected number of interactions of the CHEST model with the rest of the modules, so that it has the capacity to respond to the necessary number of requests. The established security protocols allow to safely access the model remotely.

The CHEST model obtains the boundary conditions, such as the electricity prices, energy demands, meteorological conditions, both historical and forecasted for an upcoming time period from the DB module. Then, the model generates the operation variables of an emulated CHEST for each case study (such as the electricity produced, the energy consumed from the grid, etc.), as well as the resulting energy and economic outputs associated with the system where it is virtually implemented. The economic results are retrieved by the Optimization Module that calculates the optimal operation strategy for the given conditions and sends it to the model for recalculation.

The model, represented in Figure 3, is thoroughly described in D4.2.

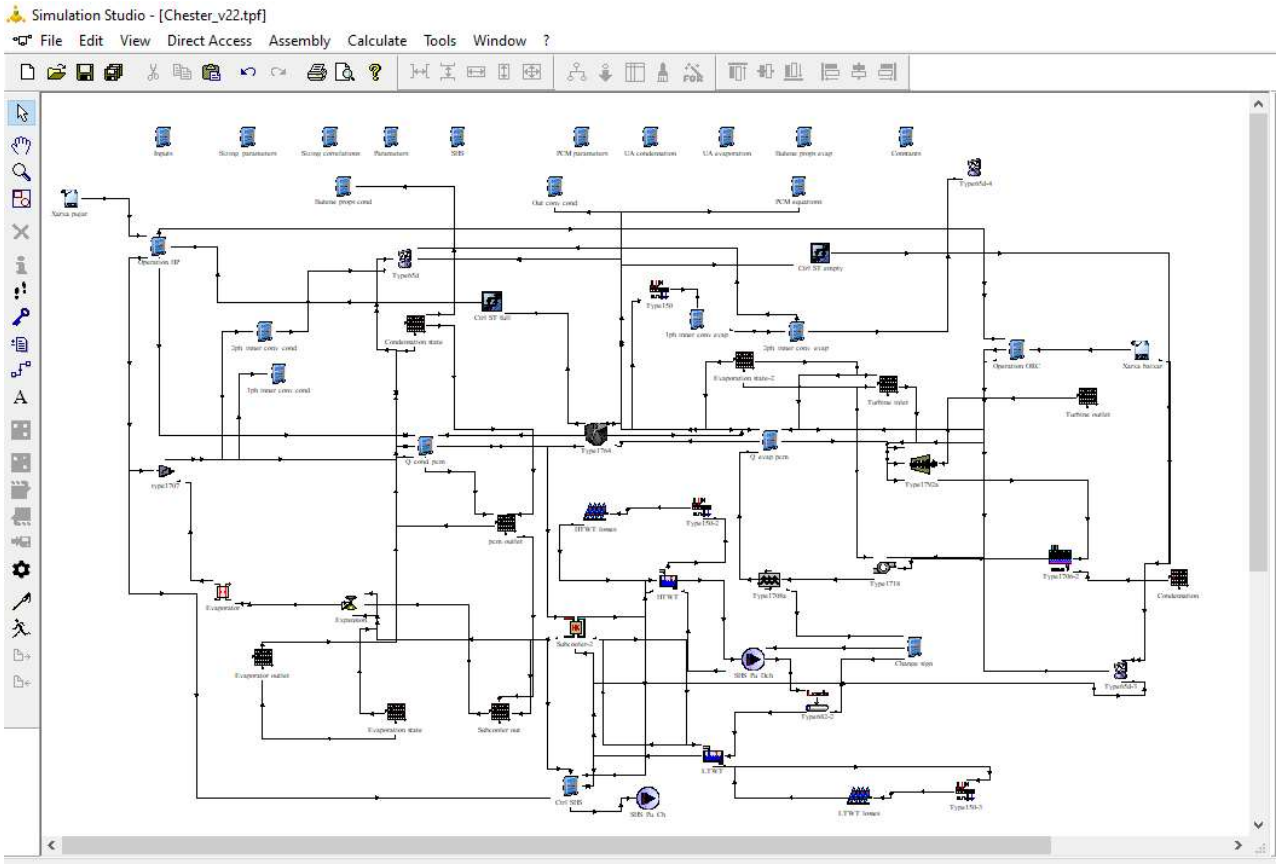

Figure 3: Screenshot of the CHEST model in the TRNSYS simulation environment

5.2. Optimizer

The optimizer module is a set of Fortran libraries for dynamic optimization that interact with the CHEST model API to get the system performance results given a set of input variables. In a generic form, the dynamic optimization problem can be expressed analytically as:

