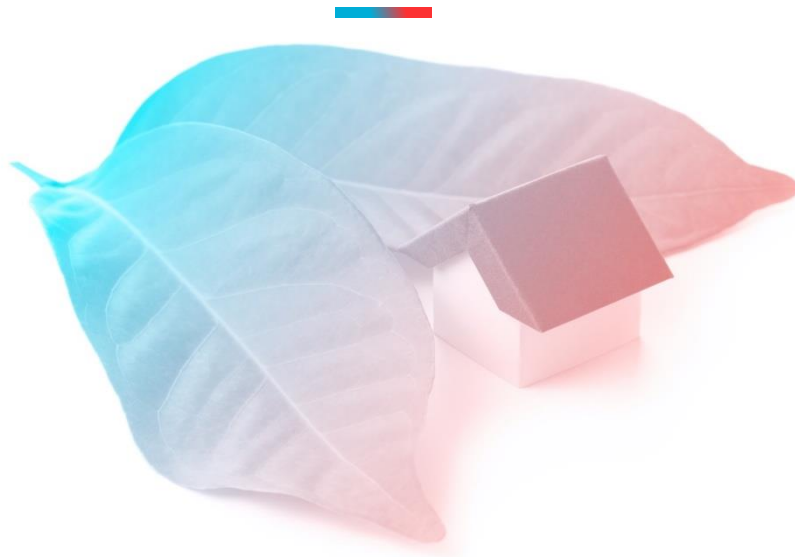




D5.4

Initial design of the IoT platform for user interaction



Authors:

Chrysovalantis-Giorgos Kontoulis (CER), Dimosthenis Ioannidis (CER), Efstathios Bozikas (CER),
Zoi Boutopoulou (CER) Zoltán Kővári (WOO), Zoltán Pásztor (WOO), Philipp Schütz (HSLU),
Susana Gutiérrez (CAR)



This project has received funding from the European Union's Horizon 2020 research and innovation programme under the grant agreement No 869821

Summary			
<p>The main purpose of this deliverable is to present the first approach and current status of the user interface of the IoT platform to be used for MiniStor's asset monitoring and control. The main goal is to develop an enhanced user experience and optimize the energy management system. Following activities in task T5.3, a framework is created to allow for the monitoring of critical system parameters and real-time optimization of the system's usage. In this context, this deliverable presents in detail the initial approach of the deployed techniques and specifications of the user interface of the IoT platform. It also details the implementation methodology for functionalities and operation modes it offers to the user, the architecture of the data layer (communication platform) as well as documentation for the REST API and monitoring device variables. System parameters are described in respect to each pilot site as there are certain differences, which occur due to the diverse type of monitoring equipment and communication protocols present at the pilot sites. Finally, the document describes the test plans and methodologies for integration and acceptance of the system's components as well as a list of the integrated software and hardware components.</p>			
Deliverable Number		Work Package	
D5.4		WP. 5	
Lead Beneficiary		Deliverable Author(S)	
Centre for Research and Technology – Hellas CERTH		Chrysovalantis-Giorgos Kontoulis (CER), Dimosthenis Ioannidis (CER), Efstathios Bozikas (CER), Zoi Boutopoulou (CER), Zoltán Kővári (WOO), Zoltán Pásztor (WOO), Philipp Schütz (HSLU), Susana Gutiérrez (CAR).	
Lead Beneficiary		Deliverable Reviewer(S)	
IERC		Carlos Ochoa (IER), Maria Lopez (IER)	
Planned Delivery Date		Actual Delivery Date	
31/05/2021		29/06/2021	
Type of deliverable	R	Report	X
Dissemination Level	CO	Confidential, only for members of the consortium (including the Commission)	
	PU	Public	X

Index

Index.....	2
List of Tables	2
List of Images	2
List of Abbreviations and Acronyms	3
1 Introduction.....	4
1.1 Scope and objectives of the deliverable	4
1.2 Structure of the deliverable	4
1.3 Relation to Other Tasks and Deliverables	4
2 Design Approach & Methodology of the IoT platform.....	5
2.1 Methodology.....	5
2.2 Context & Design	6
2.3 Approach and Technologies.....	7
2.3.1 IoT Platform Architecture	7
2.3.2 Development Technologies	8
3 IoT Data Layer Design & Implementation.....	9
3.1 Data Storage and Security.....	9
3.2 Documentation of API operations and endpoints	10
3.2.1 Endpoints for handling <i>assets</i>	10
3.2.2 Endpoints for handling <i>specifications</i>	11
3.2.3 Endpoints for handling <i>devices</i>	11
3.2.4 Endpoints for handling <i>assignments</i>	12
3.2.5 Endpoints for handling <i>schedules</i>	14
3.2.6 Endpoints for handling <i>notifications</i>	14
3.2.7 Endpoints for handling UI widgets of the <i>dashboards</i>	14
3.2.8 Endpoints for handling <i>users</i>	15
3.2.9 Endpoints for handling <i>tenants</i>	15
3.2.10 Endpoints for handling <i>sites</i>	16
3.2.11 Endpoints for handling <i>user preferences</i>	17
3.3 Data Quality	17
3.4 Device variables	19
4 IoT User Interface Design & Implementation.....	23
4.1 Functionalities and Graphical layout of the interface.....	23
4.1.1 Login Screen.....	23
4.1.2 Microgrid	24

4.1.3	Data Analytics	24
4.1.4	Control Panel	26
4.1.5	Prediction.....	28
4.1.6	DR Events	30
5	Testing Methodology.....	31
5.1	Scope.....	31
5.2	Approach and Strategy.....	32
5.3	Integration Test Plan	32
5.3.1	Activities	32
5.3.2	Entry and Exit Criteria	33
5.3.3	Test Techniques	33
5.3.4	Schedule.....	33
5.3.5	Status of Analysis Phase.....	34
5.4	Acceptance Test Plan	36
5.4.1	Activities	36
5.4.2	Entry and Exit Criteria	36
5.4.3	Test Techniques	36
5.4.4	Schedule.....	37
5.4.5	Status of Analysis	37
6	Components Integrated with the IoT platform	39
6.1	Tools for source management, development support system and bug tracking	39
6.2	Software Integration	40
6.2.1	Use case scenarios, assets and relevant software	40
6.2.2	Local RaspberryPI (RPi) Gateway development.....	41
6.2.3	Backups.....	42
6.2.4	Alerts.....	42
	Conclusions	43
	References	44

List of Tables

Table 1 Variables of indoor temperature and humidity sensors (Elvaco CMA10, Elvaco CMA11, Plugwise Sense, S+S RTF1-NTC10k).....	20
Table 2 Variables of weather stations (Davis Vantage Pro 2 plus, DeltaOHM HD52.3DP17, FInoT Agri)	20
Table 3 Variables of heat flow meters (BMeters Ultrasonis ULC, GMDM-I + IWM-PL3 + Hydrosplit-M3, Hydrocal M3, Itron CF-UltraMaXX V).....	21
Table 4 Variables of gas flow meter (Honeywell BK-G4)	21
Table 5 Variables of HVAC unit (Airvent Microplex).....	21
Table 6 Variables of 1-phase electrical energy meters (Carlo Gavazzi EM111, Carlo Gavazzi EM112, Circutor CEM C6, Schneider A9MEM2135).....	21
Table 7 Variables of 3-phase electrical energy meters (Carlo Gavazzi EM340, Circutor CVM-E3-MINI-ITF-485-IC).....	22
Table 8 List of testing activities.....	33
Table 9 Current schedule of integration tests	34
Table 10 RPi Gateway Status	34
Table 11 IoT-HEMS Services Status.....	35
Table 12 1 st Level Control Status	35
Table 13 2 nd Level Control Status.....	35
Table 14 KPI Calculations Status	35
Table 15 PCM, TCM, PVT Status	36
Table 16 Current schedule for acceptance tests	37
Table 17 Questions for first round of acceptance tests.....	38
Table 18 Summary of pre-existing monitoring systems at the demo sites	39
Table 19 Overview over common source organization and development organization tools.....	40
Table 28 Status and scripting language of different activities and modules for the local RPi.....	41
Table 29 Automatic download of central IoT data for back-up status	42
Table 30 Activity/Module for low- and mid-level alert module for RPi Gateway status.....	42

List of Images

Figure 1 Agile Methodology Steps.....	5
Figure 2 MiniStor IoT User Interface Development Schedule	6
Figure 3 IoT platform ecosystem	7
Figure 4 Main Endpoints of the IoT's platform Rest API.....	9
Figure 5 REST API supported functions.....	10
Figure 6 Endpoints for handling "assets"	11
Figure 7 Endpoints for handling "specifications"	11
Figure 8 Endpoints for handling "devices"	12
Figure 9 Endpoints for handling "assignments"	13
Figure 10 Endpoints for handling "schedules"	14
Figure 11 Endpoints for handling "notifications"	14
Figure 12 Endpoints for handling "dashboard"	15
Figure 13 Endpoints for handling "users".....	15
Figure 14 Endpoints for handling "tenants"	16
Figure 15 Endpoints for handling "sites"	17
Figure 16 Endpoints for handling "user preferences"	17
Figure 17 Data preparation steps	18
Figure 18 Diagram of the quality-assessment process	19
Figure 19 User Interface of IoT platform, Login Page.....	24

Figure 20 User Interface of IoT platform, Microgrid page	24
Figure 21 User Interface of IoT platform, Data Analytics page – Real time.....	25
Figure 22 User Interface of IoT platform, Data Analytics page – Additional widgets – Real time.....	25
Figure 23 User Interface of IoT platform, Data Analytics page - Historical data	26
Figure 24 User Interface of IoT platform, Control Panel page – Temperature Control - Manual Mode	26
Figure 25 User Interface of IoT platform, Control Panel page - HVAC Controls - Manual Mode	27
Figure 26 User Interface of IoT platform, Control Panel page - Schedules - Manual Mode.....	27
Figure 27 User Interface of IoT platform, Control Panel page - Auto Mode (AI Controlled).....	28
Figure 28 User Interface of IoT platform, Prediction's page - One Day Ahead.....	28
Figure 29 User Interface of IoT platform, Prediction's page - Real Time.....	29
Figure 30 User Interface of IoT platform, Predictions page - Error Deviation Plot - Real Time.....	29
Figure 31 User Interface of IoT platform, Prediction's page - Historical Data	30
Figure 32 User Interface of IoT platform, Prediction's page - DR Events	30

List of Abbreviations and Acronyms

In the rest of the text, the following abbreviations will be frequently used:

API	Application Programming Interface
BASH	Bourne Again Shell
CSV	Comma Separated Values
DR	Demand Response
HVAC	Heating, Ventilation and Air Conditioning
IoT	Internet of Things
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
REST	Representational State Transfer
RPi	Raspberry Pi
SQL	Structured Query Language
UI	User Interface
XML	Extensible Markup Language

1 Introduction

1.1 Scope and objectives of the deliverable

This deliverable presents work developed in the context of Task 5.3 “IoT-platform for user interaction with system for operation and performance” and is the initial deliverable of a total of two deliverables. This deliverable includes a description of the initial version of the user interface of the IoT platform, as well as modules needed to support its function and that were developed from M04 until M19. In addition, this document can serve as a guide for the REST API that is used to manage data sources used by the platform. This report also documents work done for Task 5.4 “IoT- HEMS platform prototyping and acceptance tests”. Both tasks are closely correlated but there is no deliverable associated to T5.4. One of the objectives of this document is to lay out the test plans for integration and acceptance of the MiniStor components to be incorporated, as well as for those components that have already been integrated.

1.2 Structure of the deliverable

The present document is composed of six chapters that cover:

- Chapter 1 - Introduction: aims to introduce the scope of the deliverable and to define relations with other tasks and deliverables.
- Chapter 2 - Design Approach & Methodology of the IoT platform: presents the current development status of the user interface of the IoT-platform.
- Chapter 3 - IoT Data Layer Design & Implementation: provides details about the data storage components and their structure and outlines the API endpoint documentation.
- Chapter 4 - IoT User Interface Design & Implementation: Functionalities of the interface and its graphic design.
- Chapter 5 - Testing Methodology: analysis of the testing plans to assist the integration and acceptance of the systems components.
- Chapter 6 - Components Integrated with the IoT platform: presents a list of the use case scenarios and gateway development.

1.3 Relation to Other Tasks and Deliverables

This deliverable is a product of the Task 5.3, which is part of WP5 “Automated MiniStor Self-Optimization and Control Management Platform”. Also, the deliverable includes results from the Task 5.4 as it is closely correlated to the actions taken in Task 5.3 and also there is not a dedicated deliverable for this task. The task 5.3 takes input from the Task 7.1 “MiniStor design evaluation and user behavior validation through mock-ups and user stories” that defines a mock-up user interface based on user requirements. The user interface is developed in accordance to this mock-up that can be found in D7.1 [\[1\]](#). Also, input is taken from Task 5.1 “Design of the MiniStor control and self-optimization platform” as well. Specifically, the control strategies that are going to be used for the automatic usage and optimization of the system are going to be integrated with the front-end interface. Also, the KPI formulas that are visualized in the platform have been defined in Task 6.1 “Design of the monitoring, definition of KPIs and design of remote data access” and its corresponding deliverable.

2 Design Approach & Methodology of the IoT platform

This chapter describes the methodology and the approach followed to reach the current framework architecture of the system. Note that the architecture described in this deliverable is the initial iteration out of a total of two. During the life span of the MiniStor project, validation of the systems will take place in a continuous and iterative manner to optimize interfaces, procedures and components for improved user experience. The next iteration of the deliverable (due at M39) will provide a complete version of the user interface with all the required systems and components developed from other technical work packages.

2.1 Methodology

We began the development of the IoT platform, by defining the user and business requirements for it, taking into account a state-of-the-art analysis. Using these requirements, we defined the conceptual architecture, a high-level view of the overall system, which later led to the creation of the system's static and dynamic view. As a last step we specified detailed architectural elements, like identifying inputs and outputs, data exchanges, etc. The IoT platform was developed following the Scrum methodology [2].

