Cluster-Edge architectural scheme

Project acronym: COLLECTiEF

Project title: Collective Intelligence for Energy Flexibility

Call: H2020-LC-SC3-EE-2020-2

Disclaimer

COLLECTiEF project has received research funding from European Union's H2020 research and innovation programme under Grant Agreement No 101033683. The contents and achievements of this deliverable reflect only the view of the partners in this consortium and the European Commission Agency is not responsible for any use that may be made of the information it contains.

Project no.	101033683
Project acronym:	COLLECTiEF
Project title:	Collective Intelligence for Energy Flexibility
Call:	H2020-LC-SC3-2018-2019-2020
Start date of project:	**01.06.2021**
Duration:	48 months
Deliverable title:	Cluster-Edge architectural scheme
Deliverable No.:	D3.1
Document Version>	1
Due date of deliverable:	31.08.2022
Actual date of submission:	31.08.2022
Deliverable Lead Partner:	Partner No. 4, ENERGY@WORK SOCIETA' COOPERATIVA A R.L.
Work Package:	3
No of Pages:	106
Keywords:	Architectural system specification, architectural components, Field communication, Thermal optimization, Flexibility management, Human Building Interface and Visualization, Collective Intelligence, System Design, UML

Name	**Organization**
Giuseppe Mastandrea	E@W
Marco Antonio Insabato	E@W
Luigi D'Oriano	E@W
Giuseppe Rocco Rana	E@W
Runar Solli	EM

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Jens Brage	NODA
Daniel Nilsson	VIRTUAL
Panayotis Papadopoulos	CyI
Muhammad-Salman Shahid	CSTB – G2ELab
Alfredo Astori	LSI - LASTEM
Seyedmohammad Hosseini	NTNU
Amin Moazami	NTNU
Vahid Nik	ULUND
Antonio Luparelli	CETMA

Dissemination level

PU	Public

History

Version	Date	Reason	Revised by
0.1	28.06.2022	Deliverable structure draft – First version	Giuseppe Mastandrea E@W
0.2	13.07.2022	Methodology defined	Giuseppe Mastandrea E@W
0.3	20.07.2022	First Version	Giuseppe Mastandrea E@W
0.4	25.07.2022	Internal Review and second version	Giuseppe Mastandrea E@W, Luigi D'Oriano E@W, Giuseppe Rocco Rana E@W
0.5	03.08.2022	Partners Contribution	Runar Solli EM, Jens Brage NODA, Daniel Nilsson VIRTUAL, Panayotis Papadopoulos CyI, Muhammad-Salman Shahid CSTB-G2ELab, Alfredo Astori LSI-LASTEM, Seyedmohammad Hosseini NTNU, Vahid Nik ULUND
0.6	23.08.2022	Integration of Partners Contribution and third Version -Version Ready for the Peer Review	Giuseppe Mastandrea E@W, Marco Antonio Insabato E@W, Luigi D'Oriano E@W
0.7	29.08.2022	Document review	Antonio Luparelli CETMA, Jens Brage NODA
0.8	30.08.2022	Updated version	Giuseppe Mastandrea E@W

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

0.9	31.08.2022	Document approval	Amin Moazami NTNU
1.0	31.08.2022	Final Version	Amin Moazami NTNU, Giuseppe Mastandrea E@W

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Executive Summary

This deliverable D3.1 describes the first version of the COLLECTiEF architecture to be submitted at M15. The final version will be released at M24. The aim of this deliverable is the definition of system requirements and technical specifications which are crucial and fundamental steps for the successful design of the COLLECTiEF system. Particularly, the architectural components' dependencies and specifications as an outcome of the progress of the activities conducted in task T3.1 will be presented.

Initially, in Chapter 2 the methodological approach for the definition of the system requirements is presented by indicating the relevance of system requirements with the project use cases. The technical specifications were extracted through internal elicitation using appropriate templates. Their description is important, in order to map the way for successful integration and ensure the consistency of use cases and requirements.

For the sake of completeness, the Conceptual architecture of the COLLECTiEF system is presented at the beginning of Chapter 3 with some refinements. This is a high-level view of the overall architecture, describing the four major layers of the COLLECTiEF platform, namely, the **Field Layer**, the **Distributed Layer**, the **Cluster Layer** and the **Application Layer**.

The platform also comprises the Central Repository for secure storage of the data exchanged between the modules of the platform, with external interfaces through RESTful APIs and MQTTS.

Then, in Chapter 4, the structural view of the system is presented, by providing details on the different architectural components that deliver the system's functionalities. This view provides the system's decomposition into different components, demonstrating the dependencies among them, their interfaces, the data exchanges and their functionalities.

Chapter 5 focuses on the dynamic behaviour of the system, where the use cases are correlated with each architectural component. The way that each component acts within the use cases determines its functional requirements. The updates of components' dependencies are reflected in the structure of the respective UML sequence diagrams. The preliminary UCs description is reported in this document for the sake of completeness by using the appropriate template included in Annex 1.

The deployment view is described in Chapter 6 defining the physical environment, in which the system is intended to run including hardware requirements (e.g., processing nodes, network interconnections, etc.).

Finally, Chapter 7 presents, by using the appropriate template included in Annex 2, the detailed technical specifications of the COLLECTiEF core architectural components focusing on the functionalities, inputs/outputs, interfaces and data types. A detailed description of the concept of the user-friendly human-building interface for COLLECTiEF's Edge Node is presented in Annex 3.

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Table of Contents

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

List of Acronyms

AHU	Air Handling Unit
API	Application programming interface
BMS	Building Management System
BRIG	Borted Router + iGateway (Edge Node)
CI	Collective Intelligence
CO2	Carbon Dioxide
COLLECTiEF	Collective Intelligence for Energy Flexibility
DDC	Direct Digital Control
DR	Demand Response
DRY	Don't repeat yourself
GDPR	General Data Protection Regulation
HBI	Human Building Interface
HMI	Human Machine Interface
HTTP	Hypertext Transfer Protocol
HVAC	Heating, Ventilation and Air Conditioning
HW	Hardware
IEA EBC	International Energy Agency Energy in Buildings and Communities Programme
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
JSON	JavaScript Object Notation
KPI	Key Performance Indicators
L	Load
LAN	Local Area Network
M2M	Machine to Machine
MQTT(S)	Message Queue Telemetry Transport (Secure)
OCC	Occupant-centric control
PC	Persona Computer
PD	Participatory Design
PM	Particulate Matter
PPM	Parts Per Milion
RES	Renewable Energy System
REST	REpresentational State Transfer
RTU	Remote Terminal Unit
SD	Sequence Diagram
SOAP	Simple Object Access Protocol
SRP	Single Responsibility Principle
TCP	Transmission Control Protocol
UC	Use Case

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

UTC	Universal Time Coordinated
UV-A	Ultraviolet radiation
VAV	Variable Air Volume
VOC	Volatile Organic Compounds
Wi-FI	Wireless Fidelity
WP	Work Package
XML	extensible markup language

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

List of Figures

List of Tables

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

1 Introduction

The purpose of this deliverable, as the technical output of the project, is to present the first version of the system requirements and technical specifications for the COLLECTiEF system. The deliverable describes the steps and actions performed in the first 4 months of Task 3.1 and can be considered as a key input for the upcoming tasks in WP3 concerning the development and demonstration at a small-scale real environment in G2Elab of the COLLECTiEF hardware and software technologies. Throughout the document, the main requirements and specifications of the COLLECTiEF system are described in the scope of addressing the COLLECTiEF objectives and innovation potential.

1.1 Scope and objectives of the deliverable and relevance in the COLLECTiEF framework

In this deliverable the first version of the COLLECTiEF Conceptual Architecture as well as the integrated system specifications along with the system requirements will be described. This document provides a holistic view of the COLLECTiEF overall Architecture, its building blocks, components, interdependencies among components, and related constraints, such as development methodology and interfaces for data exchanges.

The concept of the architectural framework mainly focuses on deriving the specifications of the system's key components and their functionalities based on the discussions held among the project partners on the User Needs and Business Requirements. Following the basic design principles, the following aspects are addressed:

- Conceptual Architecture Design Process: within this part, an overall view of the COLLECTiEF architecture is presented comprising the components, the interfaces between them and the connections with the external interfaces.
- Functional and Technical Specifications of Architectural Elements/Modules: the objectives of this part are the followings:
    - To provide a high-level diagram of dependencies among the different parts of the framework.
    - To describe in detail the constraints of the system elements in terms of hardware and software resources, compatibility with standards, etc...

1.2 Structure of the deliverable

D3.1 "Cluster-Edge architectural scheme" consists of eight chapters, in which the first version of System requirements, dependencies and technical specifications have been described as follows:

- **Chapter 1** presents the general description of the scope and objectives of the deliverable.
- **Chapter 2** describes the methodology, which was followed during the architectural design in order to derive the functional and technical specifications of the COLLECTiEF system. It presents the basic concepts and principles of architectural design adopted to outline the different phases and the definition of the layers and architectural elements that constitute the COLLECTiEF system.
- **Chapter 3** presents the first version of the conceptual architecture of the COLLECTiEF system through a high-level diagram introducing the four main layers comprising the COLLECTiEF system.

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

- **Chapter 4** describes the structural view of the COLLECTiEF platform describing the different architectural elements/modules that provide the system functionalities. This section also presents the breakdown of the system into different components, demonstrating how each component performs the required functions.
- **Chapter 5** presents an analysis of the dynamic behaviour of the COLLECTiEF system through use cases and sequence diagrams. This dynamic view defines how the system actually works and what responses it gives to external or internal stimuli.
- **Chapter 6** shows the deployment schema of the COLLECTiEF system covering the hardware requirements of the architectural components and tools to be used.
- **Chapter 7** presents the system's detailed architectural elements specifications.
- **Chapter 8** provides the conclusions of the overall work.

Finally, the template used to describe the project UCs is included in Annex 1 the template used for the internal elicitation of requirements and technical specifications is included in Annex 2 and as already mentioned the detailed description of the concept of the user-friendly human-building interface for COLLECTiEF's Edge Node is presented in Annex 3.

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

2 Methodology

This section presents the approach and methodology followed to define the architecture. Task 3.1 started on M12 and it will run continuously until M24 so, in this document, we will present the whole methodological process that will be followed to define the final version of the COLLECTiEF system architecture, also considering the intermediate step in M15 for the definition of the first version of architecture that comprises the first consolidation of dependencies, inputs/outputs and specifications of architectural components.

Specifically, in this document this first version is described in detail and will act as a basis for the development activities in WP3. Starting from this first version, between the M15 and M24, the architecture will be updated in parallel with the development activities. Particularly, information concerning the interfaces between the components will be further specified based on the outcomes of technical developments. Also, the description of the whole platform in terms of architecture, modules, dataflow, processes, APIs specifications and interoperability issues will be further detailed.

Figure 1 presents the process for the system requirements elicitation until M24:

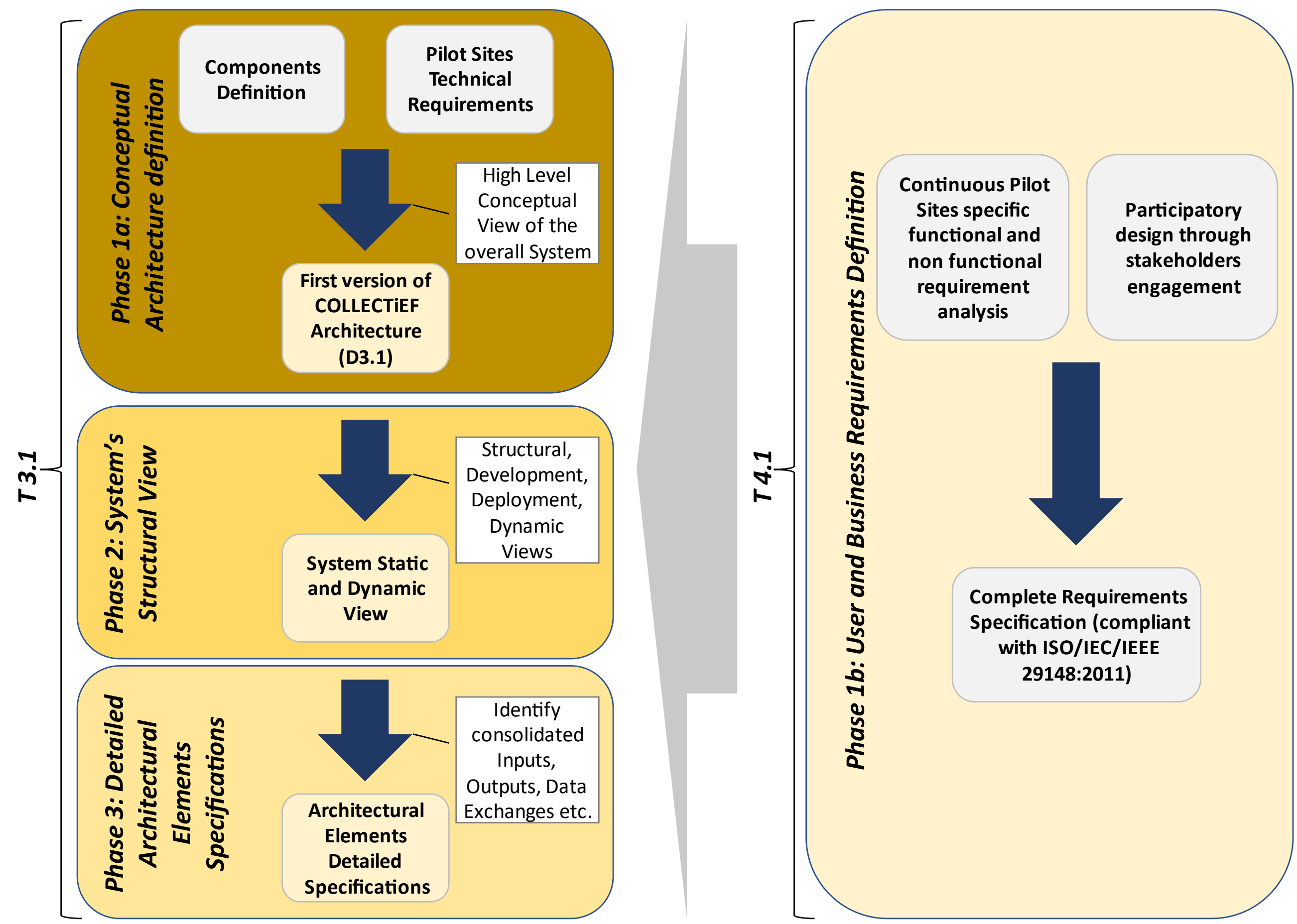


Figure 1 Architectural design approach and workflow

2.1 System Architecture Concepts and Design Fundamentals

The overall architecture of a system is the composition of different architectural system structures, which include software elements, the attributes and externally visible properties of those elements

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

along with the relationships and interfaces between them. It describes its different components and how they interact with performing the required functionality.