$$min_{u(t)}\ F(x(t), u(t), p(t), t)$$

Subject to

$$h(x(t), u(t), p(t), t) = 0$$

$$g(x(t), u(t), p(t), t) \geq 0$$

For t ∈ [0, tn]

Where F is the objective (or cost) function, x(t) is the system state vector, u(t) is a control input vector and p(t) are system parameters (mostly time independent in our case). The functions h and g represent system constraints in terms of equalities and inequalities, respectively, and tn is the optimization time frame. In the MPC implementation, the optimizer generates a set of control states u(t), and calls the TRNSYS CHEST model, which evaluates the function F. The system state is read by the model from the monitoring system (TSDB) and forecasting module.

The parameters p(t) are characteristics of the system (as heat pump capacity, PCM storage volume, …) which are hard coded in the CHEST model implementation. The optimizer defines new trajectories for the control vector (representing the state of the system within the period 0,tn) which each of its elements take values of charging, discharging or idle.

The cost function implemented is different for the case of Aalborg and Ispaster, due to the different services provided by CHEST in each case. For Aalborg, the CHEST operates as an electrical storage, selling and buying electricity from the grid in both Day-ahead Market (DAM) and balancing market Restoration Reserve (RR), so the cost function maximized is:

$$F = \sum_{t=0}^{t_n}\left(E_{sold}^{RR} \cdot V_{Up}^{RR}\right) - \sum_{t=0}^{t_n}\left(E_{Buy}^{RR} \cdot V_{down}^{RR}\right) + \sum_{t=0}^{t_n}\left(E_{sold}^{DAM} \cdot V^{DAM}\right) - \sum_{t=0}^{t_n}\left(E_{Buy}^{DAM} \cdot V^{DAM}\right) + \\ < V_{sold} > . \eta_{ORC} \cdot \rho_{PCM} \cdot Vol_{PCM} \cdot L_{PCM} \cdot [SOC_{PCM}(t = t_n) - SOC_{PCM}(t = 0)]$$

In this expression, E refers to electricity, either sold or bought according to the index, on the day-ahead market (DAM) or restoration reserve (RR), and is an output of the CHEST model. V refers to the electrical market prices at DAM and RR, that can be used upwards or downwards, depending on the frequency regulation provided. The last term in the sum takes into account the difference in energy stored at the PCM, valuated according to $<V_{sold}>$, the mean electricity selling price for the period, and considering the effect of the ORC efficiency and the latent heat (L_{pcm}) and the volume (Vol_{pcm}) of the PCM material . In short, this term takes into account that for equal cashflow solutions, the one which have the higher state of charge (SoC) of the PCM is preferred. The only constraints applied to the solution is that for each time step, set to 1 hour, the PCM storage SoC is bounded between 0 and 1. The time horizon of the optimization (t_n) is set to 24 hours.

In the case of Ispaster, the optimizer has the duty of maximizing the renewable electricity provided to the microgrid, so in this case, the cost function to maximize is the amount of renewable electricity provided to the micro-grid, which is a direct output from the model.

5.3. Forecasting module

The Forecasting Module consists of a Python library that generates forecast values for different variables by calling other Python libraries. The forecasting module is deployed as a Dockerized microservice, integrated in the Execution Module described in Section 4.1.

The suitability of a particular forecasting algorithm to predict future values of a variable depends on the characteristics of the available historical data, on how the behaviour of that variable is affected by independent (and also available) variables and/or how it behaves along the time. Thus, a preliminary analysis was carried out to select the most appropriate forecast algorithm to each variable whose forecast is required by SEMS.