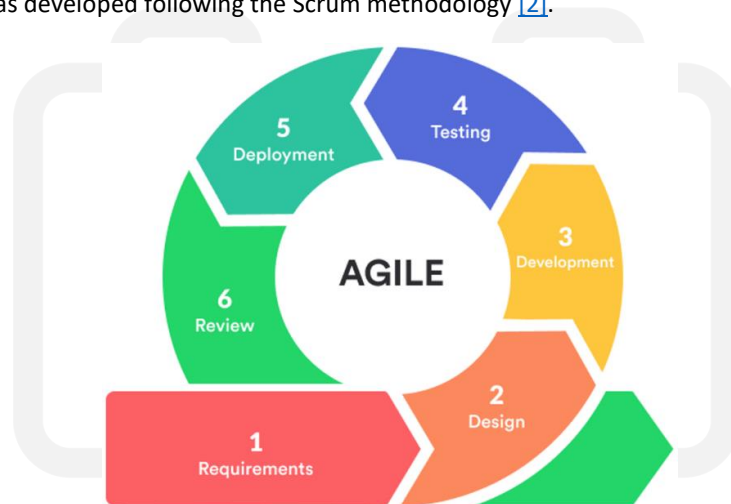


Figure 1 Agile Methodology Steps

Scrum is an Agile [3] based methodology for development. As it is usual in Agile methodologies, the project was divided into small pieces and built incrementally, while ensuring the ability to adapt and change in any step of the process. In Scrum the team decides on how to implement each task and the client only specifies what the outcomes should be. Scrum follows a sprints format as Agile (Figure 1). A sprint is "a set period of time during which specific work has to be completed and made ready for review". So each of these sprints is time allocated to a few specific tasks and deliverables, but in the end of the sprint the item made must be shippable (e.g. to the final user or other work groups). During a sprint, team members have daily short meetings to discuss their progress with the team manager and the project owner. While the team manager oversees the team, he does not give tasks to the team members directly. His role is to coach the team members so they can deliver high quality products. The project owner's is there to represent the vision of the client and set the sprints priorities. At the end of the sprint the team along with the team manager and the project owner discuss on improvement points for the next sprint. A time schedule for this task that follows this approach can be seen in Figure 2.

MiniStor IoT platform

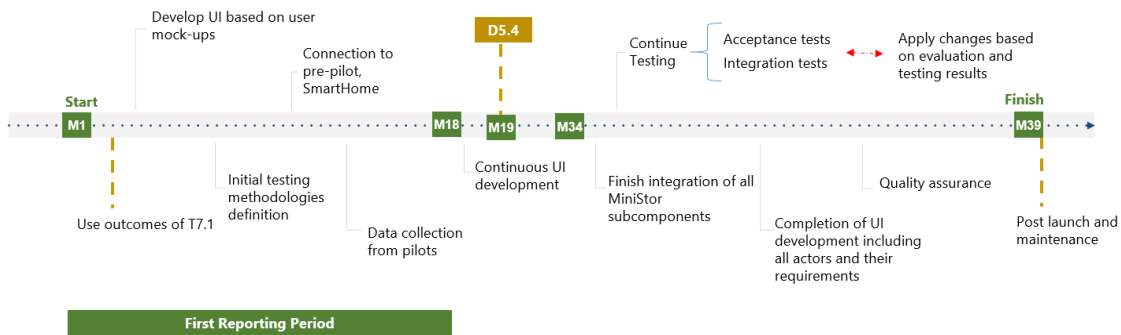


Figure 2 MiniStor IoT User Interface Development Schedule

2.2 Context & Design

The design of the user interface of the IoT platform is based on use cases and user requirements that arise from the user interaction with the MiniStor system, which were collected during the Task 7.1. The basic functionalities that resulted from this analysis include:

- i. Monitoring of the system's operation and performance (historical, current and future data) as well as the indoor and outdoor environmental conditions
- ii. Management of the system's usage by transmitting actions to the system or by utilizing the automatic self-optimization component that utilizes forecasting modes powered by machine learning to control the system efficiently
- iii. Monitoring of the Demand-Response (DR) events status, their effectiveness and manual scheduling of the system's usage
- iv. Notification system that notifies users in case of faulty equipment or when conditions set by the user are met, and, personalization of the user interface based on user preferences

The main purpose of the platform is to allow users to live in a comfortable environment while saving money by improving the energy efficiency of their dwellings. This can be achieved by optimizing the day-to-day usage of the system in order to avoid unnecessary energy-consuming actions in the dwelling with the overall objective to save money from their energy bills without affecting user wishes. To achieve maximum savings, a user can utilize the platform as described in the following scenarios:

- i. Identify potential patterns in the electricity or thermal energy consumption that allows to tune their device usage accordingly in order to reduce costs without hindering quality of life
- ii. Enable the self-optimization module in order let the system tune its usage automatically, again targeting maximum saving without any change in the perceived usage
- iii. Enable schedules for the system (e.g., thermal) to turn on just before the arrival of the user to the building (e.g., return from work)
- iv. Manage DR Events for selling excess energy to the grid based on current and predicted conditions, while being aware of the sell prices
- v. Manage DR Events for controlling the need for energy and battery balances

Note that Demand-Response is a procedure that aims to propagate signals to the users for specific time periods to adjust their power usage to better match the supply. This allows them to reduce or shift their power usage

during peak hours to reduce their financial costs. For the user types, there are two main ones, the basic user and the admin. The difference between those two users are mainly their privileges. In addition to information available to a normal user, basically, an admin can check and handle the MiniStor infrastructures, like for example to add or remove buildings or check notifications about the system's functional state. More information can be accessed in the relevant deliverable where basic types of users are described in more detail (D7.1).

The architecture of the IoT platform is based on the client-server approach, where the client (front-end) is basically the user interface that allows user interaction, and the server (back-end) provides the required services, data and management of requests for the front-end functions to work. The user interface contains five main tabs, which are: i) Microgrid (see section 4.1.2), ii) Data Analytics (see section 4.1.3), iii) Control Panel (see section 4.1.4), iv) Prediction (see section 4.1.5), and, v) DR Events (see section 4.1.6). The back-end utilizes a REST API as a communication vessel to interact in a controlled manner with the databases and the provided web services (HEMS platform).

2.3 Approach and Technologies

The structure of the IoT platform follows a central focused approach, where all IoT components interact with it via a REST API. The user interface (UI) of the platform is implemented with Angular¹, while the back-end is written on Node.js².

2.3.1 IoT Platform Architecture

The data monitoring process involves the propagation of data from the data transmission devices of each demo site, after accumulating all the necessary data locally on gateways, to the respectful databases through a REST API (Figure 3). Then, the user interface (UI) of the IoT platform retrieves the data to create intuitive plots and optimize the operation of the system. The API handles the data in JavaScript Object Notation (JSON) format, which is a standard and human-readable file format that is generally used for server communication. The JSON format is consisted of key-value pairs or key-serializable value pairs for more complex structures.

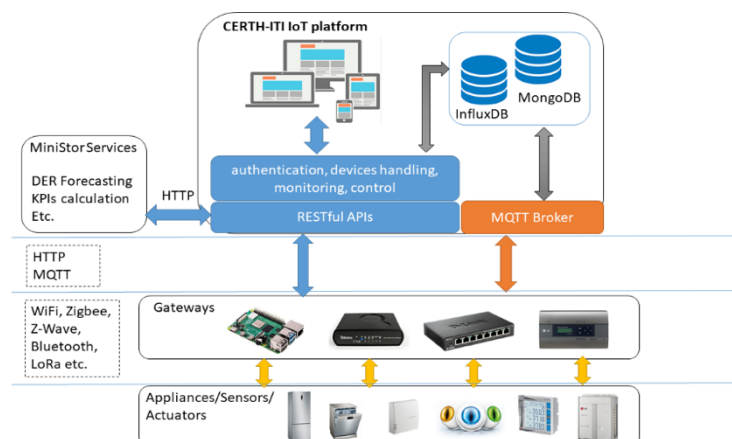


Figure 3 IoT platform ecosystem

¹ <https://angularjs.org/>

² <https://nodejs.org/en/>

2.3.2 Development Technologies

Angular is an open-source framework that is designed to support the display of dynamic content, which is a major limitation of plain HTML. This is achieved by the two-way data binding between a UI element and its corresponding data model. This means that whenever there is a change in both elements, the other is updated accordingly, making it ideal for single-page web pages. The logic of the UI elements is defined in JavaScript, a widely used web-development language.

Angular offers a set of basic functions that are natively pre-build, while at the same time allowing the programmer to create custom ones to capture more advanced scenarios. Also, custom functions are implemented by community members, which further enriches the functionality of the language. Those features make Angular a versatile framework with strong performance.

The representational state transfer (REST) is an application programming interface (API) that is used as a mean of communication with other REST web services. It is a software distributed hypermedia system, and it aims to empower data transferring over the web in a simple, secure, scalable, reliable and portable way with good performance. Node.js is a lightweight cross-platform and open-source environment written in JavaScript that offers fast performance. Generally, Node.js is a widely used technology and it is usually used to develop web servers for distributed applications.



3 IoT Data Layer Design & Implementation

This chapter describes the design and the architecture of the data layer as well as the principles followed to create a secure environment. For the needs of the Ministor project, a REST API is utilized that handles the system data as data needs to be propagated through the IoT platform as well as the HEMS platform. The REST API derives data from two databases. In the following sections, the data handling system is described.

3.1 Data Storage and Security

Data acquired through the use of the MiniStor system are collected and handled in accordance to the EU's and each demo site's national legislation, including GDPR for personal data protection where it applies. More information on the data management plan is described in D1.7 [4].

Data security is ensured by following approaches that comply with the properties of information security, which are data availability, integrity and confidentiality [5]. The sensor data collection is performed securely over the Hypertext Transfer Protocol Secure (HTTPS) that utilizes an encrypted communication protocol. In addition, the data collection in each demo site is monitored in order to ensure the correct operation of the process and the integrity of the data. Also, multi-tenancy is supported, which means that tenants have access only to their own data. These properties are achieved by the utilization of a secure REST API that allows access to the data through encrypted user authentication processes, which promotes data anonymity as well through controlled access.

The aforementioned REST API is connected to two databases. One database is used to save device, user and site details (MongoDB) and the second is used to save the monitoring data of the devices (InfluxDB). The structure of the databases is developed to support the differences of the pilot sites' residency. Access to the data can be achieved either through the user interface of the IoT platform or through code programming. The data can be accessed directly through the databases, but this method is not available publicly, as it is not a good practice due to exposure to security risks. Such security risks could be data access by unauthorized personnel or deletion of data. Instead, communication is performed through specific endpoints of a REST API that requires user authentication to enhance safety and privacy.

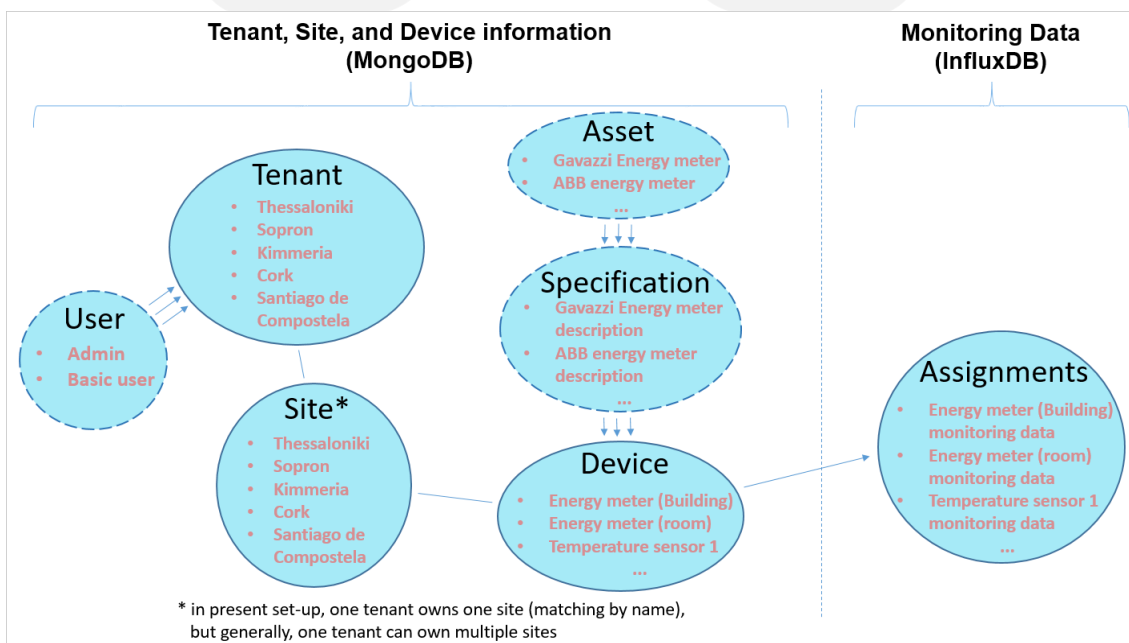


Figure 4 Main Endpoints of the IoT's platform Rest API

The API offers multiple endpoints, but the main ones are the site, user, tenant, assignments, device, specification and asset (Figure 4). The site endpoint allows interaction with information of the pilot sites' location. The user endpoint defines the type of the user and thus his privileges (e.g. basic user, admin), while the tenant manages one or more pilot sites. The asset and specification endpoints are used for the descriptive properties of the devices, while the device endpoint is used to assign a token to a device that is used as device id for the monitoring data endpoint (assignments). Finally, the assignments endpoint provides the upload and fetch functions for the monitoring data.

3.2 Documentation of API operations and endpoints

In our setting, the API allows for posting (POST) and retrieving (GET) monitoring data, while for the user, site, and device information, specifically, it further offers the ability to alter (PUT) or delete them (DELETE) (Figure 5). The API endpoint list follows.