The representation of the conceptual architecture and its architectural elements allows communication between all interested parties who are interested or concerned in the realization of the system. The definition of the overall system structure and the orchestration between the architectural elements are fundamental parts of the system development process, as architectural design decisions have a profound impact on all the development work that follows and on the execution of the development activities. Finally, all the components that comprise the system must take into account the concerns that arise during the business and user requirements process with the effective involvement and involvement of the key stakeholders.

2.1.1 Design Principles

Following the basic design principles, the architecture is open and modular, so that all potential users can use the functional part of the architecture. Furthermore, the architecture must be as technology-independent and standards-based as possible and promote (when possible) the use of generic and standardized solutions for which several key technologies are available (open source, commercial, etc.).

Based on the static and dynamic models, a number of **key design principles** have been defined and specified to ensure that architecture designers minimize maintenance costs and requirements and promote extensibility, modularity and maintainability. These can be classified as follows:

- **Separation of concerns**, which emphasizes that the overall system/application should be broken down into distinct features with as little overlap of functionality as possible. The ultimate goal of this principle is, on the one hand, to minimize the points of interaction and, on the other, to ensure greater cohesion and low coupling to reduce dependency among components.
- **Single Responsibility Principle (SRP)**, which outlines that each architectural element (e.g., core component of the system) must be responsible only for a specific feature or functionality, or even the aggregation of cohesive functions.
- **Principle of Least Knowledge** (or **Law of Demeter**), which defines that an architectural element (e.g., component or object) should never know or have direct access to the internal details of other architectural elements (e.g., components or objects).
- **Don’t repeat yourself (DRY)**, which refers to the principle of avoiding repeating the same functionality or intent in more than one architectural element of the system at the design stage by considering the possibility of replacing it with abstractions or using data normalization to avoid redundancy. Therefore, according to this principle, common functionalities are addressed in more general architectural elements or components, which can be used by each separate element to "access" or "provide" the required functionality.
- **Minimize initial design**, which emphasizes that designing more features and methods than ones required for the system at design time should be avoided. This principle mainly refers to the early stages of the architectural development process, when the project is likely to change over time. Therefore, architectural designers and developers must avoid large-scale design and the potential implementation of components in premature stages.

2.1.2 Static and Dynamic Structures

The key output of the architectural elements design process is the detailed definition of the conceptual architecture and the components that constitute the system, that is, the system structures

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

and its exposable attributes and properties. The system structures are divided into two complementary categories, static (design-time orchestration) and dynamic (runtime orchestration):

- The **static structures** mainly refer to the phase of the design of the architectural elements of the system (objects, components) and the way in which they fit together internally. The static arrangement of the architectural elements depends on the actual context of use and provides information such as associations, relationships or connectivity between them. For example, relationships define how data elements (input or output) are connected to each other. In hardware, relationships provide the physical interconnections required between the hardware components and the subsystems comprising the overall system.
- The **dynamic structures** of a system show how it works during its use, according to different scenarios and use cases defined, including how each component acts within them. Therefore, the dynamic model and the structures of the system define its runtime architectural elements and their interactions with internal or external feedback. Internal interactions refer to information flows between architectural elements and their parallel or sequential execution of internal tasks, including the potential expression of the effect they can have on information.

2.1.3 End-users and Stakeholder requirements' perspective

The COLLECTiEF project expects to adopt a participatory design (PD) process in Task T4.1 with the aim to involve the relevant stakeholder groups in the process for the requirements definition. This will facilitate the coordination between user and business requirements definition and functional requirements and technical specifications definition. This approach will be based on iterative cycles concerning capturing end-user and business needs as a reference point for the overall design, implementation and evaluation process.

Particularly, since the requirements elicitation is one of the most important parts of the system life cycle processes and must be tackled with extreme care, an iterative approach to elicit and assess the requirements until M24 will be followed. This iterative approach will be applied as depicted in Figure 2:

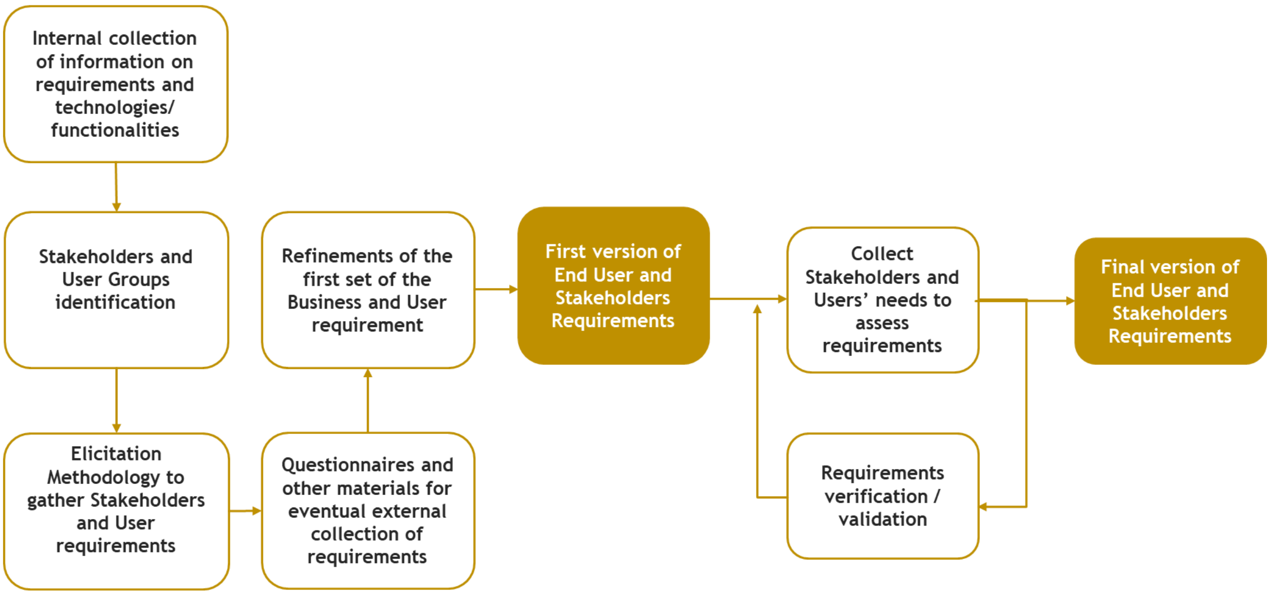


Figure 2 Methodological Flow hypothesis for the Requirements' Elicitation

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Within the COLLECTiEF project, this process will start from the definition of the first set of requirements through the study of the literature, the internal interrogation of the pilots by the internal technology providers continuing with the definition of the methodologies for identifying the stakeholders and eliciting the requirements in order to update the first set. Project presentation and survey/questionnaire for the external stakeholders will be used to define the requirements continuously during the project involving more and more stakeholders up to M24.

2.1.4 Architectural Views

In the context of COLLECTiEF, the 4+1 architectural view model [1] has been used to present concurrent views.

The concept is illustrated in Figure 3:

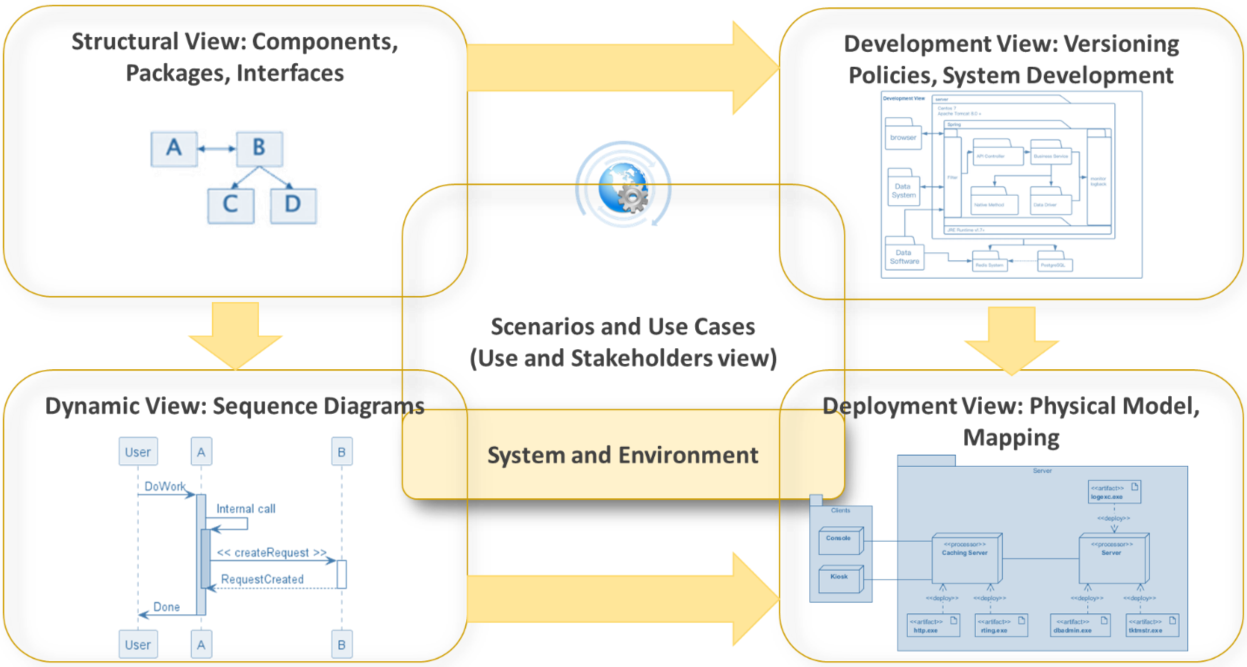


Figure 3 Architectural View 4+1 model

The 4 + 1 view model describes the software architecture using five concurrent views, each of which addresses a specific set of problems: the logical view describes the object model of the project, the process view describes the concurrency and synchronization aspects of the project; the physical view describes the mapping of the software to the hardware and shows the distributed aspects of the system, while the development view describes the static organization of the software in the development environment.

2.1.5 Architectural Elements Perspectives

Conventional views and approaches provide meaningful information in the process of deriving architecture and in defining various architectural structures. However, to expand the modularity, reliability, and credibility of the system at the design stage, it is helpful to outline and consider specific quality properties during the final stages of the architecture definition process. To define the architectural elements of COLLECTiEF, their dependencies and the respective architectural paths, the architectural perspectives are also taken into consideration, which are analogous to a point of view, as they have been described in detail for the structural/functional views, development dynamic

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

and implementation. In this report, several quality properties are considered for all architectural elements of the system, as indicated in Table 1:

Table 1 Quality properties and perspectives for architectural elements

Perspective	***Desired Quality***
General Purpose	
Performance and Scalability	*The ability of the system as a whole, including its architectural elements, to perform predictably the required performance that meets system requirements and is capable of handling increased volumes of information processing.*
Availability and Resilience	*The ability of the system as a whole to be fully or partially operational as and when required and to effectively manage failures at all levels (hardware, software) that could potentially affect system availability and credibility.*
Security	*The ability of the system to reliably and effectively control, monitor and further verify whether defined policies are met (e.g., what actions on which assets/resources) and to be able to recover from failures due to security-related attacks.*
Evolution	*The ability of the system and its architectural elements to be flexible enough in case of unexpected changes during the implementation, deployment and/or installation process.*
Additional Perspectives to cope with COLLECTiEF non-functional requirements	
Maintenance	*The system's ability to comply with coding guidelines and standards. It also includes features that must be provided to support the maintenance and administration of the system during its operational phase.*
Privacy & Regulation	*The ability of the system and its architectural elements to be compliant with the national and international laws, the GDPR policies and the other rules and standards.*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Usability	*The ease with which the key stakeholders of a system are able to work effectively and interact with it in an intuitive and user-friendly way.*

For each of the above perspectives, the importance of the four viewpoints of the COLLECTiEF framework can vary and the benefits of addressing them are essential to provide a common sense of concern that will guide the process of defining architectural elements and their subsequent development and deployment, integration and validation activities. To ensure that the COLLECTiEF architectural system based on the defined architecture will meet the functional and non-functional requirements, the perspectives proposed will be considered. These perspectives could be modified or enriched by the partners during the development phase based on the characteristics of the components.