The algorithms used for the forecast of day-ahead market electricity prices and heat demand are described in D4.1, as well as the existing free Python libraries that offer an easy employment of those algorithms and that were used for that purpose. As explained in D4.1, Artificial Neural Networks (ANN) were employed to generate forecasts of energy demand values, given the availability of independent variables that affect those demands – such as air temperature and the type of day of the week – and of a sufficient sample of historical data for training. On the other hand, future values of day-ahead market electricity prices were generated with an S-

ARIMA algorithm, since an autoregressive statistical model is better fitting to predict future values of a time series such as the one that represents these electricity prices.

In order the generate forecast of the electricity prices of the balancing markets, an additional forecast method that was not described in D4.1 had to be applied: k-nearest neighbours (KNN). KNN is a very popular supervised machine learning algorithm that can both be used for classification and regression. A supervised machine learning algorithm distinguishes itself from an unsupervised algorithm because it depends on labelled input data to produce a suitable output for unlabelled data. To do so, the algorithm learns from the available labelled data until being able to create a function that can generate the appropriate output for the new, unlabelled data. When solving a classification problem, the KNN algorithm deals with discrete output values; when solving a regression problem, the algorithm handles continuous numbers. The procedure developed to produce future values of the electricity prices of the balancing market is a dual-structured subroutine that, first, solves a classification problem: it identifies if, for each hour of the 24 hours ahead, the balancing market will open or will not open. For the timesteps it predicts that the market will be opened, the second part of the subroutine to solve is a regression problem: it creates a function that returns as output the electricity price for each hour the balancing market is open. To solve both the regression and the classification problem, the Python KNN library developed by scikit-learn, version 0.24.2 [5] is called.

For each variable whose forecast has to be regularly generated, the Forecasting Module runs the following procedure:

1. Calls the corresponding input data,
2. Converts it into the proper format,
3. Sets the parameters that will be used to create the corresponding forecast model,
4. Calls the library that creates the model,
5. Generates the forecast values for the next 24 hours, and
6. Sends the results to the DB module.

Photovoltaic (PV) and solar thermal (ST) production values for the next 24 hours are also produced by the Forecasting Module, but instead of using statistical or machine learning algorithms, these energy production values are analytically calculated. The calculation is based on the known technical parameters of the corresponding energy systems and the weather forecasts received from external providers, in particular hourly solar radiation and air temperature values. In this case, the procedure followed by the Forecasting Module is the same as the previously described, except that step 4 does not apply.

5.4. DB module

The DB module is a centralised database that stores all data exchanged and generated by SEMS components. Locating the DB module at the centre of SEMS architecture allows to keep control of all data flows from all modules so that no data can be potentially missed.

The DB module is a multi-level system capable to accommodate several databases according to different data treatment needs and data formats. The main database that the DB module integrates is the Open Time Series Database (OpenTSDB) [6], since most of the data treated by SEMS are timeseries. However, the DB Module is also ready to store relational data in a MySQL database [7] and to exchange files via an FTP server.

OpenTSDB is a free, distributed, scalable database specifically optimized to handle time-stamped data. OpenTSDB is prepared to store large quantities of timeseries and enables a classification system to ease the access and treatment of the data. Each data point is characterized by a metric name, an UNIX timestamp (seconds since epoch), a set of key-value pairs tags that identify the metric in more detail and ease its identification and access, and the actual value of the datapoint that correspond to a 64-bit integer or single-precision floating point value.

OpenTSDB is prepared for easy integration in complex and remote systems, since it provides:

- an HTTP API to enable effortless access to the stored data,
- an HTTP API to easily send generated or monitored datapoints to the database and,
- an already developed Docker image, seamlessly suiting SEMS' Dockerised architecture.

D4.1 offers further details on OpenTSDB data storage and treatment optimization and on how to make requests to its APIs.

Since most of SEMS's modules are developed in Python and OpenTSDB is the primary database of the DB module, one of the key-elements of the module is a Python library that was developed to ease the communication with the OpenTSDB APIs. This library, named Read/SendOTSDB, was developed and stored in a private repository of Python libraries and it is hosted in a physical server at Aiguasol's offices.