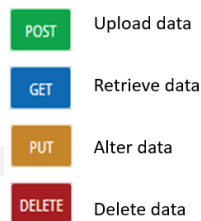


Figure 5 REST API supported functions

3.2.1 Endpoints for handling *assets*

Assets (Figure 6) define the kind of the device and its name. For example, a kind of device could be an electrical energy meter, whose name is Carlo Gavacci.

assets: Operations related to IoT assets. Show/Hide | List Operations | Expand Operations | Raw

GET	/assets/categories	List asset categories that match criteria
POST	/assets/categories	Create a new asset category
GET	/assets/categories/{categoryId}	Get an asset category by unique id
DELETE	/assets/categories/{categoryId}	Delete an existing asset category
PUT	/assets/categories/{categoryId}	Update an existing asset category
GET	/assets/categories/{categoryId}/assets	List category assets that match criteria
GET	/assets/categories/{categoryId}/assets/{assetId}	Get a category asset by unique id
DELETE	/assets/categories/{categoryId}/assets/{assetId}	Delete an existing category asset
POST	/assets/categories/{categoryId}/hardware	Create a new hardware asset in category
PUT	/assets/categories/{categoryId}/hardware/{assetId}	Update an existing hardware asset in category
POST	/assets/categories/{categoryId}/locations	Create a new location asset in category
PUT	/assets/categories/{categoryId}/locations/{assetId}	Update an existing location asset in category
POST	/assets/categories/{categoryId}/persons	Create a new person asset in category
PUT	/assets/categories/{categoryId}/persons/{assetId}	Update an existing person asset in category

GET	/assets/modules	List asset modules that match criteria
GET	/assets/modules/{assetModuleId}	Get an asset module by unique id
GET	/assets/modules/{assetModuleId}/assets	Search for assets in an asset module
GET	/assets/modules/{assetModuleId}/assets/{assetId}	Get an asset by unique id
GET	/assets/modules/{assetModuleId}/assets/{assetId}/assignments	List assignments associated with an asset
POST	/assets/modules/refresh	Refresh the list of asset modules

Figure 6 Endpoints for handling "assets"

3.2.2 Endpoints for handling specifications

The *specification's* (Figure 7) endpoint can be considered as a child of the *asset* endpoint, which is used to define the format of the measurements that can be drawn from a specific device, as well as some complementary information. Note that the term "child" means that except from its own properties, the *specification's* endpoint has the same properties of its "parent", which is the *assets* endpoint.

specifications : Operations related to IoT device specifications. Show/Hide | List Operations | Expand Operations | Raw

POST	/specifications	Create new device specification
GET	/specifications	List specifications that match criteria
DELETE	/specifications/{token}	Delete existing device specification
GET	/specifications/{token}	Get specification by unique token
PUT	/specifications/{token}	Update existing device specification
POST	/specifications/{token}/commands	Create device command for specification.
GET	/specifications/{token}/commands	List device commands for specification
GET	/specifications/{token}/namespaces	List device commands by namespace
GET	/specifications/{token}/proto	Get specification GPB by unique token
GET	/specifications/{token}/spec.proto	Get specification GPB file by unique token

Figure 7 Endpoints for handling "specifications"

3.2.3 Endpoints for handling devices

The *device's* (Figure 8) endpoint is a child of the specifications endpoint and its use is to assign a unique token to each device that is used for device identification in the different functions offered by the API. This endpoint can be considered as the virtual representation of a physical device.

devices : Operations related to IoT devices. Show/Hide | List Operations | Expand Operations | Raw

GET	/devices	List devices that match criteria
POST	/devices	Create new device
DELETE	/devices/{hardwareId}	Delete device based on unique hardware id
PUT	/devices/{hardwareId}	Update an existing device
GET	/devices/{hardwareId}	Get device by unique hardware id
GET	/devices/{hardwareId}/assignment	Get current assignment for device
GET	/devices/{hardwareId}/assignments	List assignment history for device
POST	/devices/{hardwareId}/batch	Add multiple events for device
DELETE	/devices/{hardwareId}/mappings	Delete existing device element mapping
POST	/devices/{hardwareId}/mappings	Create new device element mapping
GET	/devices/{hardwareId}/symbol	Get default symbol for device
GET	/devices/group/{groupToken}	List devices in device group
GET	/devices/grouprole/{role}	List devices in device groups with role
GET	/devices/specification/{token}	List devices using a given specification

Figure 8 Endpoints for handling "devices"

3.2.4 Endpoints for handling *assignments*

The *assignments* (Figure 9) endpoint provides device data. From this endpoint, monitoring data can be accessed under the path: assignments, device token and then measurements. Data can be retrieved either as time series or just the latest recorded value.

assignments : Operations related to IoT device assignments.

Show/Hide | List Operations | Expand Operations | Raw

POST	/assignments	Create a new device assignment
DELETE	/assignments/{token}	Delete an existing device assignment
GET	/assignments/{token}	Get device assignment by token
POST	/assignments/{token}/alerts	Create alert event for device assignment
GET	/assignments/{token}/alerts	List alert events for device assignment
GET	/assignments/{token}/degradation	List measurement events for device assignment
POST	/assignments/{token}/end	Release an active device assignment
GET	/assignments/{token}/events	List events for device assignment
POST	/assignments/{token}/invocations	Create command invocation event for assignment
GET	/assignments/{token}/invocations	List command invocation events for assignment
POST	/assignments/{token}/invocations/schedules/{scheduleToken}	Schedule command invocation
POST	/assignments/{token}/locations	Create location event for device assignment
GET	/assignments/{token}/locations	List location events for device assignment
GET	/assignments/{token}/measurements	List measurement events for device assignment
POST	/assignments/{token}/measurements	Create measurements event for device assignment
GET	/assignments/{token}/measurements/checkSensor	Get last two measurement events for device assignment
GET	/assignments/{token}/measurements/lastValue	Get last measurement events for device assignment
GET	/assignments/{token}/measurements/series	List assignment measurements as chart series
GET	/assignments/{token}/measurementsAggregated	List measurement events for device assignment
PUT	/assignments/{token}/metadata	Update device assignment metadata
POST	/assignments/{token}/missing	Mark device assignment as missing
GET	/assignments/{token}/responses	List command response events for assignment
POST	/assignments/{token}/responses	Create command response event for assignment
GET	/assignments/{token}/statechanges	List state change events for a device assignment
POST	/assignments/{token}/statechanges	Create an state change event for a device assignment
POST	/assignments/{token}/streams	Create data stream for a device assignment
GET	/assignments/{token}/streams	List data streams for device assignment
POST	/assignments/{token}/streams/{streamId}	Add data to device assignment data stream
GET	/assignments/{token}/streams/{streamId}	Get device assignment data stream by id
GET	/assignments/{token}/streams/{streamId}/data	Get all data from device assignment data stream
GET	/assignments/{token}/streams/{streamId}/data/{sequenceNumber}	Get data from device assignment data stream
GET	/assignments/{token}/symbol	Get default symbol for assignment

Figure 9 Endpoints for handling "assignments"

3.2.5 Endpoints for handling *schedules*

The *schedules* (Figure 10) endpoint is used to schedule the time and date that the heating system will be working.

schedules : Operations related to IoT schedules. Show/Hide | List Operations | Expand Operations | Raw

POST	/schedules	Create new schedule
GET	/schedules	List schedules matching criteria
DELETE	/schedules/{token}	Delete a schedule
GET	/schedules/{token}	Get schedule by token
PUT	/schedules/{token}	Update an existing schedule

Figure 10 Endpoints for handling "schedules"

3.2.6 Endpoints for handling *notifications*

The *notifications* (Figure 11) endpoint handles the notification system of the platform. Mainly, notifications can be triggered based on conditions set by the user or in order to inform the user for malfunctioning components.

notifications : Operations related to IoT notifications. Show/Hide | List Operations | Expand Operations | Raw

POST	/notifications	Create new notification
GET	/notifications	List notifications matching criteria
DELETE	/notifications/{notificationToken}	Delete notification by token
GET	/notifications/{notificationToken}	Get notification by token
PUT	/notifications/{notificationToken}	Update existing notification.
POST	/notifications/custom	Create new custom notification
GET	/notifications/kapacitor/disableScript	Disable User Notification Script.
GET	/notifications/kapacitor/enableScript	Enable User Notification Script.
GET	/notifications/kapacitor/listTemplates	List Notification Script Templates
GET	/notifications/kapacitor/listTemplateVariables	List Notification Script Variables
GET	/notifications/kapacitor/listTenantNotificationScripts	List User Notification Scripts
POST	/notifications/kapacitor/new	Create new Notification Script from template
GET	/notifications/kapacitor/removeScript	Remove User Notification Script.
GET	/notifications/userId/{userId}	List notifications matching userId

Figure 11 Endpoints for handling "notifications"

3.2.7 Endpoints for handling UI widgets of the *dashboards*

The *dashboards* (Figure 12) endpoint is used to define the menu and the widgets of each tab of the user interface of the IoT platform. This allows for dynamic construction of different layouts for each actor individually so that the platform suits their needs and to further personalize the user experience.

dashboards : Operations related to IoT dashboards. Show/Hide | List Operations | Expand Operations | Raw

POST	/dashboards	Create new dashboard
GET	/dashboards	List dashboards matching criteria
PUT	/dashboards/{dashboardToken}	Update existing dashboard.
GET	/dashboards/{dashboardToken}	Get dashboard by token
DELETE	/dashboards/{dashboardToken}	Delete dashboard by token
GET	/dashboards/tree/{userId}	List dashboards matching userId
GET	/dashboards/userId/{userId}	List dashboards matching userId

Figure 12 Endpoints for handling "dashboard"

3.2.8 Endpoints for handling users

The *user's* (Figure 13) endpoint is used to define the user type and their privileges (e.g. basic user or administrator).

users : Operations related to IoT users. Show/Hide | List Operations | Expand Operations | Raw

GET	/users	List users matching criteria
POST	/users	Create new user
PUT	/users/{username}	Update existing user.
DELETE	/users/{username}	Delete user by username
GET	/users/{username}	Get user by username
GET	/users/{username}/authorities	Get authorities for user
GET	/users/{username}/tenants	List authorized tenants for user

Figure 13 Endpoints for handling "users"

3.2.9 Endpoints for handling tenants

The *tenant's* (Figure 14) endpoint is a child of the user endpoint and contains information about a specific tenant.

tenants : Operations related to IoT tenants.

Show/Hide | List Operations | Expand Operations | Raw

POST	/tenants	Create new tenant
GET	/tenants	List tenants that match criteria
PUT	/tenants/{tenantId}	Update an existing tenant.
DELETE	/tenants/{tenantId}	Delete existing tenant
GET	/tenants/{tenantId}	Get tenant by unique id
POST	/tenants/{tenantId}/engine/{command}	Send command to tenant engine
GET	/tenants/{tenantId}/engine/configuration	Get tenant engine configuration
GET	/tenants/{tenantId}/engine/configuration/json	Get tenant engine configuration as JSON
POST	/tenants/{tenantId}/engine/configuration/json	Stage tenant engine configuration from JSON
GET	/tenants/authtoken/{authToken}	Get tenant by authentication token
GET	/tenants/device/{hardwareId}	List tenants that contain a device

Figure 14 Endpoints for handling "tenants"

3.2.10 Endpoints for handling sites

The *sites* (Figure 15) endpoint is used to interact with pilot site information as well as some functions tied to a specific pilot site, for example inspection of alerts. Note that one tenant can own or manage multiple sites.

sites : Operations related to IoT sites.

Show/Hide | List Operations | Expand Operations | Raw

GET	/sites	List sites matching criteria
POST	/sites	Create new site
DELETE	/sites/{siteToken}	Delete site by unique token
GET	/sites/{siteToken}	Get site by unique token
PUT	/sites/{siteToken}	Update existing site

GET	/sites/{siteToken}/alerts	List alerts for site
GET	/sites/{siteToken}/assignments	List device assignments for site
GET	/sites/{siteToken}/assignments/lastinteraction	List device assignments for site with qualifying last interaction date
GET	/sites/{siteToken}/assignments/missing	List device assignments marked as missing
GET	/sites/{siteToken}/invocations	List command invocations for a site
GET	/sites/{siteToken}/locations	List locations for site
GET	/sites/{siteToken}/measurements	List measurements for site
GET	/sites/{siteToken}/responses	List command responses for site
GET	/sites/{siteToken}/statechanges	List state changes associated with a site
GET	/sites/{siteToken}/zones	List zones for site
POST	/sites/{siteToken}/zones	Create new zone for site
GET	/sites/{userName}/user	Get site by unique token

Figure 15 Endpoints for handling "sites"

3.2.11 Endpoints for handling *user preferences*

The *user preferences* (Figure 16) endpoint is used to store the settings of the IoT platform for each tenant, and it mainly contains styling settings of the user interface.

userpreferences : Operations related to IoT userPreferences. Show/Hide | List Operations | Expand Operations | Raw

POST	/userPreferences	Create new userPreference
GET	/userPreferences	List userPreferences matching criteria
DELETE	/userPreferences/{userPreferenceToken}	Delete userPreference by token
GET	/userPreferences/{userPreferenceToken}	Get userPreference by token
PUT	/userPreferences/{userPreferenceToken}	Update existing userPreference.
GET	/userPreferences/userId/{userId}	Get userPreference by userId

Figure 16 Endpoints for handling "user preferences"

3.3 Data Quality

Before using the Quality criteria, a data preparation process should take place (see Figure 17). Data implementation is the process of manipulating raw data to make it available to facilitate its analysis. After gathering the data and understanding it, activities such as data cleaning, validation, transformation and/or enrichment have to be performed and, after a successful procedure, this data will be ready to be stored in the database. The most important tasks are data cleaning, data transformation and, if needed, data enrichment.



Figure 17 Data preparation steps

Data cleaning is the process of modifying data to ensure that it is free of irrelevances and incorrect information³. It is also known as data cleansing. The concepts of data cleaning and data transformation are usually closely related. Basically, data cleaning is the process that removes data that does not belong to the dataset, while data transformation is the process of converting data from one format/structure to another. Both processes are usually applied together. Regarding data enrichment, it refers to the process of appending or otherwise enhancing collected data with relevant context obtained from additional sources.

Ministor's data quality operations will be centered in data cleaning whose main aim is to identify incorrect, irrelevant, incomplete, and the "dirty" parts of a dataset and then replacing or cleaning the dirty parts of the data. This process usually improves the data quality, and it can be done automated or manually. The process of data cleaning and data transformation has to be done until the data is reported to meet the defined data quality criteria.

Data cleaning usually involves different steps, being the following the most important ones:

- Removal of unwanted observations
- Duplicate observations
- Irrelevant observations
- Fix data structure
- Filter-out outliers
- Handle missing data
- Drop missing values
- "Generate" missing values

The data cleaning service of the Ministor platform will focus mainly on removal of unwanted observations, filter-out outliers and handle missing data. To ensure timeliness and accuracy of the data, these functions will take place after the data are stored in the database system and before the application of the data in any algorithm or KPI defined in the MiniStor project, such as at the data pre-processing step of the machine learning algorithms of the task 5.3. Within the scope of the MiniStor project, a fault detection module will assist in the reduction of missing values. In addition, the REST API requires a specific data format, while it does not allow duplicate values to be stored in the databases. Note that duplicate values are considered those measurements that have the exact same date and time of recording by the corresponding device.