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

3 Conceptual Architecture

This chapter provides the COLLECTiEF conceptual architecture overview by introducing the main COLLECTiEF platform layers and sublayers along with its architectural components. COLLECTiEF's vision is to develop, validate and deliver distributed collective intelligence that implements collaborative problem solving and decision making to enable optimal energy management by leveraging the potential for flexibility in a scalable manner within buildings, neighbourhoods and urban systems. During the project, new functionalities and services will be researched and examined using the principles of the Internet of Things (IoT), the concepts of Demand Response programs and the Collective Intelligence paradigm. Figure 4 presents the COLLECTiEF conceptual architecture:

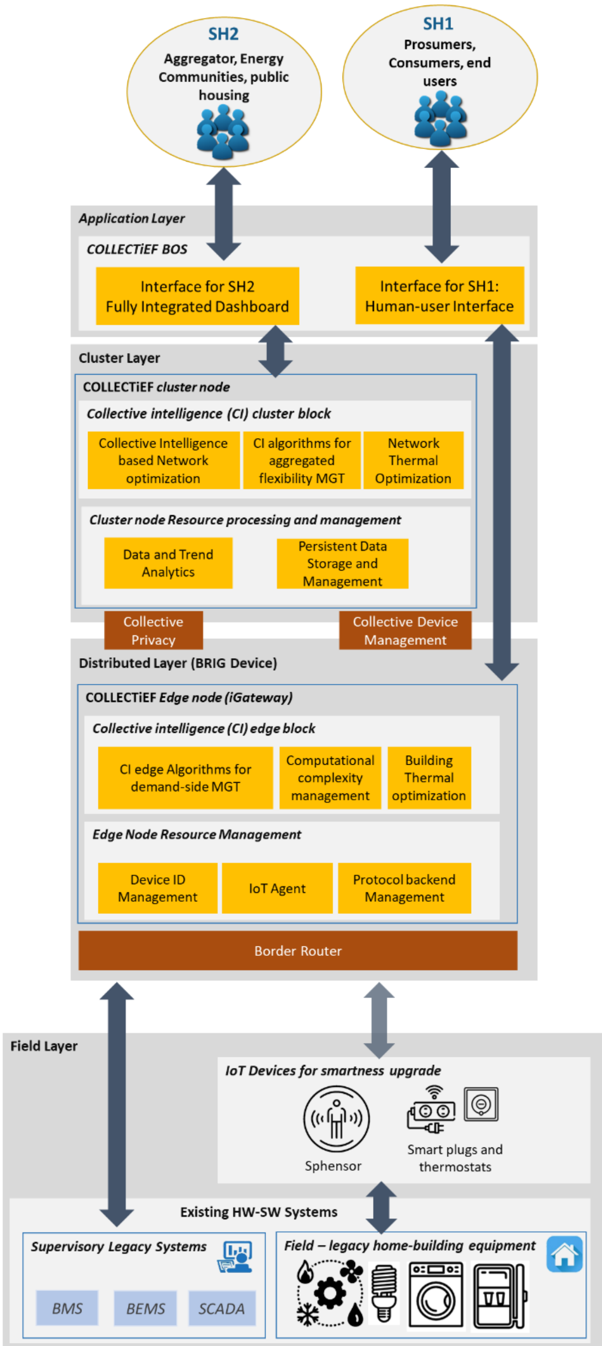


Figure 4 COLLECTiEF Conceptual Architecture

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

The main layers and sub-layers of the COLLECTiEF platform are described in brief below:

- The first layer of the architecture is the **Field Layer**, which is composed of (i) existing HW devices already in place and (ii) new IoT devices and systems needed to embrace legacy barriers by interconnecting existing HW devices and systems. The use of IoT devices provides access to real-time data from pilot sites by providing environmental data, the status of systems and electrical loads, and electrical measurements. This level is interfaced with the Border Router to forward the necessary information in real-time to the higher levels of the platform, in order to allow the functional components of the architecture to perform their own analyzes and calculations. The exchange of information is based on open communication specifications (based on MQTTS [2] and RESTful API) that realize Machine2Machine (M2M) communication through which data, information and actions are sent to the appropriate field device or level top of the platform.
- The **Distributed Layer** is the fundamental part of the conceptual framework. Physically, it relies on one or more Edge nodes implemented on a low-cost Raspberry PI 4 processing board. It includes all the components and mechanisms needed to support distributed Collective Intelligence. In particular, the Edge Node has the task of ensuring interoperability with field devices and systems and is responsible for the execution of the processing blocks of the edge CI, which include the main processes necessary to execute lightweight algorithms for flexibility management and thermal optimization services.
  This layer also puts the user in the loop through locally processed data to provide data insights, active notifications and recommendations (e.g., to apply a specific thermal strategy) to the end user through the human-building local interface that runs on the edge node.
  Particularly, this layer includes three hierarchical connected sub-layers that will be deployed on RaspBerry open board and will be named **BRIG**, notably, they are the following:
  - **Border Router:** which collects the data from the field and passes them to the edge node.
  - **Edge Node Resource Management:** which manages the field data for the implementation of the algorithms deployed at the edge level.
  - **Collective Intelligence (CI) Edge Block:** which applies the setpoint controls to the building temperature depending on energy demand and comfort.
- The **Cluster Layer** includes the COLLECTiEF Cluster Node, in which we find components for aggregated CIs and thermal optimization. This layer will be in charge of the network optimization by implementing the algorithms for the management of the aggregated flexibility for DR-based services that aim to achieve different objectives (increase energy saving, self-consumption; maximize market participation, etc.) and by applying mechanisms for thermal network optimization including dynamic supply temperature to maximize the thermal comfort of the user. In the Cluster Node, we can also find a data and trend analysis module composed of components useful for performing visual and data-driven analysis to be shown through the high-level COLLECTiEF fully integrated dashboard. In addition, Persistent Data Storage and Management provides all the necessary tools for context and data history management while the control and communication between these components will be enhanced by Collective Privacy and Collective Fleet Management.
- The upper layer, the **Application layer** contains accessible and easy-to-use HMIs (e.g., accessible by mobile phone through lightweight visualizations). Particularly, one for end-users and one for operators that enables vertical collaboration within the COLLECTiEF

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

architectural framework. The main purpose of this layer is the visualization of the output data and give to the end user the possibility to interact with the field. Bidirectional data flow is performed between the platform and the front-end layer, since several decisions of stakeholders are based on the provided results from the components of the platform.

As mentioned above, during the bottom-up process of the architecture definition, all the technology provider partners were identified. The main purpose of this phase was the identification of the architectural components that should be developed and the corresponding partner/s. During the first round of information collection, a basic template was created and circulated with requested information concerning main functionalities, dependencies, inputs needed and outputs provided. The list of architectural components along with the assigned tasks and associated partners responsibilities is presented in Table 2.

Table 2 List of identified architectural components, assigned tasks and partners responsibilities

Component	Related Task	Responsible partner	Contributing partners
Border Router	T3.2	LASTEM	E@W
Edge Node Resource Management	T3.2	E@W	LASTEM, CETMA
CI edge algorithms for demand side management	T2.2, T2.4, T3.2	ULUND	CyI, NTNU; E@W
Building Thermal Optimization	T2.2, T2.4, T3.2	NODA	CyI, E@W
Collective Privacy	T3.3	NODA	E@W
Collective Device (Fleet) Management	T3.3	NODA	E@W
Collective Intelligence based Network Optimization	T2.1, T2.4, T3.3	ULUND	NODA, NTNU, VIRTUAL
CI for aggregated flexibility management	T2.1, T2.4, T3.3	ULUND	NODA, NTNU
Network Thermal Optimization	T2.1, T2.4, T3.3	NODA	ULUND, NTNU
Cluster Node Data and Trends Analytics	T2.1, T2.4, T3.3, T3.4	NODA	VIRTUAL, CETMA
Human Building Local Interface	T3.4	VIRTUAL	CyI, CSTB
Fully Integrated Dashboard	T3.4	VIRTUAL	CyI, CETMA, NTNU

For all the components, an updated detailed description template is provided in Chapter 7 including the currently known technical specifications. The following chapter provides the first version of the structural view of the COLLECTiEF architecture and presents the main functionalities and dependencies for each architectural component.

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

4 Structural – Functional View

4.1 Overall Structural View of COLLECTiEF architecture

The structural view presents the different architectural elements that provide system functionality to end users. As part of this vision, the individual components of the system, their dependencies and high-level interfaces concerning the other components have been identified and defined. The functional system model includes the following elements:

- **Functional components** are clearly defined parts of the system that have specific responsibilities, perform distinct functions, and have well-defined interfaces that allow them to be linked with other components.
- **Dependencies** are channels that indicate how the functions of a component can be made available to other components. An interface is defined by the inputs, outputs and semantics of the provided operation/interaction.
- **External (third party) entities** are connectors (described as dependencies) that represent other systems, software programs, hardware devices, or any other entity that communicates with the system.

The following sub-sections introduce the defined architectural components with their main functionalities and the dependencies on the other components. Moreover, the necessary information for the initial definition of the detailed modules interfaces and APIs are provided acting as a basis for the definition of the final version of the architecture that will be consolidated until the end of the activities related to the Task T3.1.

Figure 5 depicts the COLLECTiEF architecture overall structural diagram with all the identified high-level dependencies that represent the main flows of information.

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

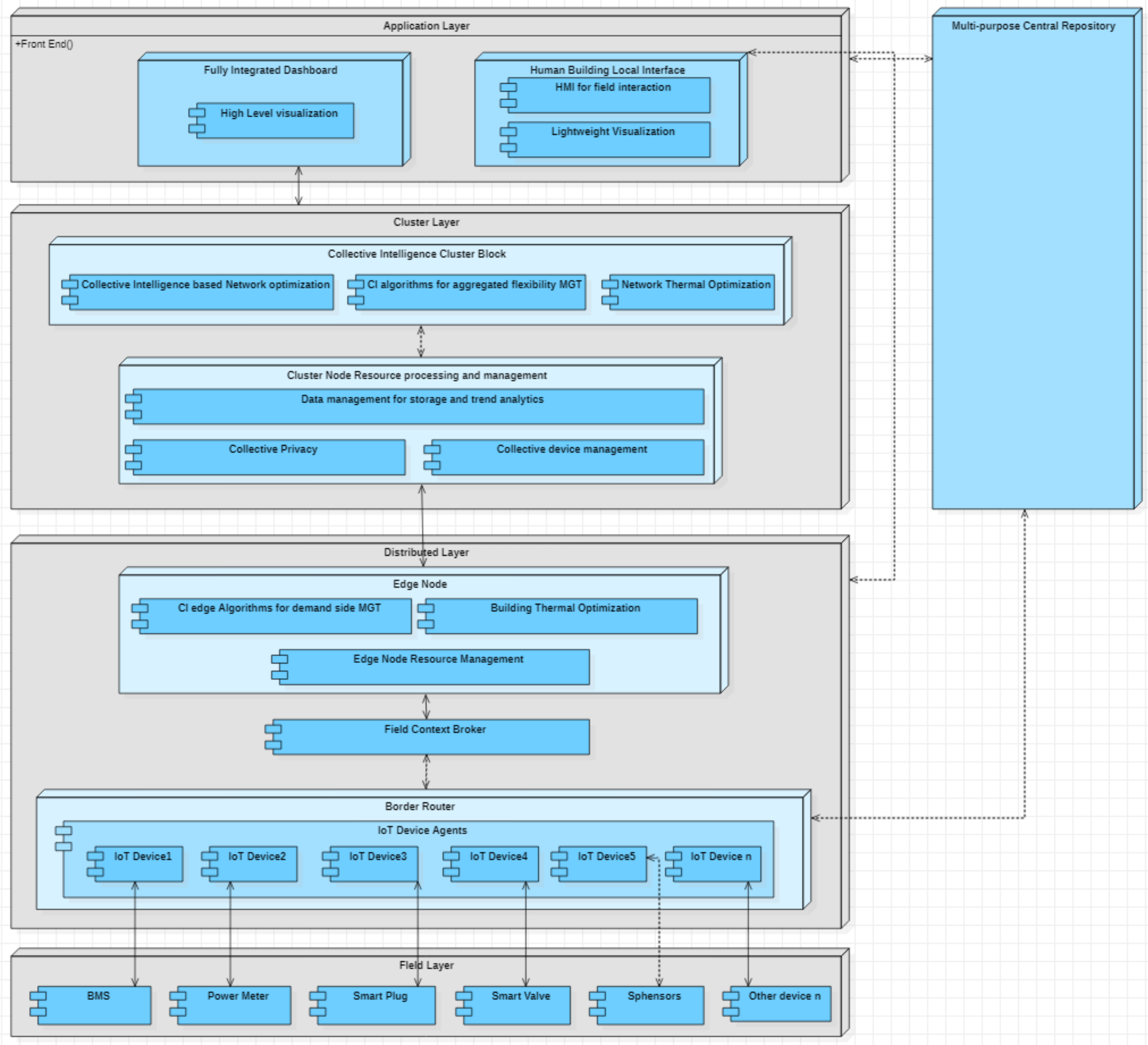


Figure 5 COLLECTiEF overall structural view

4.2 Field Communication Layer

This layer is the bottom layer of the COLLECTiEF system and refers to the communication interfaces with the field devices. In particular, communication with the field will happen through specific agents that will manage the communication with every single category of device.

These agents will run inside the Border Router that constitutes the basic interface with the physical world and performs primary information processing based on the received raw data from field devices.

The Border Router will manage the reception of data and the interaction with the field to implement controls and will communicate data to the Edge Node Resource manager through MQTT messages and JSON payloads.

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

4.2.1 IoT Devices and Field Systems

4.2.1.1 BMS

A building management system (BMS) is a control system that can be used to monitor and manage the mechanical, electrical and electromechanical services in a facility. Such services may include electricity, heating, ventilation, air conditioning, physical access control, pumping stations, elevators, and lights. A straightforward BMS consists of software, a server with a database and smart sensors connected to an Internet-compatible network. The smart sensors around the building collect the data and send it to the BMS, which is stored in a database. If a sensor reports data that is outside of predefined conditions, the BMS will trigger an alarm. In a data center, for example, the BMS might trigger an alarm when the temperature in a rack of servers exceeds acceptable limits.

Depending on the system, the BMS software can be installed as a standalone application or can be integrated with other monitoring programs. The most advanced BMSs can monitor and manage a wide range of building services across multiple platforms and protocols, providing facility administrators with a single shared view of facility operations.