Read/SendOTSDB adapts the received data into Pandas dataframe structures [8]. All data processing carried out by the library takes advantage of this format, that allows a time-efficient and easy compaction and treatment of data. The library splits the data in several small blocks that simplifies and speeds up the data exchange process, as recommended by the OpenTSDB documentation. Moreover, it works with the same tag system to identify datapoints as OpenTSDB. Therefore, all data that goes through Read/SendOTSDB is all along associated to the same set of key-values pairs that is/will be used to identify them in OpenTSDB. To simplify the process, a default set of tags was defined for SEMS data, with the following tag-keys: project, provider, location, facility, system, latitude and longitude. In order to effectively define the timestamp of the datapoints being handled, the Read/SendOTSDB takes in Python datetime objects with any time zone information and, before sending the data to the database or to the module that will employ them, it automatically translates the datetime objects to the appropriate UNIX format. This approach is helpful to avoid problems related with faulty datetime characterization of the datapoints (such as missing data due to daylight saving change time). At last, to guarantee the integrity of data sent and read from OpenTSDB, Read/SendOTSDB includes inspect points that certify there are no missing values among the datapoints and that the key-values pairs of the tags are correct.

One of the underlying issues of systems that depend on data-intensive structures, such as SEMS, is the struggle to effectively identify a high number of data and handle them without errors. Thus, Airtable [9], a referential database, with a very useful and intuitive filter system has been employed as a data management support tool. All metrics with their corresponding tag-values, description, units, modules that generate them and modules that employ them, and the case study they correspond to have been registered in Airtable.

5.5. Visualization platform

The main goal of the visualization platform is to offer an intuitive instrument to quickly and easily treat, plot and observe the high number of data generated and collected by SEMS, enabling its analysis. However, the visualization platform also helps meet other needs. On the one hand, it allows to verify if the values received by the different modules of the overall infrastructure are being correctly stored. On the other hand, it makes possible to detect if the results of the calculation modules, such as the CHEST model or the forecasting model, return anomalous values. Overall, the visualization platform helps to verify if the overall SEMS infrastructures is functioning correctly and to trace the origin of potential faults.

SEMS visualization platform was built as a set of Grafana [9] dashboards, similarly to the visualization platform created for the monitoring systems of Ispaster and Aalborg case studies, in T4.1 and described in D4.1. Both visualization platforms integrate the extended SEMS architecture. The dashboards that correspond specifically to SEMS present outputs of the system and the corresponding operation variables of the CHEST system emulated by CHEST model. Among others, visualization platform presents values that correspond to the electricity generated by the turbine, electricity consumption by pumps, income from selling electricity and district heating generated heat (when applicable), hours of operation of the system, hours of grid demand, the instantaneous and future CHEST mode of operation, as generated by the optimization module and the CHEST model, etc.

Grafana is an online open-source platform for data visualization, monitoring and analysis in real-time often employed as a Graphical User Interface (GUI) for different data sources, including OpenTSDB and MySQL, the databases DB module is built upon. Grafana enables the creation of data analytics dashboards made of highly customizable visualization panels that can adopt a variety of visualization formats, from simple graphs to heatmaps. Panels retrieve data from the data sources through queries – a request to the database that identifies the required datapoints and defines how to sample them. The format and syntax of each query depends on the type of datasource that stores the requested datapoints. D4.1. explains how to raise these queries and gives further information about Grafana's characteristics and operation features.

It shall be noted that the access protocols to access Grafana described in D4.1. are obsolete: since the date of writing of the document, the URL and the authentication protocol were updated. Users can now access the platform through https://datuma.aiguasol.coop/grafana/. The security certificate necessary to access data in OpenTSDB is no longer necessary. Data stored in the DB Module can now be accessed through a token-protected authentication process. User credentials must be requested to Aiguasol and condense in one-step the required authentication to access the DB module and Grafana.