Once the data has been properly prepared and transformed, a data quality checking method has to be applied at periodic intervals in order to verify the quality of the data gathered. Data quality is not a one-time task but a continuous process. Furthermore, the data quality checking has to be done against specified rules and multiple dimensions such as accuracy of key attributes, completeness of all required attributes, and consistency of attributes across multiple data sets, timeliness of data etc. Depending on the volume and variety of the data and

³ <https://www.formpl.us/blog/data-cleaning>

the scope of the Data Quality activities in the project, both qualitative and/or quantitative assessment methods have to be applied.

To provide completeness to this section, a diagram of the quality assessment process is represented (Figure 18).

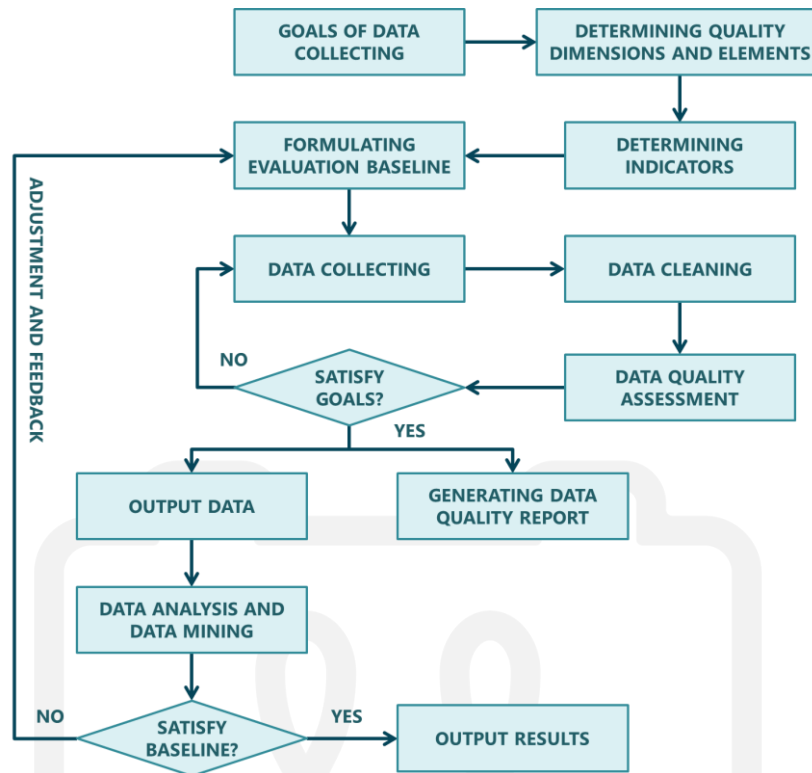


Figure 18 Diagram of the quality-assessment process

The first step in the process defined in [6] is determining the goals of the data collection process. The data sources, types, volume, quality requirements, assessment criteria, specifications and the expected goals are defined in advance.

In order to carry out the quality assessment in the MiniStor project, the platform will be expanded with a fault detection system that notifies the user when a device appears to not be working properly to ensure that the “Availability and Reliability” dimensions are retained. Also, to further reinforce the two aforementioned qualities and to validate the proper selection of the defined conditions for the fault detection module, manual data reviews will take place throughout the development procedure by using the proper tools. For example, a data analysis procedure that features data visualization and outlier detection techniques.

3.4 Device variables

For each demo site, monitoring data are organized with regard to the measuring devices used. Each device accommodates one or more sensors that are able to provide digitized measurement values for certain physical attributes, also called variables. Selection of required measuring devices was performed in Task 6.1, with the goal being the ability to capture all relevant variables that are needed for calculating the Key Performance Indicators (KPIs) of the project. For the definitions of KPIs, as well as for the list of specifications of selected devices for each site, please see 'D6.1 Design of the monitoring system and KPI definition'.

The following tables show indicative definitions of variables used in relation with each type of measurement device. Each row describes one of the possible data fields in the JSON-formatted input or response messages of the REST API described in the previous section. Also, please note that we have listed several variables separately that are not available at every demo site due to differences in capabilities of acquired devices. These variables may not be strictly necessary for KPI calculations, but their values are recorded for supplementary purposes at the request of the responsible partner.

Name	Unit	Cumulative	Aggregate	Site Availability
Temperature	°C	No	No	All
Relative Humidity	%	No	No	All

Table 1 Variables of indoor temperature and humidity sensors
(Elvaco CMa10, Elvaco CMa11, Plugwise Sense, S+S RTF1-NTC10k)

Name	Unit	Cumulative	Aggregate	Site Availability
Temperature	°C	No	No	All
Relative Humidity	%	No	No	All
Solar Radiation	W/m ²	No	No	All
Wind Speed	m/s	No	No	All
Wind Direction	°	No	No	All
Dew Point Temperature	°C	No	No	Cork, Sopron
Wind Gust Speed	m/s	No	Yes	All
Wind Gust Direction	°	No	Yes	Cork, Sopron
Barometric Pressure	mBar	No	No	Cork
Inside Temperature	°C	No	No	Cork
Inside Humidity	%	No	No	Cork
Rain Rate	mm/h	No	No	Cork
Rainfall Amount (Storm)	mm	Yes	No	Cork
Rainfall Amount (Day)	mm	Yes	No	Cork
Evapotranspiration (Day)	mm	Yes	No	Cork
Wet Bulb Temperature	°C	No	No	Cork
Heat Index	°C	No	No	Cork
Wind Chill	°C	No	No	Cork
Temp-Hum-Sun-Wind Index	°C	No	No	Cork
UV Index		No	No	Cork
Sonic Temperature	°C	No	No	Sopron
Absolute Humidity	g/m ³	No	No	Sopron

Table 2 Variables of weather stations
(Davis Vantage Pro 2 plus, DeltaOHM HD52.3DP17, FINoT Agri)

Name	Unit	Cumulative	Aggregate	Site Availability
Thermal Energy Total	kWh	Yes	No	Cork, Kimmeria, Sopron
Volume Total	m ³	Yes	No	Cork, Kimmeria, Sopron
Power	W	No	No	Cork, Sopron
Volume Flow	m ³ /h	No	No	Cork, Sopron
Supply Temperature	°C	No	No	Cork, Sopron
Return Temperature	°C	No	No	Cork, Sopron

Temperature Difference	K	No	No	Sopron
------------------------	---	----	----	--------

Table 3 Variables of heat flow meters

(BMeters Ultrasonis ULC, GMDM-I + IWM-PL3 + Hydrosplit-M3, Hydrocal M3, Itron CF-UltraMaXX V)

Name	Unit	Cumulative	Aggregate	Site Availability
Gas Volume Total	m ³	Yes	No	Cork

Table 4 Variables of gas flow meter (Honeywell BK-G4)

Name	Unit	Cumulative	Aggregate	Site Availability
Fresh Air Temperature	°C	No	No	Sopron
Supply Air Temperature	°C	No	No	Sopron
Extract Air Temperature	°C	No	No	Sopron
Exhaust Air Temperature	°C	No	No	Sopron
Supply Air Flow	m ³ /h	No	No	Sopron
Exhaust Air Flow	m ³ /h	No	No	Sopron

Table 5 Variables of HVAC unit (Airvent Microplex)

Name	Unit	Cumulative	Aggregate	Site Availability
Active Energy Total	kWh	Yes	No	All
Amperage	A	No	No	Cork, Sopron, Thessaloniki
Frequency	Hz	No	No	Cork, Sopron, Thessaloniki
Power Factor		No	No	Cork, Sopron, Thessaloniki
Voltage	V	No	No	Cork, Sopron, Thessaloniki
Apparent Power	VA	No	No	Cork, Sopron, Thessaloniki
Reactive Power	VAr	No	No	Cork, Sopron, Thessaloniki
Active Power	W	No	No	Cork, Sopron, Thessaloniki
Active Power Integrated	W	No	Yes	Cork, Sopron
Peak Value of Active Power Int.	W	No	Yes	Cork, Sopron
Reactive Energy Total	kVArh	Yes	No	Cork, Sopron, Thessaloniki
Active Energy Total (Partial)	kWh	Yes	No	Cork, Sopron
Reactive Energy Total (Partial)	kVArg	Yes	No	Cork, Sopron
Active Energy Total (Negative)	kWh	Yes	No	Cork, Sopron
Reactive Energy Total (Negative)	kVArh	Yes	No	Cork, Sopron

Table 6 Variables of 1-phase electrical energy meters

(Carlo Gavazzi EM111, Carlo Gavazzi EM112, Circutor CEM C6, Schneider A9MEM2135)

Name	Unit	Cumulative	Aggregate	Site Availability
Active Energy Total	kWh	Yes	No	All
Active Energy Total Line 1	kWh	Yes	No	Sopron
Active Energy Total Line 2	kWh	Yes	No	Sopron
Active Energy Total Line 3	kWh	Yes	No	Sopron
Amperage Line 1	A	No	No	Sopron, Thessaloniki
Amperage Line 2	A	No	No	Sopron, Thessaloniki
Amperage Line 3	A	No	No	Sopron, Thessaloniki
Frequency	Hz	No	No	Sopron, Thessaloniki
Power Factor Line 1		No	No	Sopron, Thessaloniki
Power Factor Line 2		No	No	Sopron, Thessaloniki
Power Factor Line 3		No	No	Sopron, Thessaloniki
Power Factor Total		No	No	Sopron, Thessaloniki
Voltage Line 1	V	No	No	Sopron, Thessaloniki
Voltage Line 2	V	No	No	Sopron, Thessaloniki
Voltage Line 3	V	No	No	Sopron, Thessaloniki
Voltage Line 1 to Line 2	V	No	No	Sopron, Thessaloniki
Voltage Line 2 to Line 3	V	No	No	Sopron, Thessaloniki
Voltage Line 3 to Line 1	V	No	No	Sopron, Thessaloniki
Voltage Line to Line Total	V	No	No	Sopron
Voltage Line to Neutral Total	V	No	No	Sopron
Apparent Power Line 1	VA	No	No	Sopron, Thessaloniki
Apparent Power Line 2	VA	No	No	Sopron, Thessaloniki
Apparent Power Line 3	VA	No	No	Sopron, Thessaloniki
Apparent Power Total	VA	No	No	Sopron, Thessaloniki
Reactive Power Line 1	VAR	No	No	Sopron, Thessaloniki
Reactive Power Line 2	VAR	No	No	Sopron, Thessaloniki
Reactive Power Line 3	VAR	No	No	Sopron, Thessaloniki
Reactive Power Total	VAR	No	No	Sopron, Thessaloniki
Active Power Line 1	W	No	No	Sopron, Thessaloniki
Active Power Line 2	W	No	No	Sopron, Thessaloniki
Active Power Line 3	W	No	No	Sopron, Thessaloniki
Active Power Total	W	No	No	Sopron, Thessaloniki
Active Power Integrated	W	No	Yes	Sopron
Peak Value of Active Power Int.	W	No	Yes	Sopron
Reactive Energy Total	kVARh	Yes	No	Sopron, Thessaloniki
Active Energy Total (Partial)	kWh	Yes	No	Sopron
Reactive Energy Total (Partial)	kVARh	Yes	No	Sopron
Active Energy Total (Negative)	kWh	Yes	No	Sopron
Reactive Energy Total (Negative)	kVARh	Yes	No	Sopron

Table 7 Variables of 3-phase electrical energy meters
(Carlo Gavazzi EM340, Circutor CVM-E3-MINI-ITF-485-IC)

4 IoT User Interface Design & Implementation

The purpose of this chapter is to provide an outlook of the various user interfaces developed for interacting with the MiniStor IoT Platform and its subcomponents. The Internet of Things (IoT)-platform allows for user interaction with the system for control and performance monitoring. The IoT infrastructure is composed by a network of devices and the platform acts as a communication vessel for all system components. To improve the quality of life and user experience, the platform needs to provide a user interface where users can fine tune the usage of the system for optimal operation. In view of achieving that, the user interface (UI) of the platform allows for monitoring the system's operation and controlling the operation of the devices, either manually or by utilizing the self-optimization component.

4.1 Functionalities and Graphical layout of the interface

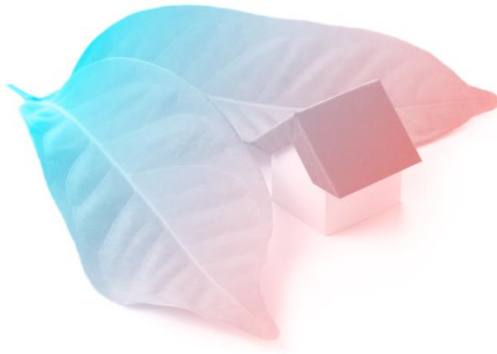
The development of the IoT platform is based on mock-ups defined in the scope of D7.1. The mock-ups were designed so that the flow of the graphical layout promotes logical and intuitive operation of the system. Functionalities of the platform have also been defined in D7.1, by gathering user requirements and usage cases from various users with different backgrounds (the "user stories"). The IoT platform has five main tabs that implement the core functionalities which are detailed in the following sub-sections. The "Microgrid" tab allows for monitoring of the current status of the system. The "Data Analysis" tab allows for monitoring real time and historical data. "Control Panel" allows the user to control MiniStor's devices and create operation schedules as well as to enable the automatic system optimization mode. The "Prediction" tab allows to follow a short-term forecast (1 day ahead), real time and historical energy generation and consumption predictions as well as the error deviation of the predicted values from the actual. Finally, the "DR Events" tab compiles all the demand-response events that are active or they are about or that have already occurred. Note that additional visualization widgets may be added in the next iteration of the platform to support new potential needs of the main MiniStor equipment, which includes the PVT, PCM and TCM components. In the following sections, the progress and results of the development of the actual platform are presented.