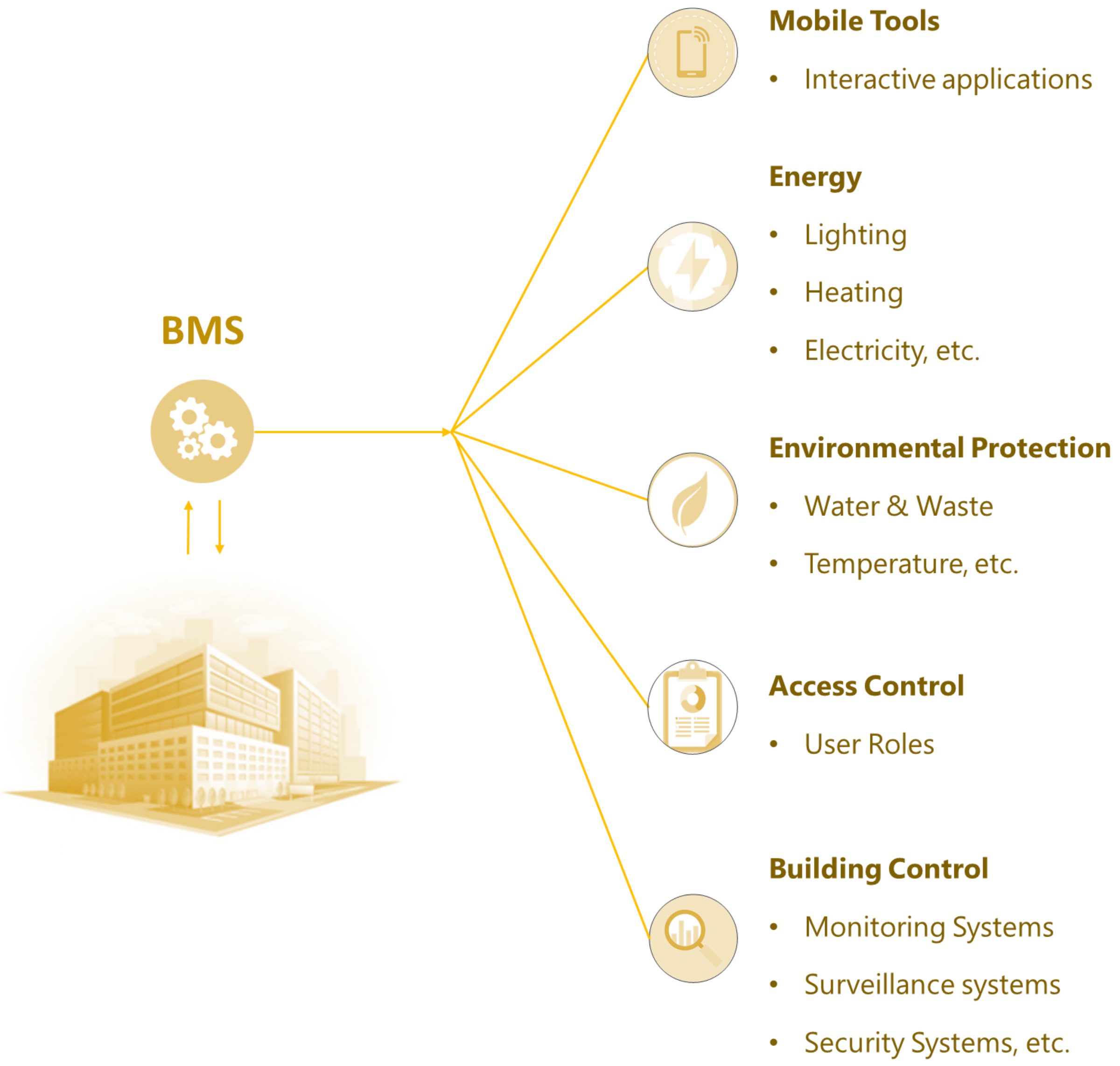


Figure 6 Example of Building Management System (BMS)

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Particularly, in COLLECTiEF the BMS is available in the Norwegian pilot for which a set of APIs to access all the analog data produced is available. In Figure 7 is shown an example of the data read through the API, in JSON format while in Figure 8 is reported the API catalog of the Norwegian BMS:


```
[
  {
    "ID": 1,
    "ZoneLetter": "T",
    "Description1": "Uteføler",
    "Description2": "320.01.RT00",
    "MaxNumberOfZones": 1500,
    "MaxValue": 30,
    "MinValue": -20,
    "MaxAllowed": 9999,
    "MinAllowed": -10000,
    "DaySetPoint": 21,
    "DeltaTemperature": 0,
    "LowSetPoint": 17,
    "ClosedSetPoint": 11,
    "PMin": 20,
    "Type": 2,
    "Acceleration": 0,
    "AccelerationStr": "",
    "Unit": "°C",
    "InndorRef": 0,
    "OverrideStatus": "",
    "BackColorString": "FFFFFF",
    "ForeColorString": "000000",
    "ActualValue": 11.2,
    "SetValue": 0,
    "Gain": 0,
    "OutdoorTemperature": 11.2062448500801,
    "RefZoneLetter": "",
    "RefZoneID": 0,
    "HasEffectCalculation": 0,
    "HasEffectRegulation": false,
    "IsEffectRegulationOn": false
  }]
```


Figure 7 Example of API contents in JSON format

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

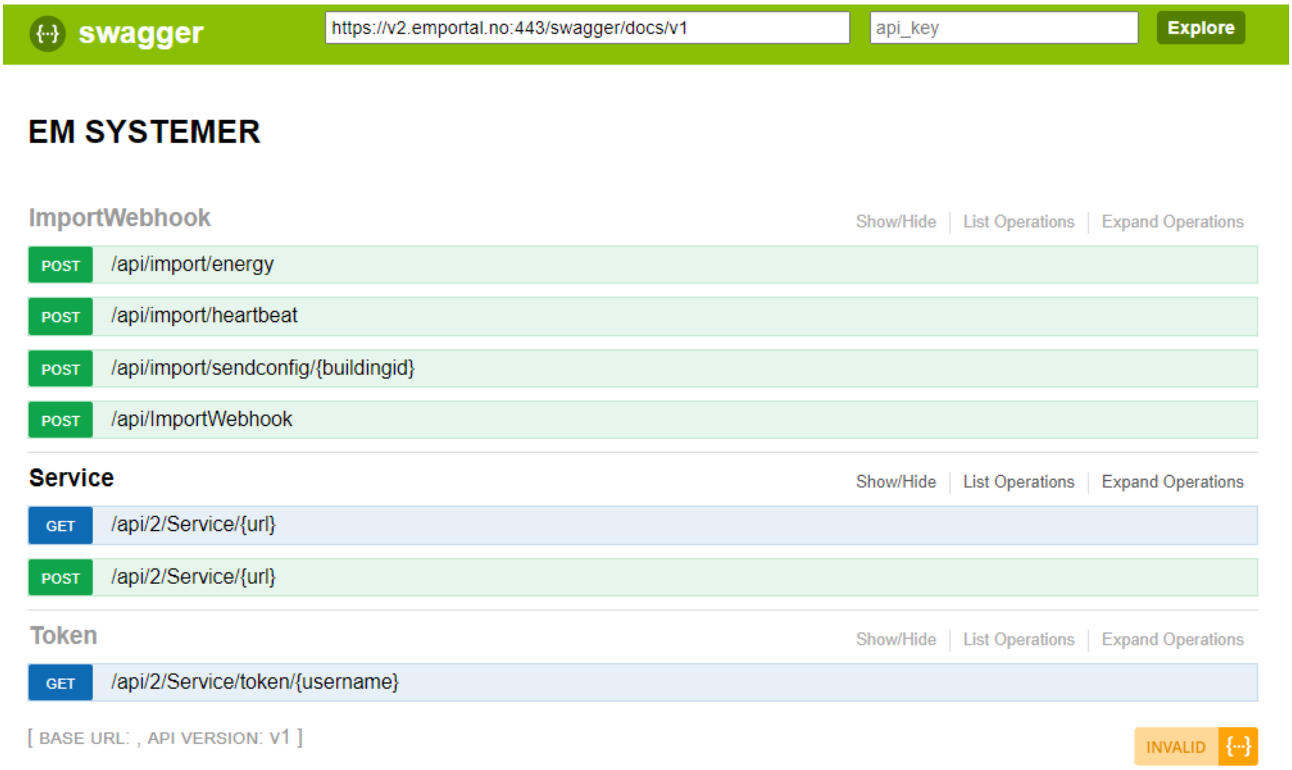


Figure 8 Norwegian pilot API catalog

4.2.1.2 *Power Meter*

In the other pilots, low voltage power meters will be installed to monitor in near real-time the energy consumption of the buildings. The power meter is a device able to monitor the energy and power consumption and quantities such as the phase current, the linked voltage and active, reactive and apparent power of the monitored environments.
The meter can be used for a single phase or three phases 3- and 4-wire network and, usually, it is interfaced with wired protocols for the collection of the data (e.g., RS485 Modbus RTU/TCP [3] or RS485 BACNET [4] communication protocol).

4.2.1.3 *Smart Plug*

Smart Plugs will be installed for monitoring and controlling individual loads. In particular, Shelly Plug was chosen for installations in pilot projects. Shelly Plug is able to send data to the COLLECTiEF BRIG device using a Wi-Fi connection and is also able to act as an Access Point. Shelly Plug has integrated a precise power meter to monitor the overall consumption of the electrical devices connected to it and it is possible to turn it on/off via the /settings endpoint. In particular, it is possible to control the Shelly Plugs using the following resources: (1) /settings/relay/0 to configure the behaviour of the plug in case of application of specific device control policies (e.g., switching on or off depending on consumption thresholds or time slots) and (2) /relay/0 to control and monitor the plug.

4.2.1.4 *Smart Valve*

Smart thermostatic valves will be installed to monitor and control the temperature of the radiators. In particular, Shelly TRV was chosen for installations in pilot projects. Shelly TRV is able to send data to the COLLECTiEF BRIG device using a Wi-Fi connection. Shelly TRV has a precise integrated temperature sensor to monitor the radiator temperature, giving the possibility to control the temperature by setting the ambient temperature in the range from 5 ° C to 30 ° C. It is possible to

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

close and open the valve via the /settings/actions topic by setting the following two types of actions: (1) valve_close to invoke when valve is closed and (2) valve_open to invoke when valve is opened. The MQTT messaging protocol is used for communication between the Shelly TRV smart valve and the BRIG device. When configured for MQTT, Shelly TRV sends values from its internal sensors on a specific MQTT topic by publishing a JSON payload with the contents of the HTTP / status endpoint. Shelly TRV supports a series of commands posted on a specific topic for setting actuators, for example to set the target temperature in a range from 5 ° C to 30 ° C.

4.2.1.5 Sphensors

Sphensor™ is a sensor fusion units developed by LSI LASTEM that allows to simultaneously measure thermo-hygrometric parameters such as temperature and relative humidity of the air, environmental parameters such as atmospheric pressure, illuminance for five different orientations, UV-A radiation and quality of the air in terms of the concentration of VOCs, CO_2 and PM1, 2.5, 4, 10. The Sphensor™ data logger sensors, using the Thread protocol, are able to send the information acquired to the Sphensor Gateway through a robust mesh radio network. The Sphensor™ Gateway, in addition to acting as a buffer memory, has been integrated with the BRIG module.

Three units with different monitoring capabilities will be used in the COLLECTiEF project. They are identified with the codes:

- **PRMPB0401**: measuring air temperature, relative humidity and atmospheric pressure;
- **PRMPB0402**: measuring air temperature, relative humidity, atmospheric pressure and illuminance in five orientations;
- **PRMPA0423**: measuring the concentration of CO_2, VOC, PM1, 2.5, 4, 10.

Sphensor™ data are published using MQTT protocol in JSON format for the upper layer. The structure of the topic on which the data will be published is reported below:

sphensor/<border router serial>/<sensor serial>/grouped_inst

An example of the JSON payload is reported below:


```
[
{
"timestamp": "2020-01-03 06:03:27",
"sensor_type": "opt3001_4",
"value": 2.2177724838256836,
"result": "ok",
"channel_index": 0
},
…......
]
```


Figure 9 Example of Sphensor JSON payload

In Table 3 there is a list of the possible couples *sensor_type/channel_index.*

Table 3 List of possible couples sensor_type/channel_index

sensor_type	channel_index	name
sht3x	0	Air temperature
sht3x	1	Relative humidity

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

ms5607	0	Cell temperature
ms5607	1	Atm. pressure
opt3001_0	0	Lux 1
opt3001_1	0	Lux 2
opt3001_2	0	Lux 3
opt3001_3	0	Lux 4
opt3001_4	0	Lux 5
adc_uva	1	UVA

4.2.2 Other Devices and Services

Interfacing with weather services is envisaged in order to retrieve data relating to historical weather conditions and forecast of weather conditions through specific APIs.

Furthermore, it is foreseen the exploitation of all those legacy systems already installed from which it could be possible to gather useful data for the higher levels of the architecture.

4.3 Distributed Layer

4.3.1 BRIG Device (Border Router + Edge Node Resource Management (iGateway))

This component constitutes the real bottom layer of the COLLECTiEF architecture system and refers to the communication interfaces with the field devices and the Edge-to-Cluster communication. Particularly, the border router inside the BRIG device is the basic interface with the physical world and together with the Edge Node Resource Management (iGateway) performs primary information processing based on the received raw data from the field devices. In addition, the BRIG device provides context interpretation of physical signals according to the identified standards for the representation of the data.

Particularly, the Edge Node Resource Management enables and ensures the Edge-to-Cluster and Edge-to-Field communication in an interoperable and intelligent way facilitating seamless, end-to-end communication via the development of an open, versatile and secure Intelligent Gateway.

Hence, The BRIG device (Border Router+iGateway) will be deployed on a single HW board, by implementing full communication with the field (legacy system, RES, distributed sensor network, BMS, ...) and guaranteeing secure data exchanging for data storage and implementation of the local control strategies and local dashboard in order to manage problems with the network and manage locally some aspects (flexibility management, optimization services).

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

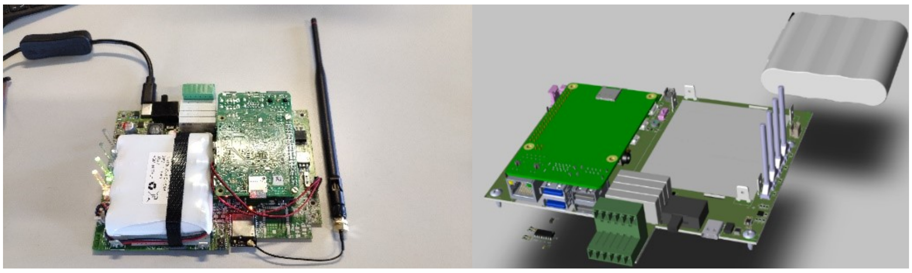


Figure 10 BRIG Device Layout and first prototype

4.3.2 CI edge Algorithms for demand side management and building thermal network optimization

Edge-level algorithms act on the basis of a signal generated at each time step from the grid/ generation-side generated by a specific function called signal generator.

This function is independent from the edge conditions and has access to the aggregated current and predicted energy use of all edge nodes within the cluster.

Signal generator requires the following steps in order to calculate the signal:

- The actual energy load (L), which is the current requirements;
- The typical energy load (MinL), which is an aggregated estimate at typical condition (anomalies are pruned);
- Typical energy load at extreme conditions (MaxL), usually with a 3 hour forecast in advance, meaning that it is a raw estimate of the maximum demand possible under anomalous conditions.