5.6. KPI calculator

The KPI calculator consists of a Python subroutine deployed in Docker that makes a request to the DB Module to access data sent by the CHEST model and calculates KPIs based on that data. The KPI calculator resorts to Pandas library in order to handle data in the format of dataframe structures and quickly generate results.

6. SEMS demonstration

The correct operation of SEMS depends on the correct development of several independent modules and their implementation in a unified system, as described in Section 3. This section aims to demonstrate each module is properly generating and/or collecting data and that the communication between different modules occurs as expected. Ultimately, through the KPI generated by the KPI Calculator and the outputs of the CHEST model, it will be possible to verify the impact of SEMS in the operation of the virtual CHEST system and in the energy systems it is integrated into, however, the validation of SEMS' results is out of the scope of the present report and will follow in the upcoming T4.5.

As previously said, among other functions, the Visualization Platform is a valuable tool to verify if each module and the overall infrastructure are properly working. Therefore, the following sections present the data plotted by the Visualization Platform that corresponds to the data retrieved and/or generated by each SEMS' component and sent to the DB Module. It shall be taken into account that SEMS is not yet implemented, and that the data displayed by the visualization panels refer to punctual tests carried out to validate the operation of the different modules and communication processes.

6.1. Archive data

The availability of archive data, that corresponds to both monitored variables and to data manually collected from external provider, had already been verified and shown in D4.1. From the date of writing of D4.1, the database where these data were stored was integrated in the DB Module. The heat availability and demand of Aalborg DH, shown in Figure 4, is an example of the many archive variables whose data is stored in the DB module.

Figure 4:Archive data - heat demand and availability - DH of Aalborg

6.2. External data and forecasted data

It has been verified that data collection from both Meteoblue's and REE's APIs has been correctly implemented. Data has been received with the expected resolution and with the configured regularity. As an example, Figure 4 and Figure 5 show the air temperature and the incident solar radiation, respectively, from Ispaster, received from Meteoblue.