4.1.1 Login Screen

The platform requires a user authentication process to allow secure access to their data. For the authentication, a secure encryption algorithms are used. Except for logging-in, users can also enable the function to save their credentials to facilitate ease of use for future usage. It can also serve for the event to change their password securely in case they forgot it via the "Forgot Password?" choice.

Welcome to the MiniStor Monitoring Platform!

Monitor MiniStor's Pilot Sites in Cork, Kimmeria, Thessaloniki, Sopron and Santiago De Compostela!



Log in to your account
Please provide your credentials...

Username

Password

Remember Me [Forgot Password?](#)

LOGIN

Figure 19 User Interface of IoT platform, Login Page

4.1.2 Microgrid

After the log-in process, a user can see a summary of the system's and environment's condition, including both internal and external conditions. External conditions are based on information from a weather station. On this tab, the user can monitor the dwelling's energy consumption. They can also find information about the status of the battery, its stored energy level and whether it is charging or giving energy to the building. Finally, they can see what part of the energy needs of the dwelling are covered by the MiniStor system as well as the amount of energy that is exported or imported from the grid on a daily basis.

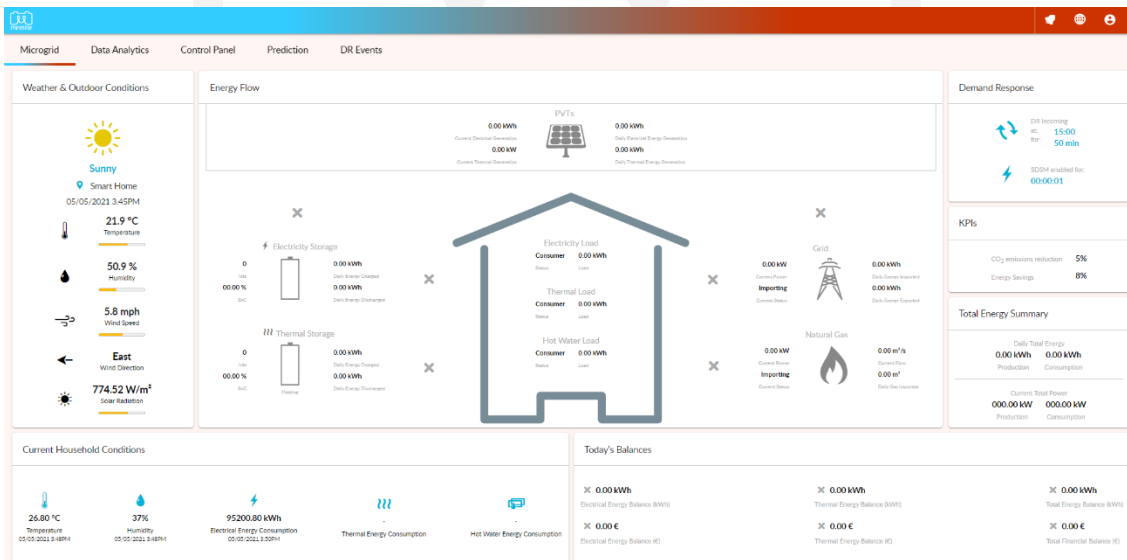


Figure 20 User Interface of IoT platform, Microgrid page

4.1.3 Data Analytics

In this tab, the user can see the electrical and thermal production and consumption of the PVTs as well as the correlated costs from the system's usage in the form of graphs. Also, the user can see the battery status, including the generation, load and current storage. The "Data Analytics" tab allows the user to select between "Real Time" and "Historical Data" modes. For the "Real Time" tab, the user can see the current status of the

system, while for the “Historical Data” option, the user can see measurements of any past date by selecting the proper date range.

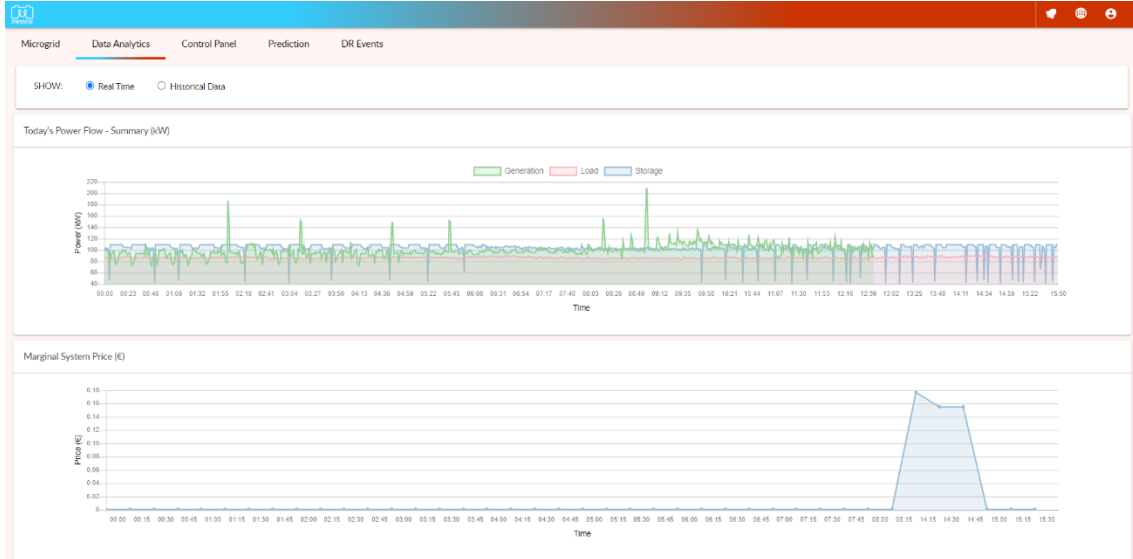


Figure 21 User Interface of IoT platform, Data Analytics page – Real time

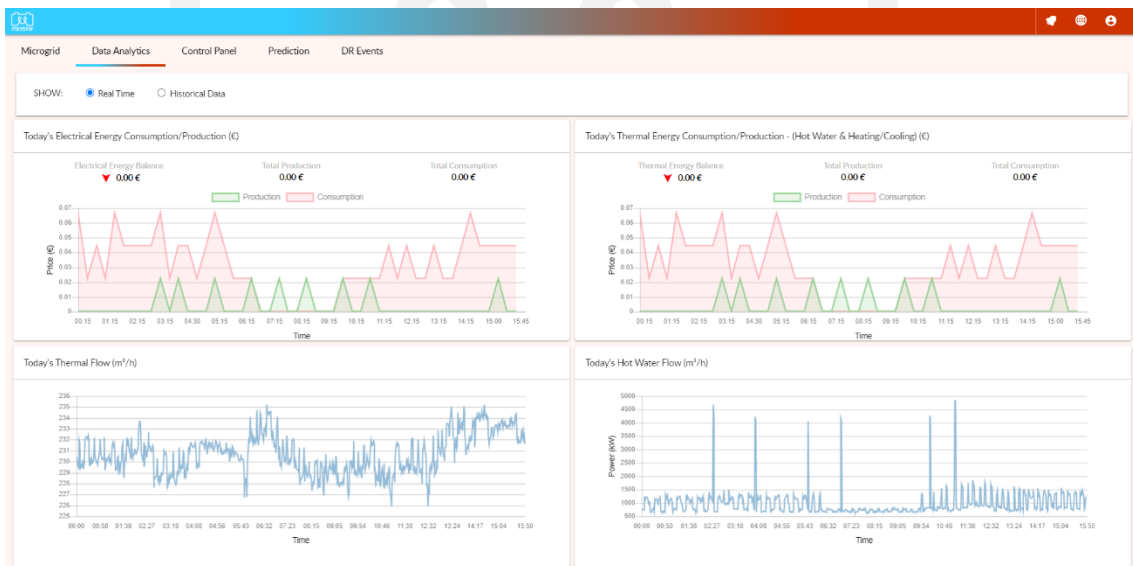


Figure 22 User Interface of IoT platform, Data Analytics page – Additional widgets – Real time

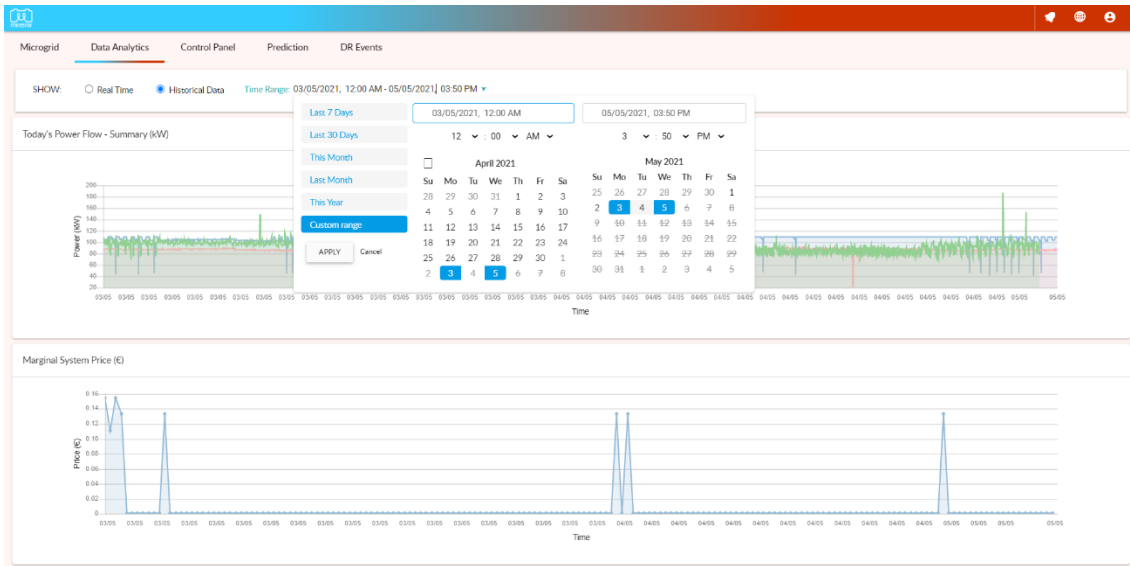


Figure 23 User Interface of IoT platform, Data Analytics page - Historical data

4.1.4 Control Panel

In the “Control Panel” tab, the user is able to control the heating and cooling systems in any individual room in the building that is connected to the system. The user can set a target temperature for the next day or create recurring temperature schedules, as well as schedules for specific occasions. For example, when they are not at home. They can also set their own comfort levels and let the system optimize its usage by taking control of the heating and cooling based on the weather and their previous behavior.

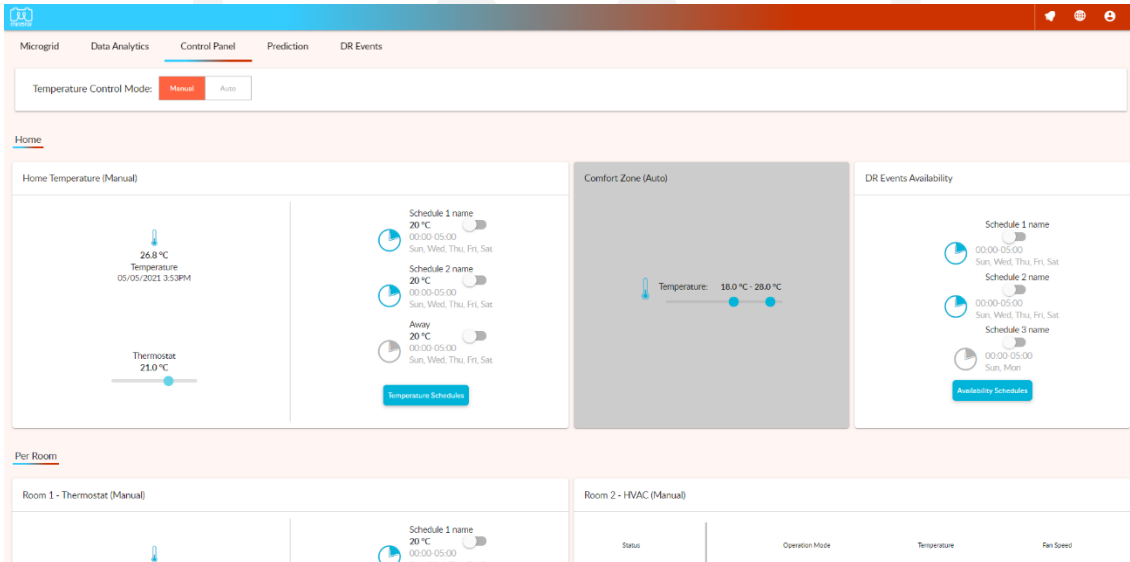


Figure 24 User Interface of IoT platform, Control Panel page – Temperature Control - Manual Mode

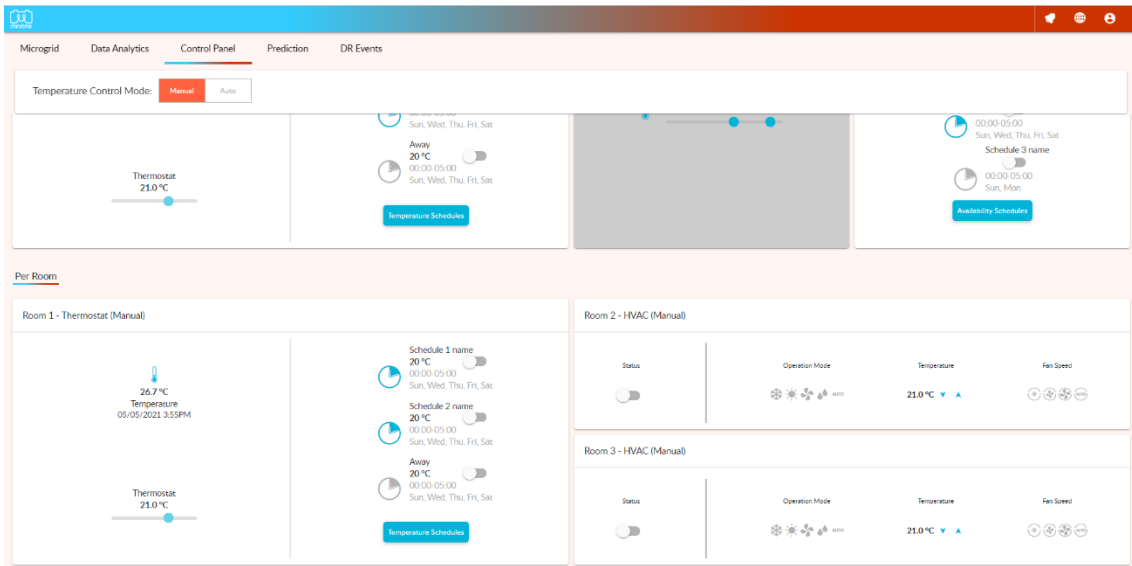


Figure 25 User Interface of IoT platform, Control Panel page - HVAC Controls - Manual Mode