The signal output is an integer between zero and 5. Particularly, the signal is a normalized energy load value with respect to the load under extreme conditions (maximum value for the normalization) and the load under typical conditions (minimum value for the normalization) thus the signal is calculated as follows:

$$signal = (L - MinL)/(MaxL - MinL)\ for\ MinL < L < MaxL$$

A specific function is in charge of gathering and storing the available controls on the demand side management, as well as the respective features depending on the type of user. Particularly, a set template is utilized but, the policies of control can be imported also by the user. Particularly:

- Use types depend on the building template
- The use types determine an initial set of controls and their respective values within a given edge
- The thresholds and the respective steps can be manual inputs
- Other things, like change rate over time as well as any other pattern and difference can be a further input (signal engagement determines a threshold for the signal to be actually controlled)

After all these controls are set the output is a dictionary/JSON for the values, with the given in Figure 11:

```
controls = {
    "setp_temp":{
       "values":np.arange(22,
```


This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

```
      28),
      "# Setpoint temperature"[
         "C"
      ]"features":{
         "step":1,
         "changeRate":2,
         "engagementSignal":1
      }
   },
   "vent_rate":{
      "values":np.arange(1,
      12,
      3),
      "# Ventilation rate per area"[
         l/m2/s/10
      ]"features":{
         "step":3,
         "changeRate":3,
         "engagementSignal":3
      }
   },
   "plug_load":{
      "values":[
         0,
         1
      ],
      "# Plug loads"[
         "on/off"
      ]"features":{
         "step":-1,
         "changeRate":.5,
         "engagementSignal":4
      }
   }
}
```

Figure 11 Control patterns dictionary example

A specific function, named "control sets generator" will generate a possible set of controls to be chosen considering the current control setpoint, reading the current signal and setpoint values, and generating new ones based on the rate of change and the steps. After that, this function generates all the possible control combinations and passes a part of them as possible control setpoints to improve the comfort or signal behaviour.

The reward is calculated at the current step and is based on the setting of the previous one, using an arbitrary equation whose result is obtained from the energy parameter as well as the comfort indicator.

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

For each case, energy and comfort have different weight on the reward calculation, following Table 4 for each case

Table 4 Weights of energy and comfort in the calculation of the reward

COEFFICIENT VALUE	C+	C-
E+	1	0.5
E-	0	0

With E+ or C+ being a Boolean that is true if the level of comfort is higher at the current time than in the successor step, and minus being the case of a negative difference of energy.

$$RD(E) = -\Delta E/E(t)*100$$

$$\text{w/ } \Delta E = E(t) - E(t+1)$$

Lack of energy is always bad while lack of comfort is more tolerated in the coefficient.

The final calculation of the reward mechanism, applied at time t+1, is ***Reward(t) = RD(t)*COEFFICIENT.***

Finally, the history repository generator generates a data structure to record the controls and their reward dividing to hour, signal, and month in a structure similar to Table 5:

Table 5 Data structure of controls with respective reward

Month structure	signal	Hour (24 columns)
12 unique values, one per moth	0 to 5	Value for each hour

All information described up to now is used to perform a series of operations at each timestep of the timeseries data as reported below:

1. **Setup Section**
   1.1 Read the control features repository, containing the data in JSON format
   1.2 Read the control features from the templates
   1.3 Read randomness: define a random probability to choose in order to make the behaviour mixed and not fall into bad equilibriums
   1.4 Read *temperatureDiscomfortThreshold* (abbreviated as *tempDiscomfortThresh* from now on)
   1.5 Get the data structure for the history repository generator
2. **Loop section (at each iteration)**
   2.1 At first it gets the reward output from the reward generator, based on the chosen control at the previous time step: the way the reward is generated will now be delineated at its own section
   2.2 The reward is passed as output, together with the previous control value, whose calculation function will be explained in its own section

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

2.3 The control setpoint is obtained from the previous iteration of the script, which means that at each iteration the control setpoint value must be saved for the next iteration just like it happened for the reward mechanism
2.4 Next I save both reward and control in a history repository, together with hour, measured signal and date of acquisition.
2.5 After that the room temperature is read and the signal_generator function is called, which will be explained in its own step, getting an integer variable called signal between 0 and 5, which is then used to determine the direction of the tree

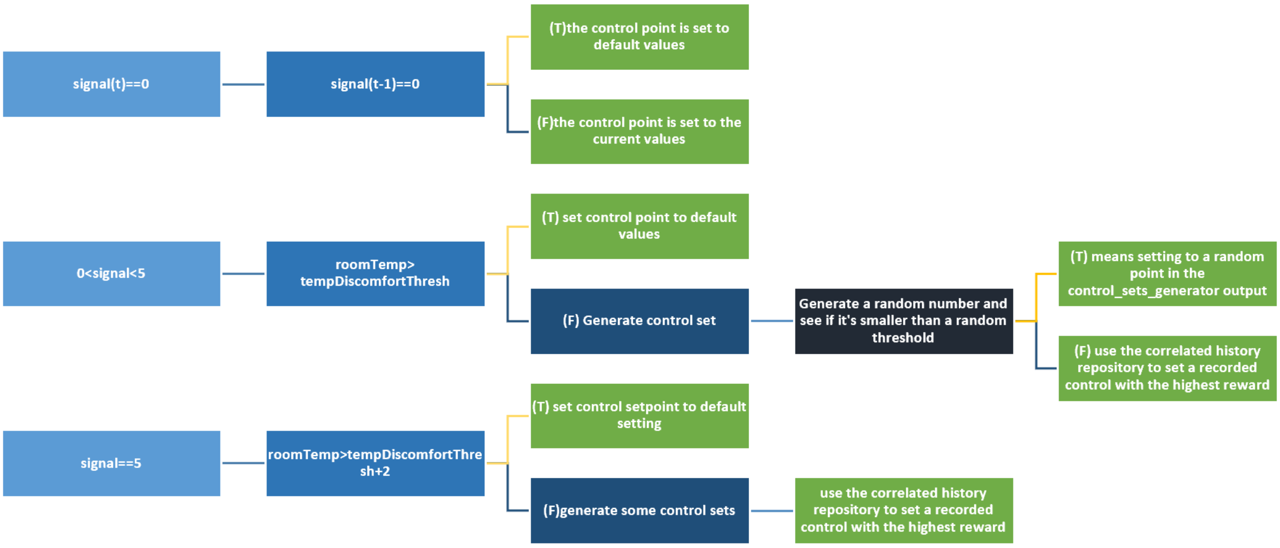


Figure 12 Decision tree performed after the signal is received

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

4.4 Cluster Layer

4.4.1 Cluster Node Resource processing and management

This is the block that takes care of receiving and managing the data coming from the different edge nodes and sending to them the data useful for the algorithms that run locally (e.g., price signal) as conceptually represented in Figure 13 in which the control and communication will be empowered by Collective Privacy and Collective Fleet Management.

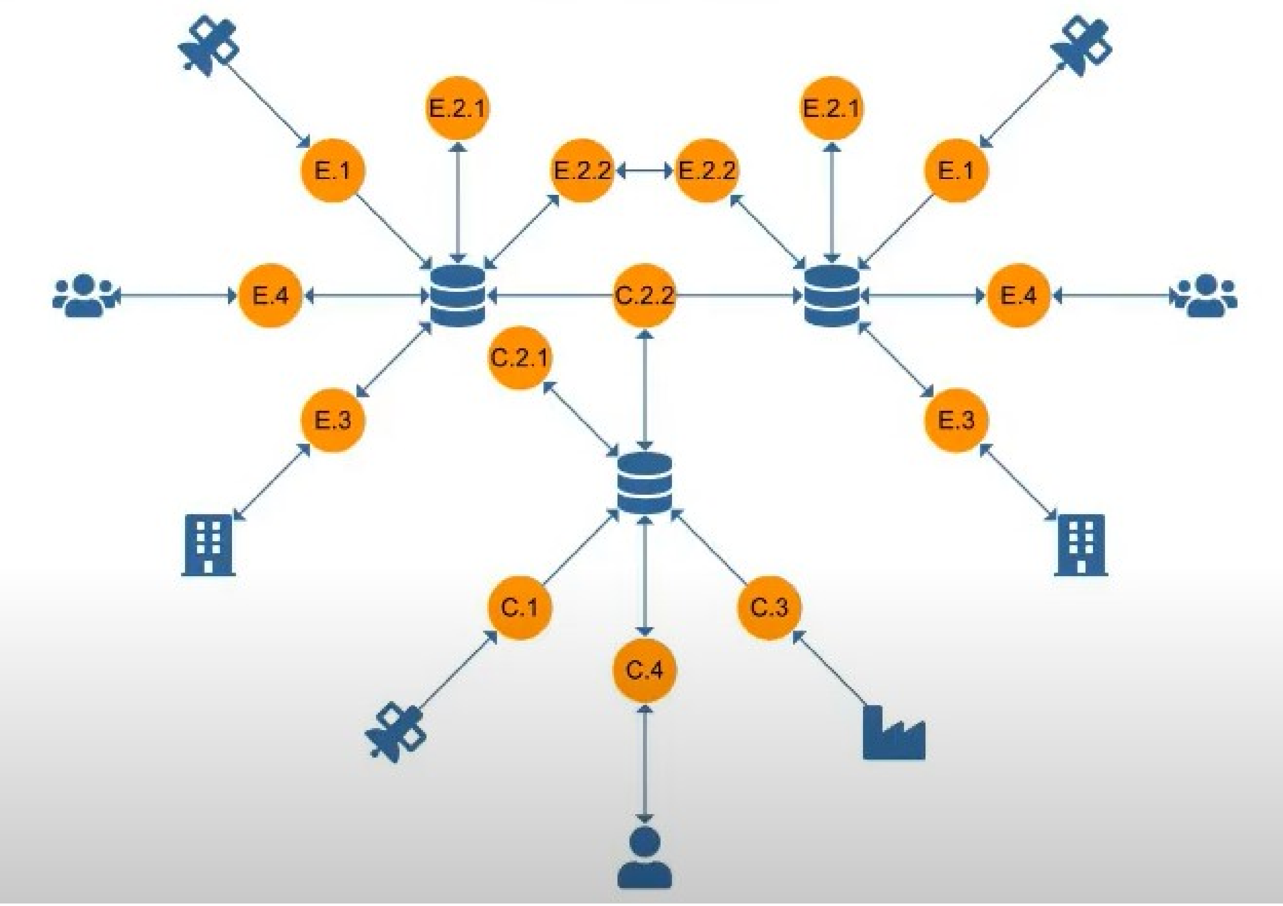


Figure 13 Conceptual diagram of the functioning of a system of one Cluster Node and two Edge Nodes

There are at least one data base per component, i.e., one for the cluster node and one for each edge node. The cluster node is only involved with data for aggregated demand, and communicate with the edge nodes through broadcasting of a synthetic price signal and possibly additional messages, thus upholding privacy. The edge nodes translates the broadcasted signals into local control actions, e.g., by optimising energy demand with respect to a synthetic price signal while upholding comfort in the form of a number of inequalities on the expected consequences of the control actions. The edge nodes then coordinate their actions by means of edge node to edge node communication in accordance with the principles of Collective Intelligence.

The cluster node and edge nodes may use an off the shelf solution such as the NODA self-host [5] to manage orchestrate internal components, or some other version of the same kind of technology. Communication with other parts of the system will be achieved through a combination of MQTT and REST, with the goal of keeping the number of technologies low in order to facilitate speed of development and ease of maintenance.

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Contrary to the final product, to facilitate development and evaluation, the cluster node and edge nodes will utilise a REST API to download weather data and weather forecasts from the central database, thus guaranteeing that the same data will be available for evaluation purposes. In addition, the central database will also curate and provide data on energy prices as well as the aggregated demand for the areas being controlled.

4.4.2 CI Cluster Block

The components of the Cluster Block will run the high-level algorithms for computing the signals and other messages to be communicated to the Edge Block. The individual algorithms will be implemented by the group responsible for their respective design, thus guaranteeing accuracy and speed of development.

4.5 Application Layer

The upper layer of the COLLECTiEF system represents the integrated display structure that allows for better user interaction. This level receives data from the lowest layers, which is further analyzed and interpreted. The main objective of this analysis is to allow all interested parties to share a complete view of the functionality of the platform provided, gaining a more active role during the operation of the platform. As a result, bi-directional data flow is done between the platform and the front-end components, as different stakeholders make decisions based on the platform results provided. These decisions must be returned to the platform as input values that lead to an iterative process with the main purpose of concluding the optimal solution for the COLLECTiEF system.

The Application Layer includes two main Graphic User Interfaces:

- Human Building Local Interface
- Fully Integrated Dashboard

4.5.1 Human Building Local Interface

Human Building Local Interface is used as the interface between the edge node and the end user. In the context of COLLECTiEF architecture the local interface is expected to display near real-time operational information tailored to user ability to coordinate and control the interaction with the local devices. From the point of users, they can supervise the current state of their devices and enter information according to their energy plans making sure that the decisions taken by the end users directly affect the operation of the node and the network. The main functionalities of this HMI are:

- Real-time local environmental, energy and thermal measurement monitoring.
- Detailed user-friendly analysis based on real-time energy consumption/production and flexibility values.
- Visualization of the device states and of the settings implemented by the algorithms of the COLLECTiEF edge node.
- Configuration of the desired strategies.

4.5.2 Fully Integrated Dashboard

This component will deliver all the necessary information that represents fully the operation of the Cluster Layer of the COLLECTiEF platform via a series of statistical diagrams, real-time flow diagrams, and comparative visualization with historical data etc. to provide a visualization of different aspects related to COLLECTiEF. Stakeholders can use the information delivered by dashboards to plan their actions regarding the DR strategies planning and the services that can be offered by evaluating the results achieved by the strategies followed in recent times.

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

5 Dynamic View

The dynamic view analysis of the system provides detailed information and defines how the system actually works within the runtime environment and how it performs in response to the external (or internal) signal. The interactions between system actors and system components are usually data streams representing information exchanged in the parallel or sequential execution of internal tasks.

The UCs is presented by using a template defined in annex 1, a table based in specific standard [6] in which all necessary information for a specific process is described: from high-level information, such as the name of the UC, to a detailed step-by-step analysis of the realization of the UC as well as the actors involved.

The logic of these complex operations is presented through Sequence Diagrams [7], [8], [9] defining the functionalities of each of the key architectural components and the execution flows within each use case. In this chapter, the first version of the project use cases is presented by introducing also the sequence diagrams.