Figure 5: Air temperature in Ispaster - received through Meteoblue's API

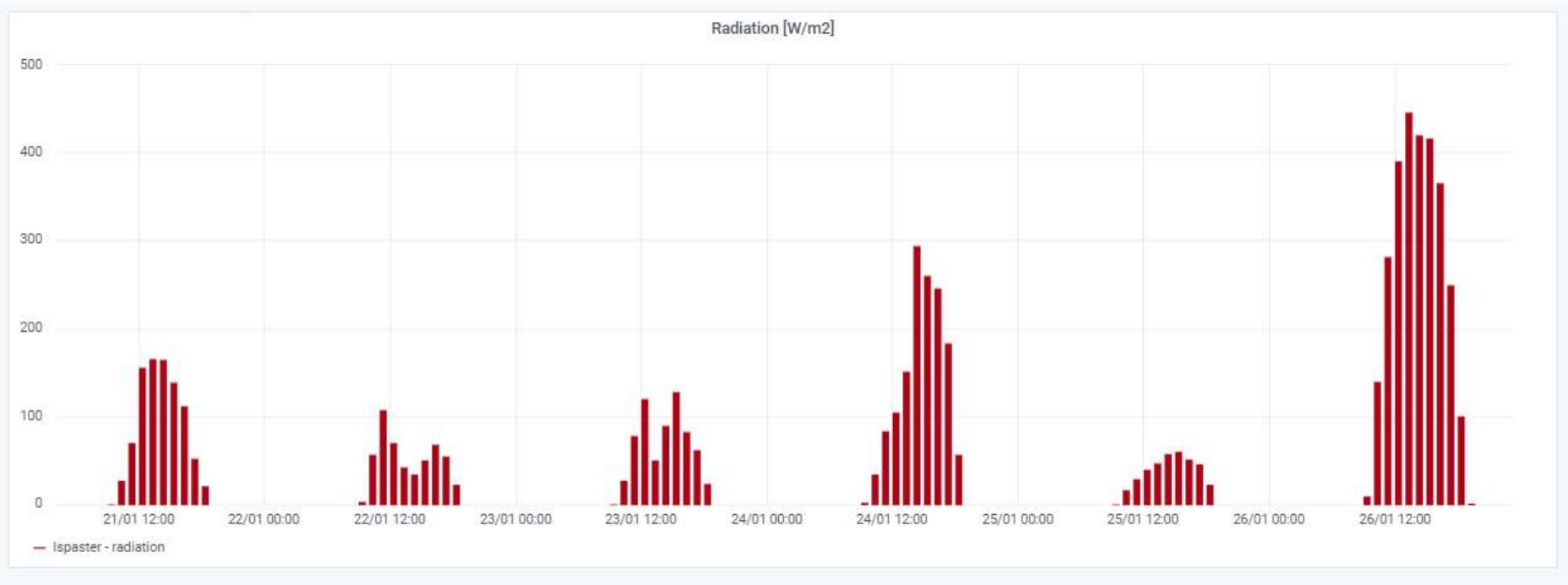

Figure 6: Incident solar radiation in Ispaster - received through Meteoblue's API

Figure 7 presents the electricity price of the day-ahead market of Spain. The real data, from REE's API can be seen in green.

The communication and automatization of the forecasting module with the DB module has also been confirmed. Figure 7 shows the forecasted values in yellow. The forecast shows a good representation of the electricity price at the DAM, but misses to track the sudden changes of the original data, which is something normal in ARIMA models (autoregressive), that tend to smooth the forecasted signal. In general, there is a stochastic component in the price that can't be identified by the forecast, at least, without the incorporation of exogenous drivers of such steps. ARIMA models are black-box models that get good overall fitting but can't follow sudden changes.