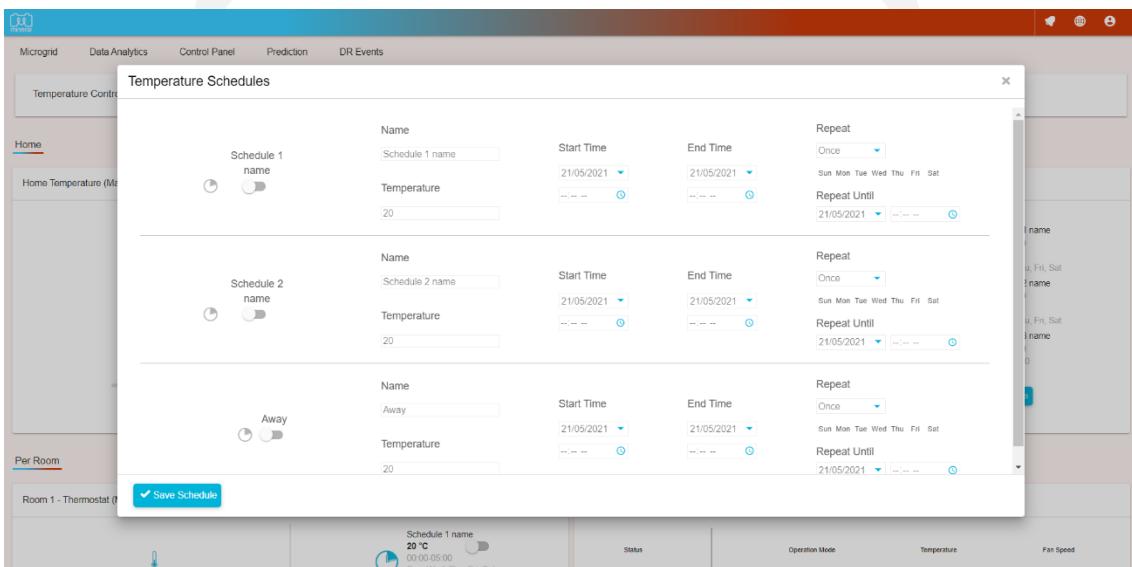


Figure 26 User Interface of IoT platform, Control Panel page - Schedules - Manual Mode

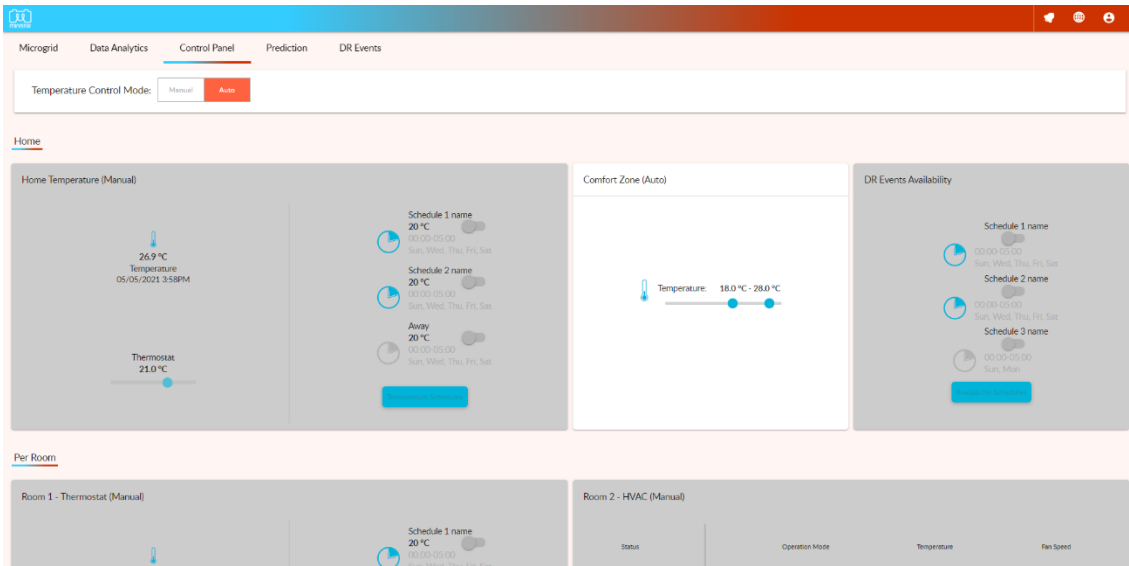


Figure 27 User Interface of IoT platform, Control Panel page - Auto Mode (AI Controlled)

4.1.5 Prediction

In the Prediction tab, the user can see plots that include forecasts for the production and consumption of the total power of the corresponding demonstration site connected with the MiniStor system. The user can see the predicted values for the next day in the "One day ahead" tab. If they want to compare the forecasted values with the real ones, they can select the "Real Time" option. Finally, if the user wants to check the accuracy of the forecasted values compared to the real ones in any specific date, they can choose "Historical Data" and set their desired time range. Note that in both "Real Time" and "Historical Data" modes, the user can see the error deviation of the predictions compared to the real values in a plot.

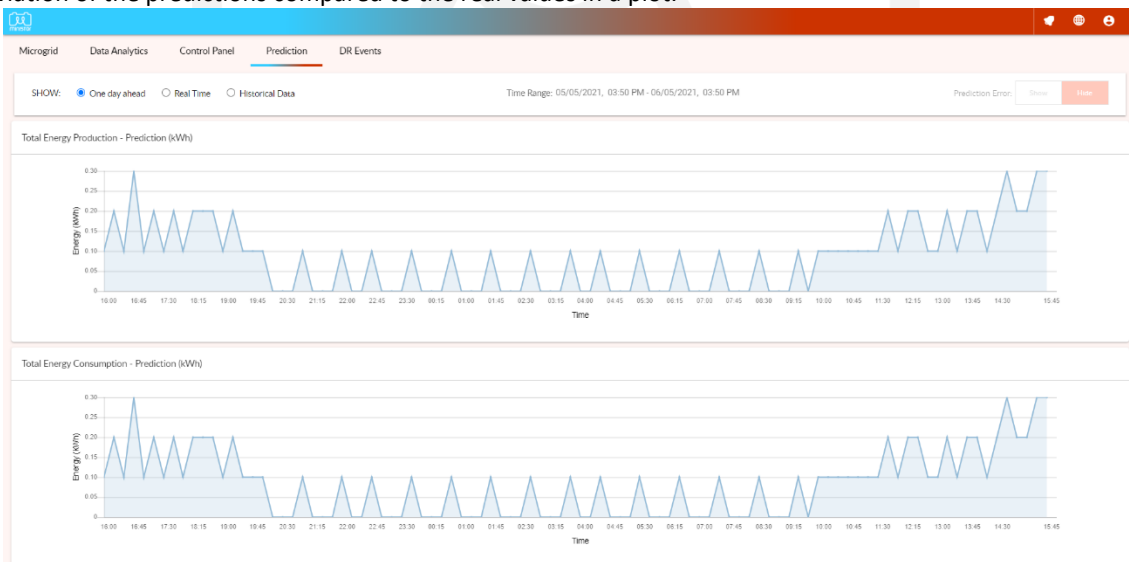


Figure 28 User Interface of IoT platform, Prediction's page - One Day Ahead

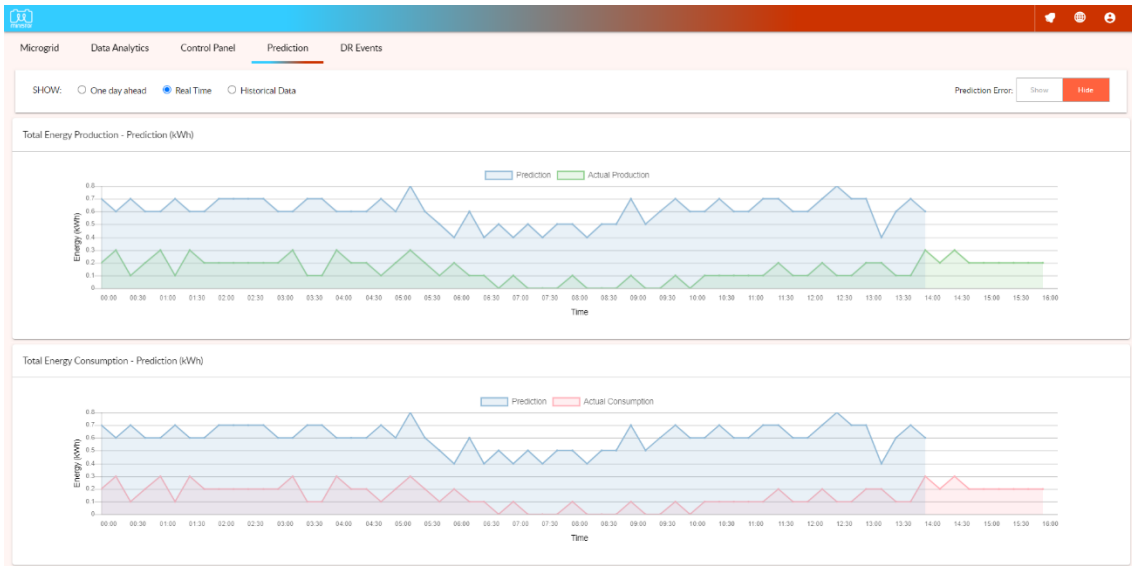


Figure 29 User Interface of IoT platform, Prediction's page - Real Time

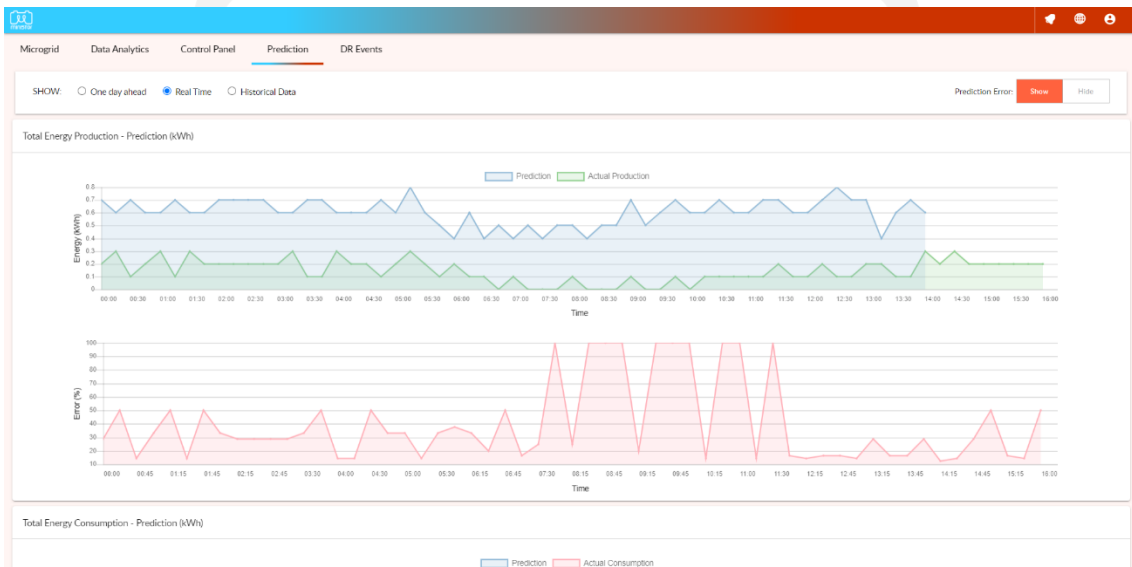


Figure 30 User Interface of IoT platform, Predictions page - Error Deviation Plot - Real Time

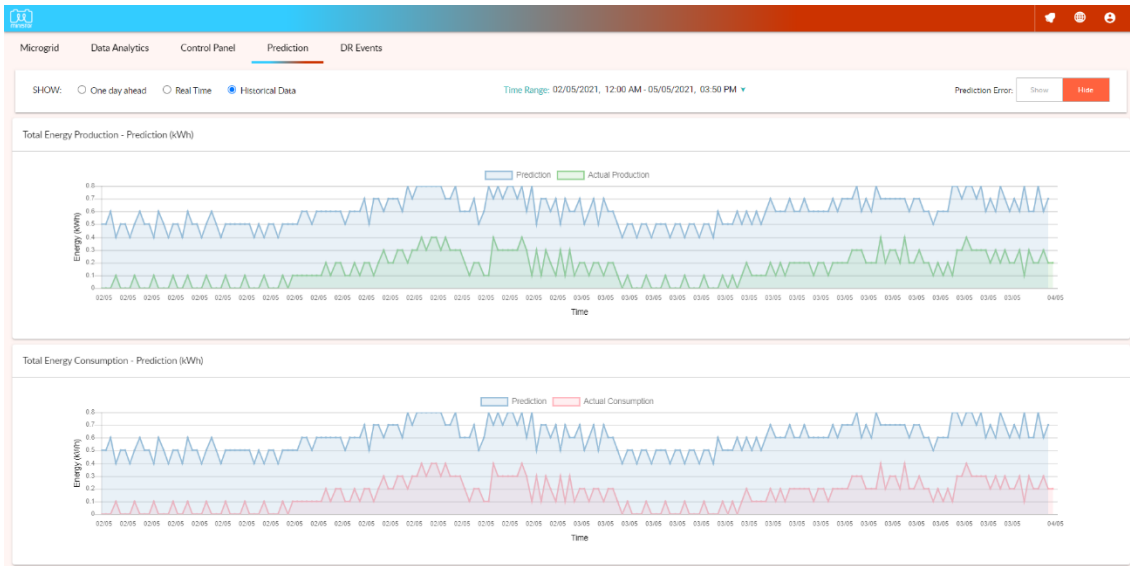


Figure 31 User Interface of IoT platform, Prediction's page - Historical Data

4.1.6 DR Events

In the "DR Events" tab, the user can view current, completed, and upcoming demand-response events. Then, the user is able to decide whether to act or not in order to reduce the system's consumption. Note that DR Events are developed in task 5.2 and are based on market conditions explored in D3.8.