5.1 Use cases and sequence diagrams

In this first phase of T3.1 activities, four main UCs have been defined through an internal consultation of the partners responsible for the pilots with the internal technology providers. In this section, detailed descriptions and sequence diagrams are reported for each of them.

5.1.1 UC01: Demand Side Management: Room temperature control and energy flexibility

Table 6 UC01: Demand Side Management: Room temperature control and energy flexibility

UC Description	
UC Name	UC01: Demand Side Management: Room temperature control and energy flexibility
Version	V0.1
Authors	E@W, T3.1 partners
Last Update	1st in 02/08/2022
Brief Description	The goal is controlling the room temperature and loads depending on energy demand and comfort as well as network constraints
Assumptions and Pre-Conditions	• A control interface is available for the Resident in the room. • The system has sensors and actuators to control the building temperatures. • The buildings are connected • The central repository data are stored with Collective Privacy in mind

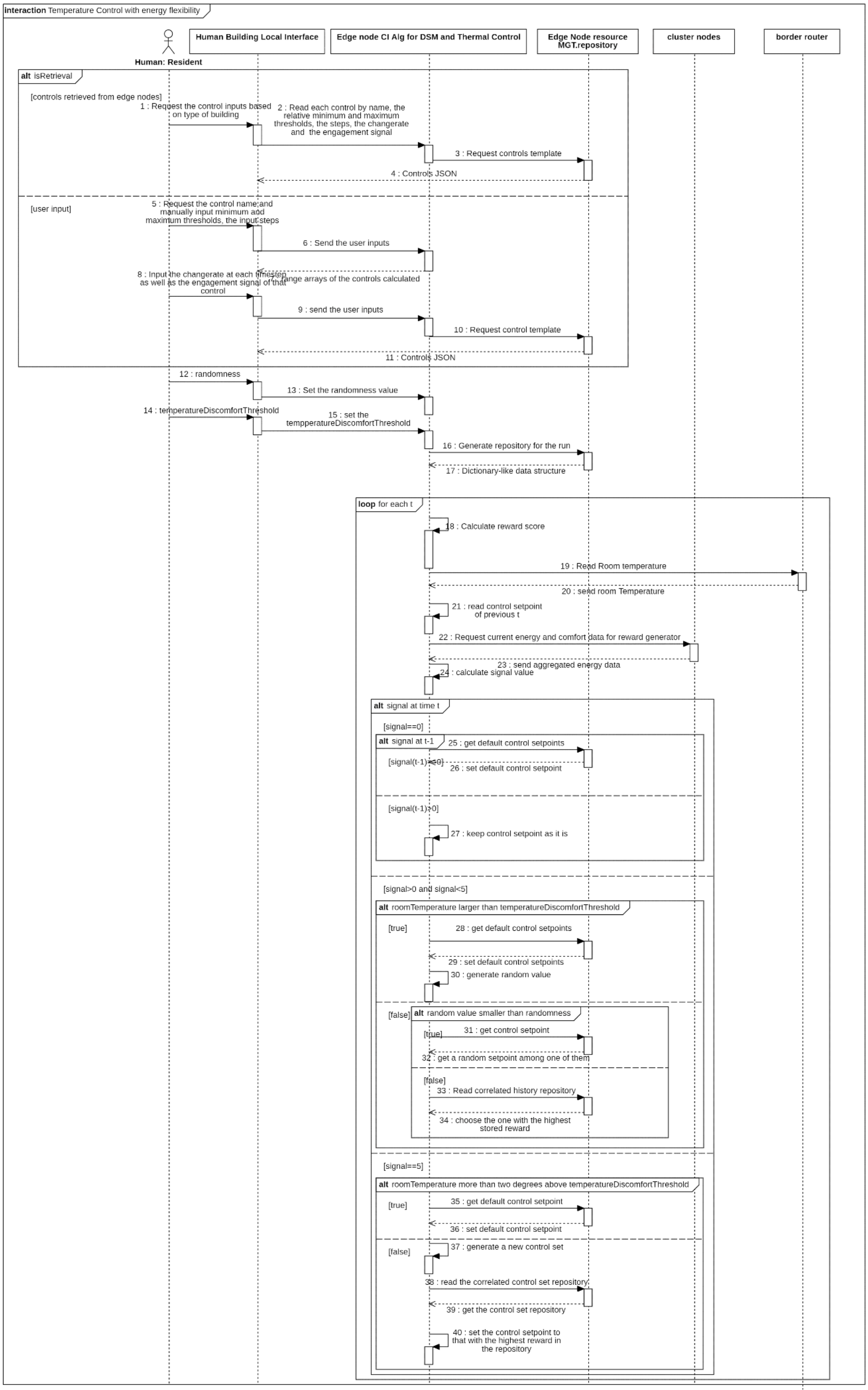
This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Goal (Successful End Condition)	Find an optimal configuration for the control set point value
Post-Conditions	System functioning according to the set point set iteratively
Involved Actors	Inhabitants, owners of the building
UC Initiation	The residents need to improve housing comfort while saving energy.
Main Flow	1. Collect all the available controls on the demand side management and their features from a template based on the use type; characteristics like value thresholds and step are always a user input, while the change-rate and the engagement signal are optional 2. The control features are loaded from the templates 3. Read the randomness parameter in order to not stick to suboptimal equilibrium 4. Read the temperature discomfort threshold 5. Generate the data structure relative to the history repository 6. Get the reward output from the reward function, whose coefficient is based on the difference between the previous energy value and the current as well as the difference between the previous comfort score index and the current, while the energy value is the relative difference between the current energy demand and the one at the previous iteration 7. Get the control setpoint of the previous iteration 8. Save control setpoint and reward in the history repository, containing date of acquisition 9. Get the signal value, which is an integer between 0 and 5, based on the normalized energy demand value with the typical demand as the minimum and the forecasted extreme demand as the maximum

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

	10. When the signal is between 1 and 4 generate the control set from the control features and the possible control combinations 11. Generate a random number between 0 and 100 and compare to the value the randomness parameter 12. When the random number is smaller than randomness, one of the control setpoints is randomly chosen
Alternative Courses	10a. When the signal is exactly 0 then the signal at the previous timestamp is checked 11a. In case of signal at the previous timestamp equal to 0, then the control points are set back to default values
	11ab. In case the signal at the previous timestamp is larger than zero then the control point is set to the current values
	11b. In case the signal is exactly 5 and the room temperature is more than 2 degrees above the temperature discomfort threshold the control setpoint is set back to the default setting
	10bb. In case the signal is exactly 5 and the room temperature is more than 2 degrees some control sets for the setpoint are generated 11bb. From the correlated history repository, the recorded control with the highest reward is selected as the new control setpoint
	11c. In case the room temperature is above the temperature discomfort threshold then use the defaults for the control setpoints instead of randomly generating new ones
Relationships with other UCs	UC02, UC03
Architectural Elements / Services Involved	• Cluster node • Edge node • Controllable heat pumps • Controllable loads • Border router to sensor fields • Central database
UML Sequence Diagram	

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683



This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Figure 14 UC01-SD: Demand Side Management: Room temperature control and energy flexibility

5.1.2 UC02: Human Building interaction

Table 7 UC02: Human Building interaction

UC Description	
UC Name	UC02: Human Building interaction
Version	0.1
Authors	E@W, T3.1 partners
Last Update	1st at the 02/08/2022
Brief Description	The HBI will allow users to interface and interact with the edge nodes that control and monitor the temperature and loads resulting in an effective control of energy parameters for the building, as well as receive and apply suggestions to improve efficiency according to the KPIs. The user may also apply its own controls.
Assumptions and Pre-Conditions	• A control interface is available for the Resident in the room. • The control algorithms are active and in constant function • The data can be received and works under a premise of local privacy
Goal (Successful End Condition)	The data is visualized in a valuable and meaningful way to the user so that the user may make proper decisions for saving resources
Post-Conditions	The system keeps functioning whether or not the parameters have been changed.
Involved Actors	Building Occupants, Managers, Owners
UC Initiation	Users access the interface
Main Flow	1. User accesses the interface and views among the possible selections of data 2. User selects the information of interest to view, coming from the relative components and services involved in the operation 3. The local interface requests the necessary data from each component in order to present it to the user, some control suggestions are given 4. The user reads the information and closes it down when he has obtained the needed information

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Alternative Courses	3a. In case the user takes up the suggestions it can apply them directly 4a. The system applies the necessary to controls to the components 5a. Go back to step 4
	3b. User applies his own controls that go back to the building control and management algorithms 4b. The system receives the controls 5b. Go back to step 4
Relationships with other UCs	UC01
Architectural Elements / Services Involved	• Thermal network optimization • Building thermal optimization and flexibility management • CI-based building control and management algorithms • NODA Network Optimization • Collective Intelligence based control system
UML Sequence Diagram	

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

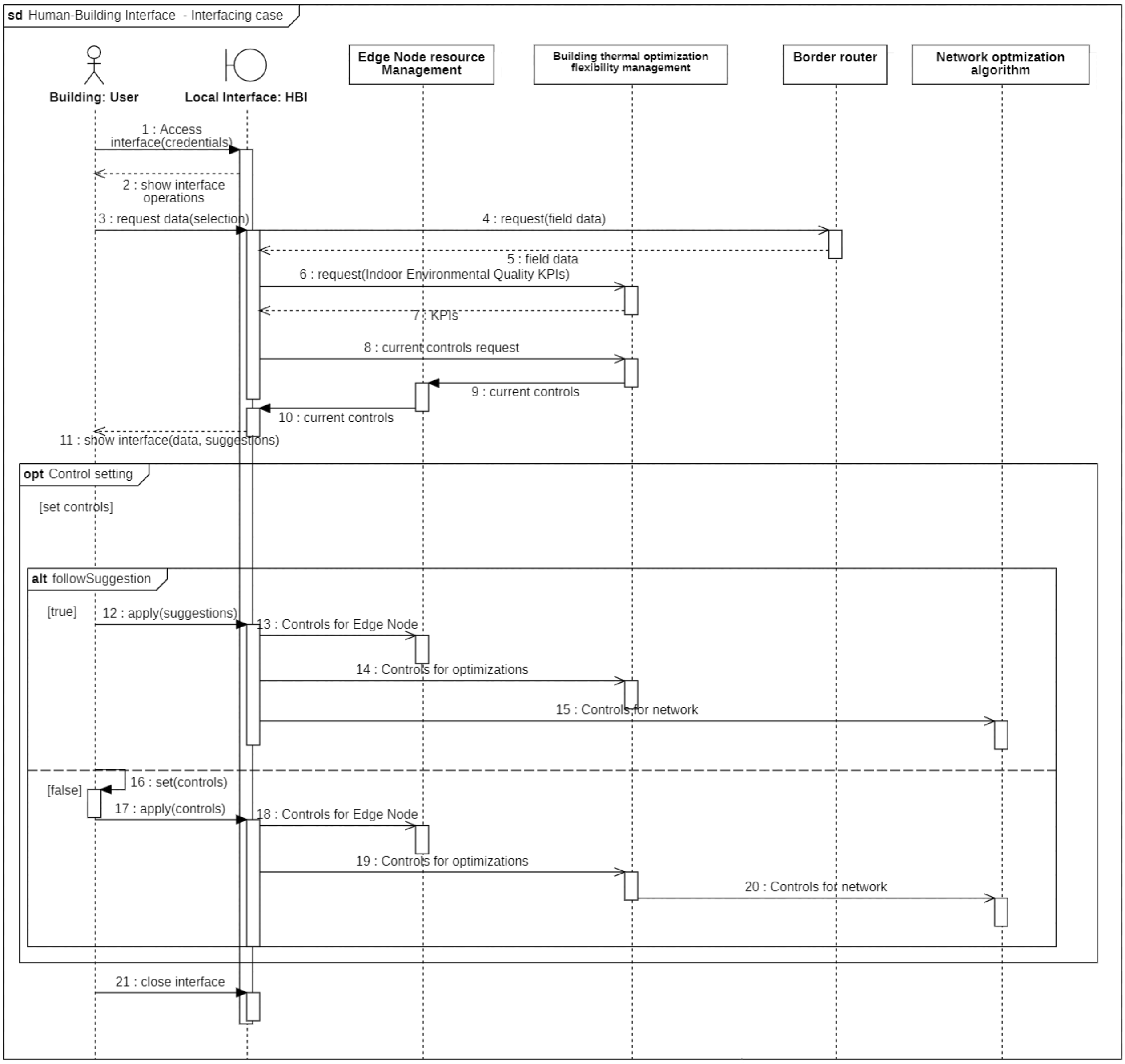


Figure 15 UC02-SD: Human Building interaction

5.1.3 UC03: Network and local thermal optimization based on cluster node and edge nodes communication

Table 8 UC03: Network and local thermal optimization based on cluster node and edge nodes communication

UC Description	
UC Name	UC03: Network and local thermal optimization based on cluster node and edge nodes communication
Version	0.1
Authors	E@W, T3.1 partners

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Last Update	1st at the 02/08/2022
Brief Description	Optimise network-wise heat demand by constructing a price signal (modulo a positive scalar) to the end of advising the edge nodes of when to consume more and when to consume less, while the edge nodes calculate a temperature setpoint according to external and internal constraints, and finally send back a price forecast that the cluster nodes use to calculate the next price signal.
Assumptions and Pre-Conditions	• Sensors and temperature actuators must be connected to the respective edge node • The edge nodes must be connected to the cluster node through communication system (e.g., NODA self-host) • The data is obtainable from a common database with Collective Privacy as well as encrypted message exchange
Goal (Successful End Condition)	Through the price signal, the building groups and respective heating/cooling systems are able to save up on energy costs while providing comfort
Post-Conditions	The set points of the edge node actuators have been changed according to constraints
Involved Actors	Property owners, managers and energy providers (indirectly)
UC Initiation	Once the Node is set up, the data is continuously retrieved and processed on local data instances as well as a main database.
Main Flow	1. The database stores and makes available the latest weather to both cluster and edge nodes and price signal forecast to the cluster node 2. At a higher frequency, the database also sends additional data to the cluster node for the price signal calculation 3. From there the price signal forecast of the cluster is sent to the edge node 4. The edge node sensors send to the edge node algorithms all the necessary data to calculate the temperature set point and energy demand 5. The temperature set points are sent to the respective edge node actuators 6. The demand forecast is also calculated by the edge node algorithms and sent to the main database to construct the modulated demand forecast, which is used for the next iteration 7. The user is informed of the actuations carried out