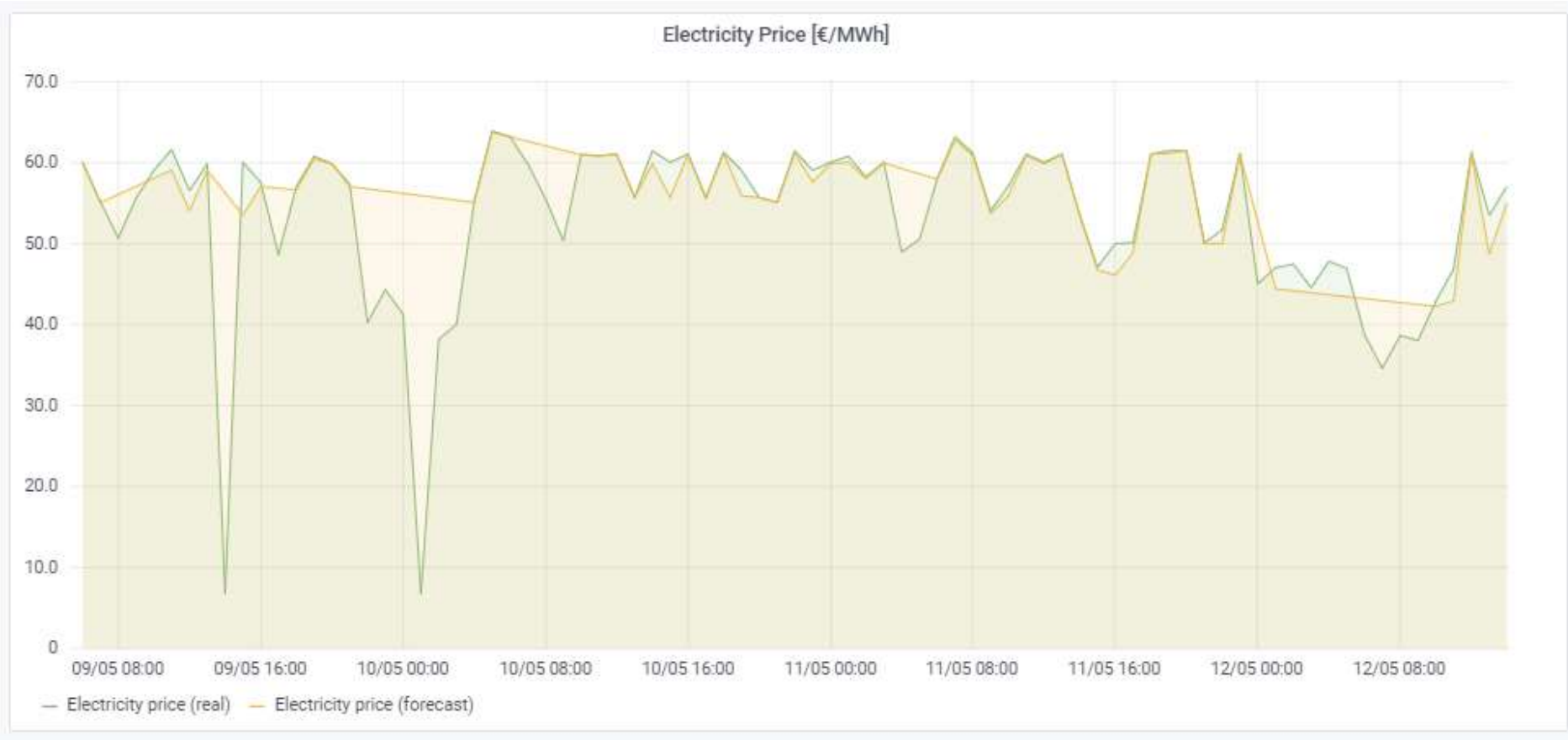

Figure 7: Real values from external provider and forecasted values generated by the forecasting module

6.3. CHEST model and optimizer outputs

As previously explained, data generated by the CHEST model and the optimizer are accessed via request to the model API. The following images allow to verify whether the outputs of both modules are correctly stored in the DB module. As an example of CHEST model outputs, the electricity produced by the ORC and the heat absorbed by the HP are presented in Figure 8 and Figure 9, while Figure 10 shows a discretization data panel that represents the operation strategy produced by the Optimizer and provided to the CHEST model and, by its turn, to the DB Module, after the Execution Module issuing a request to the model API. Along with the discretization panel, a legend is presented. The legend displays the number of periods CHEST system was set to work in each operation mode (the "x" in the legend means "times") and also the total relative amount of time the system was configured in each operation mode. The relative amount of time shown by the legend corresponds to a percentage of the total period of time displayed by the panel. The time horizon of the panel is regulated by the visualization platform user. The information shown by this panel can span from the moment SEMS was deployed until the present moment (presenting both historical and current data), and it can also present the strategy generated by SEMS for the next 24 hours, if the user configures the time horizon of the analysis up to the next 24 hours.