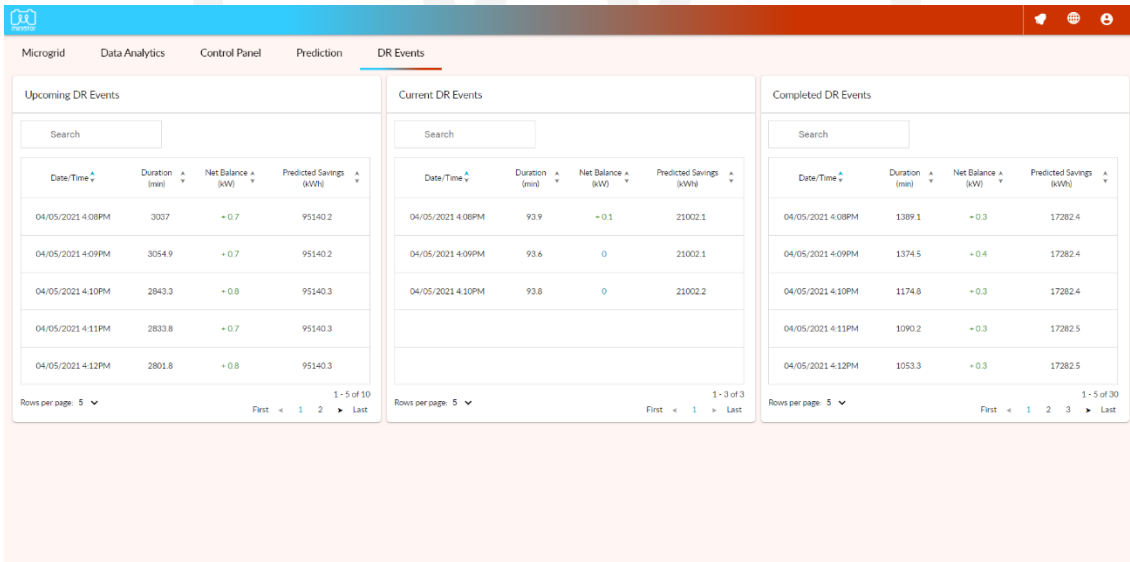


Figure 32 User Interface of IoT platform, Prediction's page - DR Events

5 Testing Methodology

This section covers the testing methodologies that have been developed for the integration and acceptance assessments of the system's hardware and software components. Also, the chapter presents an outlook of the time schedules, planned actions and the state of the analysis phase.

Software testing is a way to assess the quality of the software and to reduce the risk of software failure during operation. Testing plays an important role in the development lifecycle, both in traditional and agile software project management models. Testing is not limited only to executing the finished programs and checking if they are working correctly. It also includes performing these assessments in a methodical manner, which also involves activities such as planning, analysis, design, implementation of test suites, and evaluation.

During the project we are following the terminology and guidelines that has been laid down by ISTQB®, the International Software Testing Qualifications Board⁴. One of the key activities in testing management is to create a **Test Plan**. Its goals are the following:

- Establishing the structure for all the testing efforts related to the development of software modules in the project
- Identifying the scope, i.e. all involved subsystems of the software
- Introducing the testing policy, including approach and strategy
- Presenting a layout for the tests, including:
 - Activities
 - Test types and techniques
 - Roles and responsibilities
 - Schedule

The Test Plan document is maintained continuously throughout the course of development. It has an ever-evolving nature following the progress of testing activities. It also has to address changes in conditions and requirements. The First version of the Test Plan for MiniStor project has been created and approved in Nov 2020 (M13). Throughout this chapter, we present the plan in its current state as of writing this document, with a summary of ongoing testing activities.

5.1 Scope

According to communications between project partners, and through established documentation on proposed architecture and deployment, from the perspective of testing activities the MiniStor software system consists of the following modules, indicated by the current responsible for its calculations:

- Smart HEMS platform – CERTH
 - including: frontend, backend, data layer, security layer
- High level services:
 - 2nd level control and forecasting – CARTIF
 - KPI calculations – HSLU
 - various other HEMS services, including data quality, reporting etc. – CERTH
- Local components:
 - 1st level control – CARTIF
 - PCM, TCM, PVT controllers – SUNAMP, CNRS, ENDEF
 - Monitoring sensor gateway (Rpi) – WOODSPRING
- Separate software:

⁴ <https://www.istqb.org/>

- AR/VR tool – CERTH

As it has been indicated in the list above, for each module there is a distinct team (i.e., consortium member organization), who is responsible for the development work. In general, developer teams are aiming to employ an iterative lifecycle model in the project. For example, modules that are needed for pre-install monitoring are developed first, in order to be able to deploy the monitoring functionality as early as possible. Development cycles of different modules are not strictly synchronized across organizations. Certain project goals and deliverables provide a natural timeframe for progress.

5.2 Approach and Strategy

Generally, we can distinguish the following test levels:

- Component/Unit tests
- Integration tests
- System tests
- Acceptance tests

It has been agreed upon the start of development, that focus of the testing efforts will be on **integration** and **acceptance**. As different modules are developed by different organizations, there are natural boundaries among the interfaces. One of the main goals with testing is to ensure that the integration between modules created by separate organizations can be as seamless and reliable as possible.

This plan does not force the formal use of unit tests for individual components/modules, or complete system level tests. This can be attributed to the fact that the main objective of the project is demonstration of the technology, and there is a limit on what can be achieved with the given time and budget constraints. Nonetheless, every team is encouraged to use the best practices within their development platform, especially if they encounter any mission critical units. In the future, we may include reports on additional tests performed outside the scope of this plan.

Thinking about the system as a whole, another important priority is the actual usability of the software. Despite the complexities of the MiniStor hardware, the end user should be able to handle the UI comfortably, yet being able to access all the important data and settings. This means that our testing efforts must aim to ensure adequate acceptance by users.

5.3 Integration Test Plan

In summary, we have to consider the interface boundaries where the separate modules communicate with one another, mainly with attention to those connections where works of different organizations interact.

5.3.1 Activities

One very important point to clarify are the roles i.e., who is doing what. WOODSPRING have accepted responsibility for coordination of the testing effort, also expecting that most of the testing activities will have to be carried out by them. However, in the activities table below, the term *Authorized tester* will be used (the person who will be responsible for the testing – based on the kind of test and her knowledge), since each module must be examined at the analysis stage, where it would be more appropriate to be tested by the owner of the control code or independently. The main considerations are expected to be related to the question, whether

the tester would need a white-box type of understanding or access to code and infrastructure to create the tests.

Activity	Description	Responsible team
Planning	Creating and maintaining the Test Plan	WOODSPRING
Monitoring	Checking if progress is according to plan	CERTH as Task Leader, with participation of CARTIF as WP Leader and WOODSPRING
Analysis	Determining what to test and who should test it (who is the Authorized Tester)	WOODSPRING, with participation of the Module Owner
Design	Determining how to test	Authorized Tester, with participation of WOODSPRING
Implementation	Creating the tests	Authorized Tester
Execution	Running the tests	Authorized Tester
Completion	Creating reports and drawing conclusions	WOODSPRING, with participation of Authorized Tester

Table 8 List of testing activities

5.3.2 Entry and Exit Criteria

For most modules, the starting of any testing activities beyond analysis will require a certain state of readiness regarding both sides of the interfaces. In conjunction with the iterative development cycles, this should mean that at least the first coding phase is finished and the developer team is ready to receive the feedbacks from testing (entry criteria).

Exit criteria must be specified during analysis for each module individually. Suitable monitoring metrics are to be established in order to follow testing progress, and to be able to create a condition for termination, which is always more preferred to reach before the end of the scheduled deadline.

5.3.3 Test Techniques

Generally, the integration tests will be focused on functional characteristics, primarily with a black-box approach where this can be done. Non-functional attributes like usability, performance and security etc. are of secondary concern, but during analysis it must be considered whether these are to be tested, if the module is involved in a mission critical function.

5.3.4 Schedule

Initial scheduling for integration tests has been aimed to be able to conduct at least one cycle of test execution and feedback for most of the software modules before the expected delivery and installation of the main MiniStor equipment.

To accommodate for the restrictions posed by the Covid-19 pandemic and to avoid delays, the Thessaloniki Smart Home test bed is utilized to proceed with the development of the platform as part of the monitoring system that is already installed. At the moment, monitoring systems are live at all pilot sites, except for Santiago de Compostela that joined the project recently at M12. The demo sites are fully supported by the data layer and by the user interface of the platform. As a next step, the platform's support is going to be expanded to fully

cover all of the pilot sites that faced difficulties in the installation process of their monitoring devices. During this period, integration tests will be performed. In the following sections, brief summaries are included for current status regarding each module, and a complete report on integration test iterations and results will be included in D5.5.

Note: Unless specified, dates apply to the year 2021.

	RPi Gateway (WOO)	HEMS Services (CERTH)	1 st Level Control (CARTIF)	2 nd Level Control (CARTIF)	KPI Calculations (HSLU)	PCM, TCM, PVT (SUNAMP, CNRS, ENDEF)
Analysis	Nov 2020 – Jun 2021					TBD.
Design	Jun – Jul	Jun – Aug	Jun – Aug	Jun – Sep	Jun – Sep	
Impl.	Jul – Aug	Aug – Sep	Aug – Sep	Sep – Oct	Sep – Oct	
Execution	Aug	Aug – Sep	Aug – Sep	Oct	Oct	
Completion	Oct – Nov					

Table 9 Current schedule of integration tests

We must note that proposed dates only cover the first iteration of development/test/feedback cycle at this time. It is possible that for some modules more than one cycle will be applied if needed, which does not necessarily imply the quality of the software module is low. Through a more agile coordination between partners this will be improved in order to be able to test at intermediate stages.

5.3.5 Status of Analysis Phase

Module	RPi Gateway (WOODSPRING)
Status	Analysis nearly completed
Proposed Tester	WOODSPRING
Interface(s)	<ul style="list-style-type: none"> Measurement/Data logging devices IoT REST API
Approach/Strategy	Methodical search for common or likely types of failures, behavior on unexpected conditions. Examples include malformed input data, connection problems, fault tolerance to other hardware anomalies.
Entry criteria	Module is ready for testing
Exit criteria	Further analysis required

Table 10 RPi Gateway Status

Module	IoT-HEMS Services (CERTH)
Status	Analysis in progress
Proposed Tester	WOODSPRING
Interface(s)	<ul style="list-style-type: none"> WEB interface for user interaction IoT REST API to connect with other services Need to be clarified if specialized interfaces will be needed for other high-level HEMS services
Approach/Strategy	<ul style="list-style-type: none"> WEB: Search for common or likely types of failures, also exploratory testing, especially in relation to user requirements (E.g. analyzing whether anything could go wrong if a user makes a mistake in pre-defined usage scenarios)

	<ul style="list-style-type: none"> REST API: Methodical search for cases where input data fails to be recorded or displayed correctly in the user interface Supplementary: testing of security features
Entry criteria	Module is partially ready, testing could be conducted in incremental steps
Exit criteria	Further analysis required

Table 11 IoT-HEMS Services Status

Module	1st Level Control (CARTIF)
Status	Analysis in progress, preliminary tests have been conducted
Proposed Tester	WOODSPRING and/or CARTIF Further analysis required if pure black-box testing is possible
Interface(s)	<ul style="list-style-type: none"> Low-level interfaces with sensors/actuators/controllers of PCM, TCM, PVT Connection to HEMS
Approach/Strategy	Methodical search (malformed input, connection problems), testing for compliance to project-specified standards regarding possible MiniStor states and operation modes.
Entry criteria	<ul style="list-style-type: none"> Configuration/program needs to be ready to be installed on PLC device HEMS interface need to be completed for manual mode Otherwise, ready for Design phase
Exit criteria	Complete when planned tests have been executed

Table 12 1st Level Control Status

Module	2nd Level Control (CARTIF)
Status	Analysis to be started
Proposed Tester	WOODSPRING
Interface(s)	HEMS platform
Approach/Strategy	Testing tolerance for anomalous data, checking for compliance to project-specified standards regarding possible control states and operation modes.
Entry criteria	<ul style="list-style-type: none"> Preliminary implementation required to be able to start Otherwise, tests could be conducted incrementally
Exit criteria	Complete when planned tests have been executed

Table 13 2nd Level Control Status

Module	KPI Calculations (HSLU)
Status	Analysis to be started
Proposed Tester	WOODSPRING
Interface(s)	HEMS platform
Approach/Strategy	Analytical testing of correct implementation of project-specified KPIs, also checking tolerance for anomalous data.
Entry criteria	Need a list of KPIs that are to be calculated continuously
Exit criteria	Complete when planned tests have been executed

Table 14 KPI Calculations Status

Module	PCM, TCM, PVT (SUNAMP, CNRS, ENDEF)
Status	Pending, identity of technology providers uncertain
Proposed Tester	WOODSPRING and/or relevant technology provider

Interface(s)	1 st Level Control PLC
Approach/Strategy	<ul style="list-style-type: none"> • Need to be clarified if digital control logic is included with these devices • Need to be clarified if there could be safety concerns with independent testing
Entry criteria	Technical specifications should be known to conduct further analysis
Exit criteria	

Table 15 PCM, TCM, PVT Status

5.4 Acceptance Test Plan

It is important to assess the quality of behavior and capabilities regarding the system as a whole, especially from the perspective of end users. Incorporating feedback from acceptance tests can establish a confidence in the user interface and other software modules, before deployment of MiniStor to the demo sites.

5.4.1 Activities

Regarding the acceptance tests, WOODSPRING has accepted all the major responsibilities. Similarly, to integration tests, progress monitoring and controlling duties would involve CERTH as Task Leader, with participation of CARTIF as Work Package Leader.

Representatives of demo site owner organizations will be kindly asked to take part in the execution phases. Also, other interested colleagues in the project consortium are welcomed to join as a tester.

5.4.2 Entry and Exit Criteria

Execution of the tests shall be started as soon as the UI for the HEMS platform will be accessible remotely, and synchronization of monitoring data is solved in relation to the Sopron demo building (which is managed by WOODSPRING). The latter is required to be able to evaluate the user interface in relation to the actual observed conditions.