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Alternative Courses	If specified, the only demand forecast used is the one coming from the local cluster and not the general database
Relationships with other UCs	UC01, UC02, UC04
Architectural Elements / Services Involved	• Communication with the cluster node, in order to manage the edge node • Cluster node connection to the central database • Weather APIs to retrieve the data necessary for cooling and heating • Each edge node and their own local database • Central Database

UML Sequence Diagram

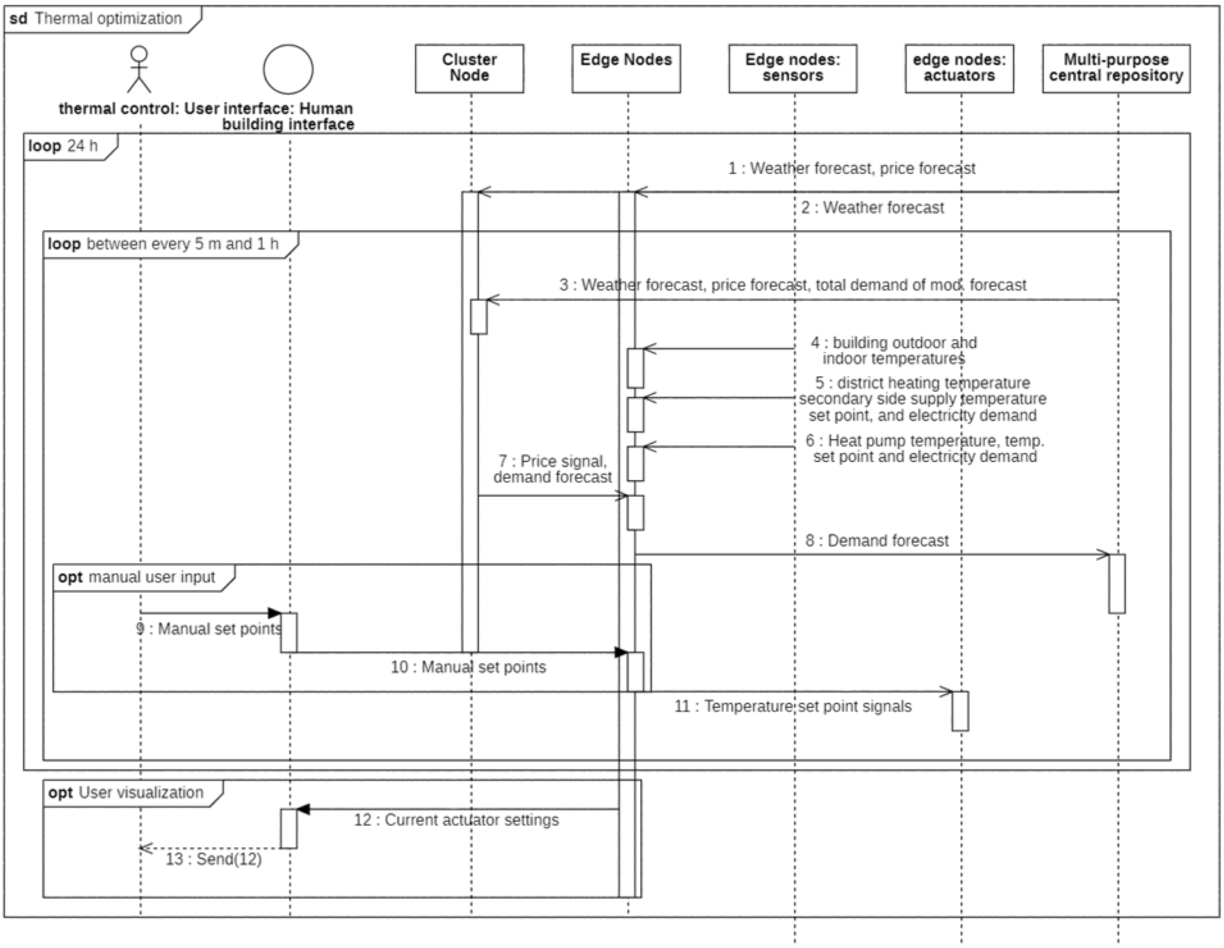


Figure 16 UC03-SD: Network and local thermal optimization based on cluster node and edge nodes communication

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

5.1.4 UC04: User high-level interaction through the Fully Integrated Dashboard

Table 9: UC04: User high-level interaction through the Fully Integrated Dashboard (Cluster Node)

UC Description	
UC Name	UC04: User high-level interaction through the Fully Integrated Dashboard (Cluster Node)
Version	0.1
Authors	E@W, T3.1 partners
Last Update	1st at the 02/08/2022
Brief Description	Cloud-based dashboard to visualize information at the level of building clusters to property owners, managers and energy providers
Assumptions and Pre-Conditions	The cluster node is set up and the relative actors have an access to the dashboard, which in turn can access the information from the cluster nodes and the edge nodes
Goal (Successful End Condition)	Provision of the necessary high-level insights for each actor
Post-Conditions	Visualization of the energy consumption, heating and performance of the applied algorithms
Involved Actors	Property owners, managers and energy providers
UC Initiation	Dashboard is set up for access by the relative actors
Main Flow	1. The actor accesses the dashboard through a web application 2. The actor desires to view the data, in a general or specific manner according to the necessity 3. The dashboard requests the required data from each component 4. The received data is presented to the actor as requested
Alternative Courses	None defined
Relationships with other UCs	UC03
Architectural Elements / Services Involved	Cluster Node software components such as: • Collective Fleet Management • Noda Network Optimization, thermal network optimization and aggregated flexibility management • Collective Intelligence based control system • Data and Trend analytics Integrated Dashboard
UML Sequence Diagram	

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

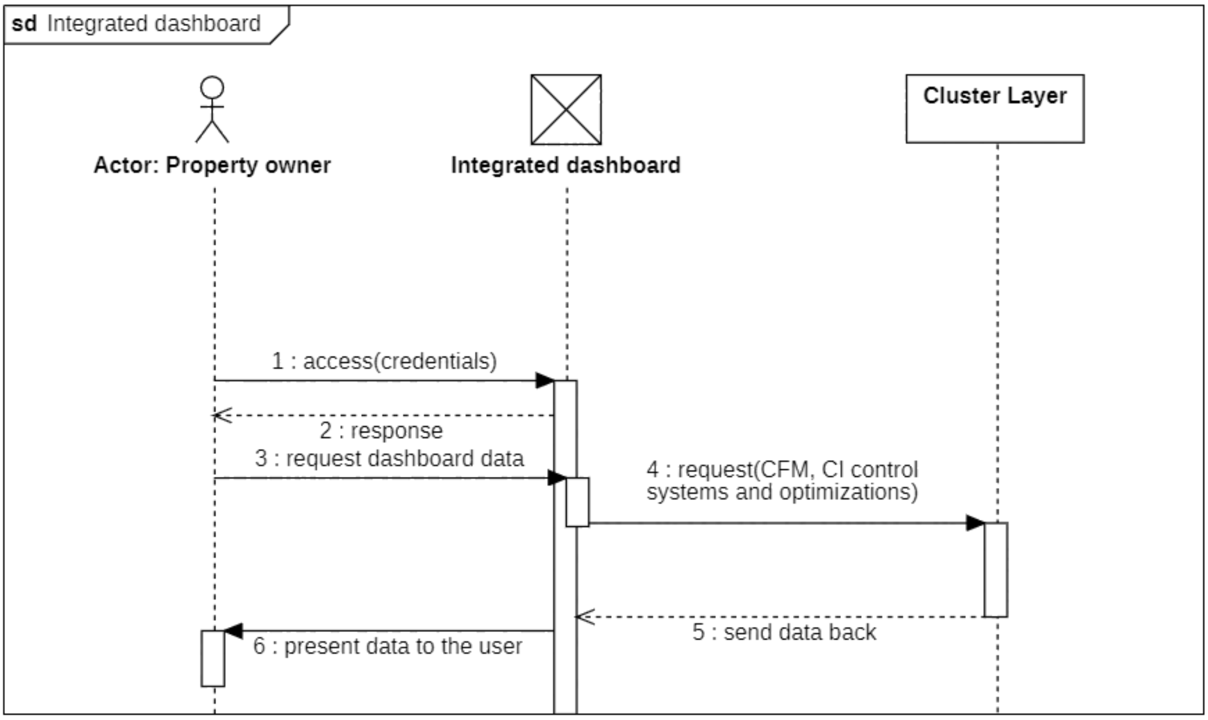


Figure 17 UC04-SD: User high-level interaction through the Fully Integrated Dashboard (Cluster Node)

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

6 Deployment View

The Deployment View presents aspects of the system that are connected with the realization of the system's components in the physical world. This view defines the physical entities of the environment, in which the system is intended to perform its running operations, including:

- Technical environment (e.g., processing nodes, network interconnections, etc.);
- Mapping of software elements to the runtime environment;
- Third-party software requirements;
- Network requirements.

The architectural view reported in Figure 18 provides a first overview of the deployment environment of the COLLECTiEF platform, which depends on the pilot sites topology and on the architecture components characteristics. This architectural view covers the currently known hardware requirements of the software modules and the tools to be used represented in Deployment Diagram.

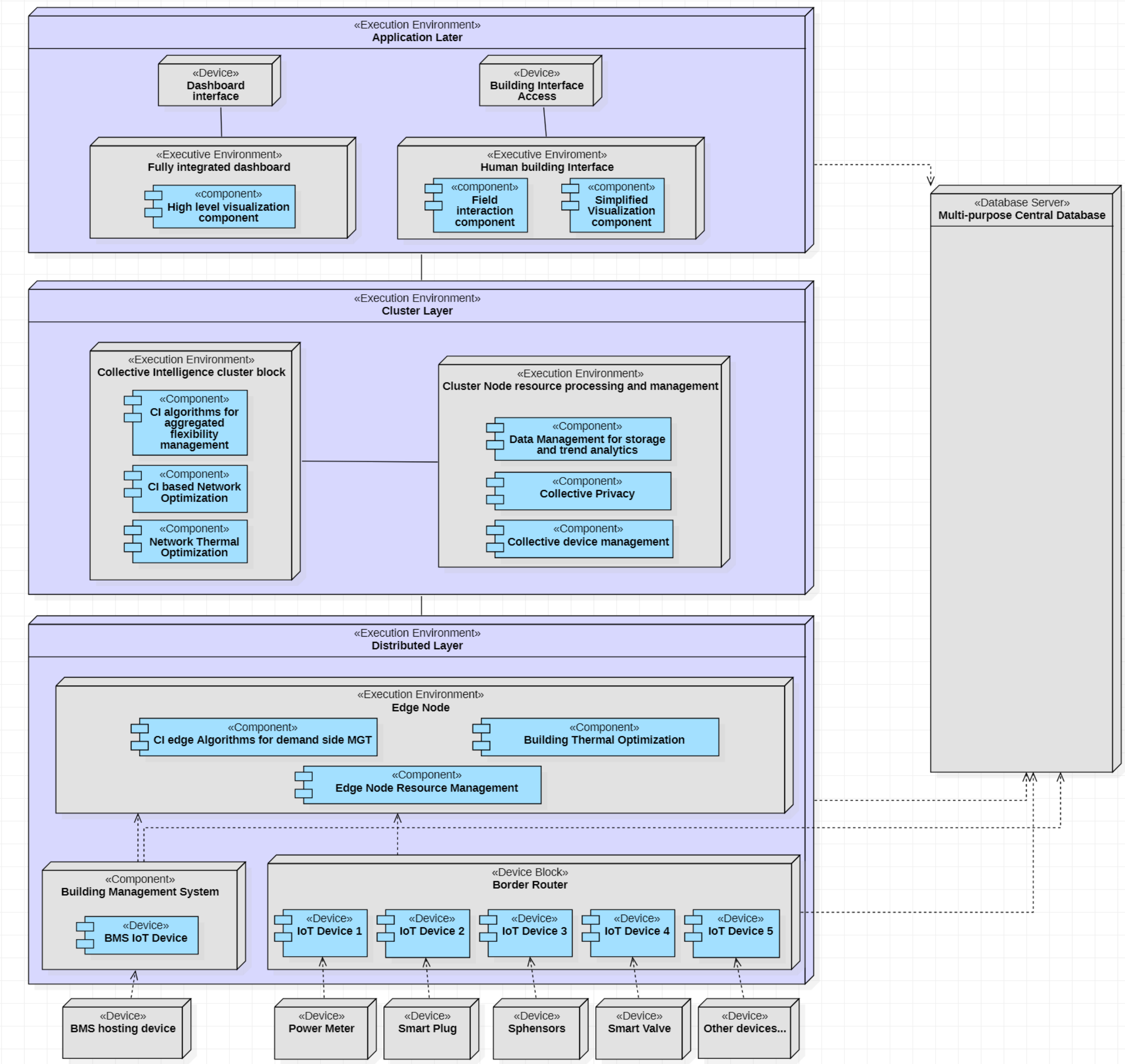


Figure 18 COLLECTiEF Deployment Diagram

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

7 Architectural Components Detailed Specifications

7.1 Field Layer devices

7.1.1 APIs for Outdoor measurements service

Table 10 APIs for Outdoor measurements service

Name of New Component/Service:	*API for Outdoor measurement services integration (Third Party system)*
Type:	*Software*
Functionality:	*Weather data collection from building-integrated weather stations and Open-API weather data.* - *Historical weather conditions* - *Forecasted weather conditions*
Input Connections & Interfaces: From which components it receives input	*Building-integrated weather stations – MySQL database* *Open-API weather data – Meteostat, or similar*
Output Connections & Interfaces: To which components it sends the results	*Edge Node*
Relevant Use Cases	*UC02, UC03, UC04*