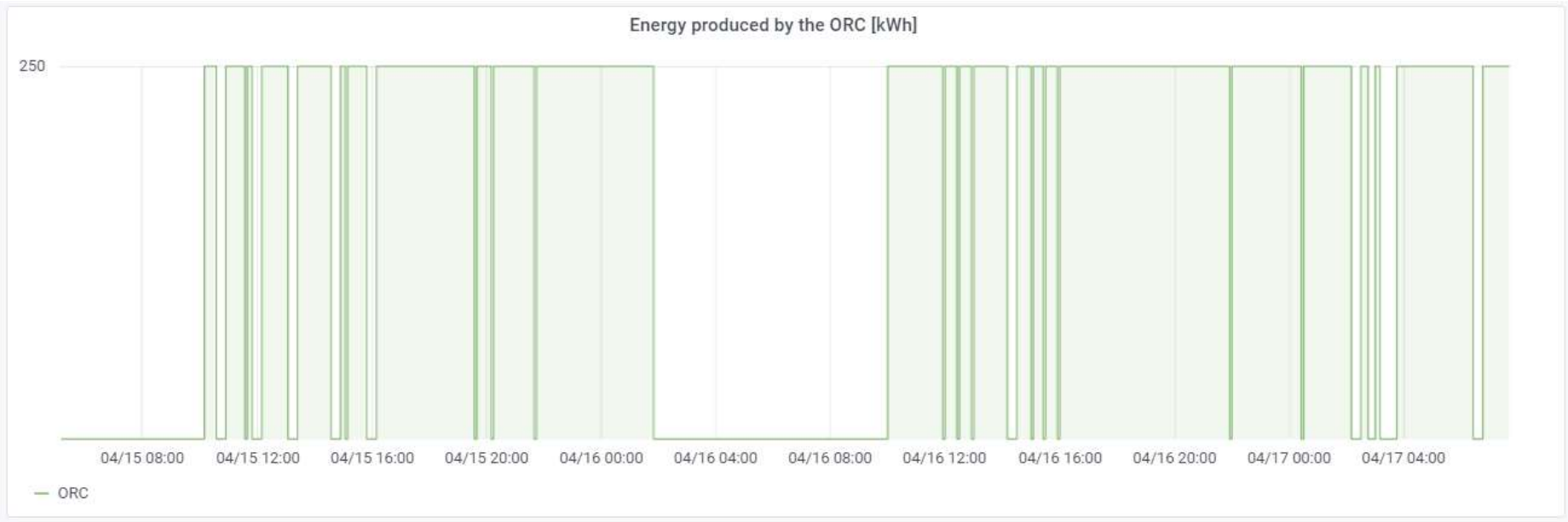

Figure 8: CHEST model outputs – electricity produced by the ORC

Figure 9: CHEST model outputs – heat absorbed by the HP

Figure 10:SEMS operation strategy

6.4. KPI Calculator

The integration and operation of the KPI calculator was also verified through testing. Figure 11 shows the accumulated power to power efficiency ratio of the CHEST system obtained for a testing period.

Figure 11: Power to power ratio

7. Conclusions

This deliverable presents the architecture based in Dockerized microservices and independent modules developed and deployed for CHEST's Smart Energy Management System. SEMS was applied to the virtual CHEST system of Aalborg and Ispaster case studies, with the aim of maximizing the benefits provided by the CHEST system in both case studies.

Several tests were carried out to verify the correct functioning of the different modules and communication processes. The results of these tests were observed through the visualization platform that ultimately works as the SEMS user interface. The infrastructure was tested and has shown to be robust, reliable and to have an efficient performance. In the case of Aalborg, a significant increase of the revenues associated with the storage and arbitrage grid services when compared with static price strategies was observed. For Ispaster, the SEMS wasn't able to improve the CHEST performance, and this is assumed to be a consequence of the high amount of excess of PV electricity, the constraints on the optimization in order to fulfill the demand as well as the relatively low P2P ratio at this site, that takes the optimization of the use of renewable electricity to a trivial solution.

As next steps, in T4.5, the operation strategies that the SEMS optimizer generates will be analysed and validated, as well as the CHEST outputs and KPIs that result from the implementation of those strategies.

References

[1] Red Electrica de España, «Esios,» [En línea]. Available: https://www.esios.ree.es/es/analisis. [Último acceso: 20 9 2021].

[2] Meteoblue, «Meteoblue API Dataset,» [En línea]. Available: https://www.meteoblue.com/es/weather-api/dataset-api/index. [Último acceso: 2 10 2021].

[3] S. Klein y e. al., «TRNSYS 18: A Transient System Simulation Program,» Solar Energy Laboratory, University of Wisconsin, Madison,USA2017.

[4] Bell Thermal Consultants, «www.coolprop.org».

[5] L. L. G. B. M. P. F. M. A. G. O. G. V. Buitinck, «API design for machine learning software: experiences from the scikit-learn project,» de *ECML PKDD Workshop: Languages for Data Mining and Machine Learning* , 2013.

[6] «OpenTSDB,» [En línea]. Available: http://opentsdb.net/.

[7] MySQL, «MySQL documentation,» [En línea]. Available: https://dev.mysql.com/doc/. [Último acceso: 2021 9 22].

[8] The Panda's development team, «Data Structures for Statistical Computing in Python,» Zenodo, February 2020. [En línea]. Available: https://doi.org/10.5281/zenodo.3509134. [Último acceso: 2021 9 22].

[9] Airtable, [En línea]. Available: www.airtable.com. [Último acceso: 2021 9 22].

[10] “Grafana,” [Online]. Available: https://grafana.com/plugins/grafana-worldmap-panel. [Accessed April 2019].