During the early analysis stage partial coverage from data collection will be considered i.e., before all the associated modules are ready.

A set of appropriate exit criteria must also be set up for the tests. An ultimate goal would be to improve the system until all participant testers rate the package favorably in every way, but given that at least some of the rating system will inherently be subjective, good threshold values must be devised for the stop condition.

5.4.3 Test Techniques

We are aiming to conduct both User Acceptance Testing and Operational Acceptance Testing. In the case of WOODSPRING, they also act as system operator of the Sopron demo site. It is also planned to include both functional and non-functional attributes in the tests. Tests would be executed manually by people with different backgrounds and perspectives.

Tests will be based on user stories that were documented in Work Package 7 of the project. Test reports will be recorded as questionnaires with response types such as checklists, scorecards with "1-to-10" rating type items, Likert scale and free text.

As part of the Operators Acceptance Testing, such responses are proposed to be investigated in the analysis phase. Consideration will also be made to include in this phase testing of the AR/VR Tool, as it is expected to play an important role in the MiniStor hardware installation process.

5.4.4 Schedule

The schedule for the first testing cycle is presented below.

Activity	Proposed Date
Analysis	Dec 2020 – Jun 2021
Design	Jun 2021
Implementation	Jun 2021 – Jul 2021
Execution	Jul 2021 – Aug 2021
Completion	Aug 2021

Table 16 Current schedule for acceptance tests

5.4.5 Status of Analysis

In this section we present our approach in relation to the ongoing first analysis phase of acceptance tests. During the early analysis stage, it was found that one or more initial acceptance test rounds provide valuable feedback before actual installation of MiniStor. This has the following two benefits:

- Early detection of any major issue in relation with the user interface allows timely correction. Even during the installation phase, operators and technical staff need to consult the UI to check recent behavior.
- Users can operate in 'safe mode' with the monitoring equipment installed but without actual machinery to control. In this mode there is no worry of causing any harm to the system or accidentally incurring into large bills. This mode can help demo site representatives getting used to the system, for example. It also provides an opportunity to include a larger set of subjects for testing, and to present the UI as a "simulation environment" to potential users who will not be available to participate after installation of the main equipment.

The main goal of any user acceptance test is to evaluate if the product conforms to specified user requirements. Earlier in the MiniStor project, as part of WP7 the document "**D7.1 Design evaluation through user validation in pre-demonstration site**" was created in order to capture these requirements. Using well-established concepts from Agile software development, the requirements were formulated as either user stories or use cases. These provide the backbone for the questionnaires that are to be created through acceptance testing.

Our aim is, that by the end of all software development efforts, users would feel that their main functional requirements are met. For the first testing round though, we must select relevant items from the list to include in a questionnaire, as we already know the answer to those needs related to unimplemented features. Also, we do include complex use cases in the test until at least the 1st level control system and related UI setting on the platform are operational.

The following template was used in D7.1 to formulate user story statements for requirements:

"As a [type of user], I [want to], [so that]."

Now we turn these into questions:

"As a [type of user], can I [do that], [so that]?"

With regard to already implemented functionality, we propose the following questions to include in the first round of acceptance tests.

	User Type	Epic	User Story
1.	Resident/Building Manager	Monitoring	As a user, can I view the temperature of each room in my apartment, so that I can decide when to turn on the heating?
2.	Resident/Building Manager	Monitoring	As a user, can I see the temperature or humidity data in the past? E.g. in a given day, so that I can monitor the conditions of my building.
3.	Resident/Building Manager	Monitoring	As a user, can I see the energy consumption of the building, so that I can monitor its behavior?
4.	Resident/Building Manager	Monitoring/Data Analytics	As a user, can I see the monitoring data in the form of graphs so that it is easier to observe the state in a certain period?
5.	Resident/Building Manager	Monitoring/Data Analytics	As a user, can I choose the period for which I would like to see the data?
6.	Resident/Building Manager	Monitoring	As a user, can I see the status of my building's conditions in a summary, so that I can easily grasp the overall picture of its condition?

Table 17 Questions for first round of acceptance tests

As new functions are gradually implemented in the software system, this set of questions can be extended with items related to other epics, like Control, Notifications, KPIs, Prediction, etc.

During the first tests, it can be measured how long it takes for a typical test subject to thoroughly evaluate each question. This information could be used as an additional entry criterion for subsequent testing rounds. This will be useful in order not to ask the same subjects too often to repeat a mostly identical test over and over again.

6 Components Integrated with the IoT platform

In the following sections, we provide detailed information about the progress and installation status of relevant hardware and software components as well as the development tools that are necessary to provide real-time monitoring data for the IoT platform.

As part of the effort to demonstrate the capability of MiniStor to be accommodated in existing environments as well as in newly constructed buildings, it was important to take into account the IT infrastructure and any environmental monitoring capability that might have already been present at each demo site. In coordination with demo site representative partners, it was established that in two of the demonstration cases the existing building management system can be used to receive all additional measuring devices needed. In the same sites, the system can also be connected to the IoT platform directly. In order to outline why certain development activities are necessary in some cases but not in others, we summarize relevant pre-existing infrastructure in the following table.

Demo site	New building	Existing local monitoring system	Local gateway
Cork	No	No	RaspberryPi to be installed
Kimmeria	No	Yes	No, direct connection
Santiago de Compostela	No	Yes	No, direct connection
Sopron	Yes	No	RaspberryPi to be installed
Thessaloniki	No	Yes	RaspberryPi already present, with data transfer software

Table 18 Summary of pre-existing monitoring systems at the demo sites

6.1 Tools for source management, development support system and bug tracking

To support a swift development process, toolkits are required for the organization of program sources, organization of development activities / sprints and handling of potentially identified problems and bugs.

In this project two platforms have been selected due to previous experiences of the development team as well as the assessment of the different options in Table 19:

- *Gitlab*⁵
Gitlab is an open source, web-based Tool that supports development operations (DevOps) during the lifecycle of a project. As the name suggests, it is based on the git technology and features also wiki-functionalities, issue tracking, continuous integration and deployment pipeline features that support the development team in the integration, but also testing and deployment of the developed tools.
- *Openproject*⁶
To ensure an on-time completion of the tasks as well as facilitate the continuous overview over the

⁵ <https://about.gitlab.com/>

⁶ <https://www.openproject.org/>

ongoing activities, the tool Openproject has been chosen. This tool offers all required functionalities for management of the (development) project.

Name	Description	Pro	Cons
CVS	Well established framework for source code organization and development tracking https://savannah.nongnu.org/projects/cvs	Successfully applied in industry for decades	Last official release in 2008
Git	One of the industry standards for code organization and management https://git-scm.com/	One of the most used software	Bug tracking and wiki functionality not included in tool.
Github	Online platform for source organization and management based on the git technology. Offers also the tracking of issues as well as bugs. https://github.com/	Wide user basis, online solution, worldwide access	Not open-source, local hosting of servers not possible
Gitlab	Open-source tool for the organization of code and issues based on the git technology. https://about.gitlab.com/	Wide user basis, online solution, local hosting possible, open-source tool	Project management tasks (time plans) challenging to implement
OpenProject	Open-source project management software for the organizational aspects of a project. openproject.org	Simple to use, offers all required functionality, can be locally installed	
Monday.com	Monday.com	Intuitive user interface, very wide user basis	Online only solution, no local hosting possible

Table 19 Overview over common source organization and development organization tools.

6.2 Software Integration

In the following paragraphs, we report the status of any software components that are required to be developed for integration purposes.

6.2.1 Use case scenarios, assets and relevant software

In this subsection, we describe the main use cases for each pilot site, the supported hardware assets and the current integration status of the hardware components with the platform in regard with their corresponding software components.

The pilot site in Thessaloniki constitutes a demonstration platform for residential buildings and offices. For the case of Sopron, the dedicated space that will be served by MiniStor is a residential area. For the case of University of Santiago de Compostela, the pilot site is a university apartment that is going to be used in a family setting. The demonstration site in Cork is a residential house. Finally, for the case of Kimmeria, the dedicated space is dormitory student apartments inside the campus of DUTH.

Currently, the demonstration sites have installed sensors to monitor indoor and outdoor conditions. Those sensors include electrical energy meters, room sensors for measuring temperature and humidity, environmental sensors to monitor weather conditions, gas and air flow meters (only for Cork and Sopron) and the respective data-logging hardware to save the sensor measurements. Note that for the University of Santiago de Compostela, sensor installation is on-going as they joined the project in M12. Also, the MiniStor system (PVTs, battery etc.) is going to be installed to all pilot sites and it is also going to be integrated with the platform. More details for the hardware equipment are described in D6.1 [7].

The platform is able to support all pilot sites with their respective assets and the integration for all sites is on-going. Also, the platform is built to fulfill identified stakeholders' requirements and the possible use cases as defined in D.7.1, Thessaloniki's and Sopron's pilot sites have already been integrated with their existing assets, while the rest of the pilot sites are also supported. Also, the assets have been defined for each demonstration site and they also have been mapped to relevant endpoints on the API.

6.2.2 Local RaspberryPI (RPi) Gateway development

At two of the demo sites, in Cork and Sopron, there was no existing environmental monitoring or smart building management systems installed. This condition required to add a local gateway device in order to provide facilities for the following functions:

- Data readout for measurement devices without direct IP interface (e.g. RS-485)
- Short-term data storage in case of temporary Internet connectivity issues
- Local data backups in order to provide additional long-term data security
- On-site conversion to JSON-format for use with the REST API described in section 3.3
- Providing secure VPN connection for remote configuration

In order to fulfill mentioned roles, it was decided that a small form-factor industrial computer was required, for which the RaspberryPi 3 B+ SBC (Single Board Computer) was selected.

The following table describe the software-related activities for the local RPi.

Activity/Module	Status	Scripting language
Configuration of Elvaco CMe3100 M-Bus data logger	Complete	
Readout of CMe3100 through REST API	Complete	BASH
Readout of DeltaOhm weather station through Modbus RTU	Complete	BASH
Readout of Airvent HVAC unit through Modbus TCP/IP	Complete	BASH
Readout of Davis weather station through Modbus TCP/IP	Complete	BASH
CSV-to-JSON and XML-to-JSON conversion	Complete	BASH
JSON file upload through REST API	Complete	BASH
CERTH's and DUTH's monitoring sensor data upload through REST API	Complete	Python
Configuration of TMPFS (Temporary File System)	Complete	BASH

Table 20 Status and scripting language of different activities and modules for the local RPi

6.2.3 Backups

In case of those demo sites where the local RPi gateway is used, there are a couple of implicit data backup mechanisms in operation:

- The CMe3100 data logger automatically keeps every measurement until its storage medium is full, after this it operates in 'First-In-First-Out' mode, providing a rolling window and keeping only the recent data. Window width for example in the case of Sopron demo site is estimated to be around 30 days.
- On the RPi, the device readout scripts produce CSV or XML files which are kept even after conversion and upload.

There are a couple of development activities planned in relation of the improvement and extension of these measures. It is worth mentioning that any procedures of such type will strictly follow the applicable data protection guidelines.

Activity/Module	Automatic download of central IoT data for back-up
Status	Planned
Scripting language	BASH
Description	In order to provide a third location for safekeeping data from possibly other demo sites as well, central data storage will be periodically accessed through a secure channel (REST API) to back-up the data
Applicable demo sites	All

Table 21 Automatic download of central IoT data for back-up status

6.2.4 Alerts

As the environmental monitoring period was planned to start before development of the IoT interface could progress to a stage where functions as reports and alerts could be finished, requirements have arisen to provide a low-level alert system. The goal of this function is to provide means for demo site representatives to be informed as soon as possible and at the development stage, of any hardware or software component malfunction, or prevention of data recording. The following software development actions have been proposed and started:

Activity/Module	Status	Scripting language	Description	Applicable demo sites
Low-level alert module for RPi Gateway	Planned	BASH	Periodic check on the RPi of such conditions as e.g., there are adequate free storage space, VPN connection is active, and central IoT server is accessible. Sends email message if a persistent problem is encountered. This will only work if Internet access of the RPi itself is not prevented.	Cork, Sopron
Mid-level alert module for checking if data uploads are working	Manual mode completed, automation in progress	BASH	From a third location, the script checks periodically through the REST API if there are recent data uploaded to the IoT platform for every measurement device in relation of a certain demo site. If some 'latest values' are too old, an email alert is generated.	All

Table 22 Activity/Module for low- and mid-level alert module for RPi Gateway status

Conclusions

This document describes current functionalities of the initial implementation of the IoT platform and its user interface that were developed within Task 5.3, allowing users to monitor performance and control the system. These functionalities will optimize system usage through an intuitive user interface that receives inputs from the monitoring system. The deliverable describes the methodology that was followed to reach its current architectural and development state as well as future directions. It also includes the functions and actions that the IoT platform offers as well as a description of the data layer. This data layer includes the architecture of the data mechanisms as well as documentation for the API endpoints.

In addition, testing methodologies were described, as defined in Task 5.4. Their aim is to evaluate interfaces and functionality of components and ensure the organized and seamless integration of both hardware and software components. The main pages of the user interface are programmatically completed according to the existing mock-ups. Thessaloniki's and Sopron's pilot sites are integrated with the user interface and the platform is able to seamlessly support all demo sites that have installed sensors with a simple and fast procedure. For the next iteration of the deliverable, the aim is to further enrich the functionalities based on user feedback and continue the testing methodologies along the way to ensure the proper development of the system.



References

- [1] "MiniStor - Deliverable D7.1 Design evaluation through user validation in pre-demonstration site".
- [2] <https://www.digite.com/agile/scrum-methodology/>
- [3] <https://www.wrike.com/project-management-guide/faq/what-is-agile-methodology-in-project-management/>
- [4] "MiniStor - Deliverable D1.7 Data management plan".
- [5] <https://www.lbmc.com/blog/three-tenets-of-information-security/>
- [6] Cai, L., & Zhu, Y. (2015). The Challenges of Data Quality and Data Quality Assessment in the Big Data Era. Data Science Journal, 14, 2. DOI: <http://doi.org/10.5334/dsj-2015-002> (Available at <https://datascience.codata.org/articles/10.5334/dsj-2015-002/>)
- [7] "MiniStor - Deliverable D6.1 Design of the monitoring system and KPI definition."