Input Parameters

Attribute/Para-meter	***Short Description***	***Data Type***	***Data Format***	***Value Range & Frequency***	***Data Received From***
Air Temperature (AirTC_Avg)		*float*	*csv*	*Celsius, [-40,60], 6 samples per hour (every 10 minutes)*	*Building-integrated weather stations – MySQL database*
Relative humidity (RH)		*Integer*	*csv*	*Percent (%), [0,100], 6 samples per*	*Building-integrated weather stations – MySQL database*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

				hour (every 10 minutes)	
Rain Intensity		*float*	*csv*	*, [],6 samples per hour (every 10 minutes)*	*Building-integrated weather stations – MySQL database*
Raining	*Percentage of dry sensitive surface of the sensor*	*Binary*	*csv*	*No units, [on,off] (precipitation in progress / no precipitation)*	*Building-integrated weather stations – MySQL database*
Solar irradiance		*float*	*csv*	*W/m2, [0,1600], 6 samples per hour (every 10 minutes)*	*Building-integrated weather stations – MySQL database*
Wind speed (WS_ms_Avg)		*float*	*csv*	*Meters/second (m/s), [0,60], 6 samples per hour (every 10 minutes)*	*Building-integrated weather stations – MySQL database*
Wind direction (WindDir)		*float*	*csv*	*Polar degrees, [0,359], 6 samples per hour (every 10 minutes)*	*Building-integrated weather stations – MySQL database*
Atmospheric pressure (BP_mbar_Avg)		*float*	*csv*	*mbar, [],6 samples per hour (every 10 minutes)*	*Building-integrated weather stations – MySQL database*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Time	*Time of observation*	*string*	*JSON*	*(YYYY-MM-DD hh:mm:ss), 6 samples per hour (every 10 minutes)*	*Meteostat (https://dev.meteostat.net/formats.html#time-format)*
temp	*air temperature*	*float*	*JSON*	*Celsius, 6 samples per hour (every 10 minutes)*	*Meteostat*
dwpt	*dew point*	*float*	*JSON*	*Celsius, 6 samples per hour (every 10 minutes)*	*Meteostat*
rhum	*relative humidity*	*Integer*	*JSON*	*Percent (%), [0,100], 6 samples per hour (every 10 minutes)*	*Meteostat*
prcp	*one hour precipitation total*	*Float*	*JSON*	*Milimeters, (mm), [],6 samples per hour (every 10 minutes)*	*Meteostat*
snow	*snow depth*	*integer*	*JSON*	*Milimeters, (mm), [],6 samples per hour (every 10 minutes)*	*Meteostat*
wdir	*wind direction*	*Integer*	*JSON*	*Degrees (o), [],6 samples per hour (every 10 minutes)*	*Meteostat*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

wspd	*average wind speed*	*float*	*JSON*	*Kilometers/hour (km/h), [],6 samples per hour (every 10 minutes)*	*Meteostat*
wpgt	*peak wind gust*	*Float*	*JSON*	*Kilometers/hour (km/h), [],6 samples per hour (every 10 minutes)*	*Meteostat*
pres	*sea-level air pressure*	*Float*	*JSON*	*hPa, [],6 samples per hour (every 10 minutes)*	*Meteostat*
tsun	*one hour sunshine total*	*Integer*	*JSON*	*Minutes (m),*	*Meteostat*
coco	*weather condition code*	*integer*	*JSON*	*No units*	*Meteostat*
Attribute/Para-meter	***Short Description***	***Data Type***	***Data Format***	***Value Range & Frequency***	***Data Sent To***
All	*All*	*All*	*Datafram a*	*ALL*	*Edge Node*

Software Requirements/Development Language	*Python*
Hardware Requirements	*CampellMeteo (Pyranometre PippZonen CMP3 | SMP3, Precipitation detector DeltaOHM_HD2013.3, WindSonic)*
Communications	*Open-API, MySql connector*
Status of the development of the component	*already developed*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

7.1.2 Smart Thermostats/Valves

Table 11 Smart Thermostats/Valves

Name of New Component/Service:	*Smart Thermostats/Valves*
Type:	*Software*
Functionality:	*Interface software with the smart thermostats (Ecobee and Sensibo for the Cypriot pilot syte and Shelly TRV for the others) for collecting measurements and sending commands to the HVAC systems* - *Read and write room set point temperature,* - *Read and write fan mode,* - *Read and write operation mode (cooling/heating)*
Input Connections & Interfaces: From which components it receives input	*Ecobee smart thermostat - API* *Sensibo smart thermostat – API* *Shelly TRV MQTT Communication*
Output Connections & Interfaces: To which components it sends the results	*CI lightweight edge algorithms for the management of the local flexibility*
Relevant Use Cases	*UC01, UC02, UC03*

Input Parameters

Attribute/Para-meter	***Short Description***	***Data Type***	***Data Format***	***Value Range & Frequency***	***Data Received From***
Set point air temperature	*Current set point of the room air temperature (tag if it is defined by the user or the*	*Float64*	*JSON*	*Celsius (oC), [18,30], 4 samples per hour (every 15 minutes)*	*IoT/Smart thermostat*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

	Edge Node algorithms)				
Fan mode	*Current fan mode of the Fan Coil, Split, A/C Unit (tag if it is defined by the user or the Edge Node algorithms)*	*Float64*	*JSON*	*No units, [0,1,2,3] or [1,2,3], 4 samples per hour (every 15 minutes)*	*IoT/Smart thermostat*
Operation model	*Current operation mode of the Fan Coil, Split, A/C Unit (tag if it is defined by the user or the Edge Node algorithms)*	*string*	*JSON*	*No units, [heat, cool] or [heat, cool, off], 4 samples per hour (every 15 minutes)*	*IoT/Smart thermostat*
Output Parameters					
Attribute/Para-meter	***Short Description***	***Data Type***	***Data Format***	***Value Range & Frequency***	***Data Sent To***
Set point air temperature	*Current set point of the room air temperature (tag if it is defined by the user or the Edge Node algorithms)*	*Float64*	*JSON*	*Celsius (oC), [18,30], 4 samples per hour (every 15 minutes)*	*Edge Node*
Fan mode	*Current fan mode of the Fan Coil, Split,*	*Integer*	*JSON*	*No units, [0,1,2,3] or [1,2,3],*	*Edge Node*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

	A/C Unit (tag if it is defined by the user or the Edge Node algorithms)			*4 samples per hour (every 15 minutes)*	
Operation model	*Current operation mode of the Fan Coil, Split, A/C Unit (tag if it is defined by the user or the Edge Node algorithms)*	*string*	*JSON*	*No units, [heat, cool] or [heat, cool, off], 4 samples per hour (every 15 minutes)*	*Edge Node*

Software Requirements/Development Language	*Python (Pyecobee, Sensibo API), Shelly TRV MQTT communication*
Hardware Requirements	*Ecobee3 lite* *Sensibo Air* *Shelly TRV*
Communications	*WIFI connectivity*
Status of the development of the component	*partially developed*

7.1.3 Smart Plug

Table 12 Smart Plug

Name of New Component/Service:	*Smart Plug*
Type:	*Device*
Functionality:	• The plug supports on and off commands to activate and deactivate the device.

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

	• Set a max_power threshold setting in order to set the plug off in case of overpower • support auto_on and auto_off settings -- these are timers in seconds which will turn ON or OFF the plug when it has been turned OFF or ON respectively. Thus, the user can set a limit for how long the plug can be ON or OFF.
Input Connections & Interfaces: ***From which components it receives input***	• Physical button • HTTP request, through the local web interface (API request, sent through wifi) • A command sent via the cloud (through Shelly app)
Output Connections & Interfaces: ***To which components it sends the results***	Get data through MQTT subscription for the respective topics, monitor the situation through the respective web interface
Relevant Use Cases	UC01, UC02, UC03

Input Parameters					
Attribute/Pa ra-meter	***Short Description***	***Data Type***	***Data Format***	***Value Range & Frequency***	***Data Received From***
max_power	Overpower threshold	number	JSON	*Watt*	*Cloud, web interface settings*
led_power_d isable	Disable LED indication for output status	boolean	*JSON*		*Cloud, web interface settings*
actions	Set the actions parameters	hash	*JSON*		
Output Parameters					

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Attribute/Pa ra-meter	***Short Description***	***Data Type***	***Data Format***	***Value Range & Frequency***	***Data Sent To***
Relays (/status)	Contains the current state of the relay output channels.	array of hashes	*json*		*HTTP request to whatever agent makes the respective API call*
meters (/status)	Current status of the power meter	array of hashes	*json*		*HTTP request to whatever agent makes the respective API call*
Temperature (/status)	Internal device temperature in °C	*number*	*json*	*Celsius*	*HTTP request to whatever agent makes the respective API call*
Overtempera ture (/status)	*True when device has overheated*	*bool*	*json*		*HTTP request to whatever agent makes the respective API call*
tmp.tC (/status)	*Internal device temperature in °C*	*number*	*json*	*Celsius*	*HTTP request to whatever agent makes the respective API call*
tmp.tF (/status)	*Internal device temperature in °F*	*number*	*json*	*Farenheit*	*HTTP request to whatever agent makes the respective API call*
tmp.is_valid (/status)	*Whether the internal temperature*	*bool*	*json*		*HTTP request to whatever agent makes the respective API call*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

	sensor functions correctly				
Power (/meter/0)	Current real AC power being drawn	*number*	*json*	*watts*	*HTTP request to whatever agent makes the respective API call*
is_valid (/meter/0)	Whether power metering self-checks OK	*bool*	*json*		*HTTP request to whatever agent makes the respective API call*
Overpower (/meter/0)	Value in Watts, on which an overpower condition is detected	*number*	*json*		*HTTP request to whatever agent makes the respective API call*
Timestamp (/meter/0)	Timestamp of the last energy counter value, with the applied timezone	*Number*	*json*		*HTTP request to whatever agent makes the respective API call*
Counters (/meter/0)	Energy counter value for the last 3 round minutes in Watt-minute	*Array of numbers*	*json*		*HTTP request to whatever agent makes the respective API call*
Total (/meter/0)	Total energy consumed by the attached electrical appliance in Watt-minute	*Number*	*json*		*HTTP request to whatever agent makes the respective API call*

Software Requirements/Development Language	*The device API is language agnostic, what matters is making the proper HTTP request or MQTT topic subscription*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Hardware Requirements	*The plug itself, a way to connect to the web interface or the Shelly app or a server that acts as MQTT broker*
Communications	*MQTT enabled, make sure the topic for each option, both for input and output, is known.*
Status of the development of the component	*Already developed and usable. Documentation is open for use*

7.1.4 Sphensor

Table 13 Sphensor

Name of New Component/Service:	*Sphensor*
Type:	*Device*
Functionality:	*The Sphensor is able to measure thermo-hygrometric parameters such as temperature and relative humidity of the air, environmental parameters such as atmospheric pressure, illuminance for five different orientations, UV-A radiation and quality of the air in terms of the concentration of VOCs, CO2 and PM1, 2.5, 4, 10*
***Input Connections & Interfaces:* From which components it receives input**	*Building-Environmental conditions*
***Output Connections & Interfaces:* To which components it sends the results**	*It communicates with the Border Router through the Thread protocol that allows the connection and communication with the Edge Node.*
Relevant Use Cases	*UC01, UC02, UC03*
Input Parameters	

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Attribute/Para-meter	***Short Description***	***Data Type***	***Data Format***	***Value Range & Frequency***	***Data Received From***
Temperature		*Numerical*	*JSON*	*-30...60°C*	*Field*
Relative Humidity		*Numerical*	*JSON*	*0...100%*	*Field*
Atmospheric Pressure		*Numerical*	*JSON*	*600...1100 hPa*	*Field*
Lux		*Numerical*	*JSON*	*0.1...90 klx*	*Field*
UV-A		*Numerical*	*JSON*	*0...200 μW/cm2*	*Field*
VOC and equivalent CO2		*Numerical*	*JSON*	*0...1000 ppm*	*Field*
PM (1, 2.5, 4, 10)		*Numerical*	*JSON*	*0...1000 μg/m3*	*Field*
CO2		*Numerical*	*JSON*	*0...5000 ppm*	*Field*
Internal Temperature		*Numerical*	*JSON*	*-40±60 °C*	*Field*
Internal Pressure		*Numerical*	*JSON*	*700...1100 mbar*	*Field*
Output Parameters					
Attribute/Para-meter	***Short Description***	***Data Type***	***Data Format***	***Value Range & Frequency***	***Data Sent To***
Temperature		*Numerical*	*JSON*	*-30...60°C*	*Border Router (Edge Node)*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Relative Humidity		*Numerical*	*JSON*	*0...100%*	*Border Router (Edge Node)*
Atmospheric Pressure		*Numerical*	*JSON*	*600...1100 hPa*	*Border Router (Edge Node)*
Lux		*Numerical*	*JSON*	*0.1...90 klx*	*Border Router (Edge Node)*
UV-A		*Numerical*	*JSON*	*0...200 µW/cm2*	*Border Router (Edge Node)*
VOC and equivalent CO2		*Numerical*	*JSON*	*0...1000 ppm*	*Border Router (Edge Node)*
PM (1, 2.5, 4, 10)		*Numerical*	*JSON*	*0...1000 µg/m3*	*Border Router (Edge Node)*
CO2		*Numerical*	*JSON*	*0...5000 ppm*	*Border Router (Edge Node)*
Internal Temperature		*Numerical*	*JSON*	*-40±60 °C*	*Border Router (Edge Node)*
Internal Pressure		*Numerical*	*JSON*	*700...1100 mbar*	*Border Router (Edge Node)*

Software Requirements/Development Language	*The API is language agnostic, what matters is making the proper HTTP request or MQTT topic subscription. The communication with the border router happens via threads protocol*
Hardware Requirements	*Connection to the Border Router*
Communications	*MQTT enabled, make sure the topic for each option, both for input and output, is known.*
Status of the development of the component	*Already developed*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

7.1.5 BMS – EMS Server

Table 14 BMS - EMS Server

Name of New Component/Service:	*BMS - EMS Server (BMS System from EM Systemer)*
Type:	*Software* *Resides in a Building Automation Server*
Functionality:	*API for data exchange. Reading of temperature and air quality (CO2) levels from rooms and adjusting setpoint for room control or other loads in the building.*
Input Connections & Interfaces: From which components it receives input	*The BMS system will receive input from the BRiG via API.*
Output Connections & Interfaces: To which components it sends the results	*The result of the API query will be sent back to the BRiG device.*
Relevant Use Cases	*UC01, UC02, UC03*

Input Parameters					
Attribute/Para-meter	***Short Description***	***Data Type***	***Data Format***	***Value Range & Frequency***	***Data Received From***
Setpoint	*Indoor room temperature setpoint.*	*Object*	*JSON*	*Depends on the source and configuration of the BMS system.* *Often in the range from 10 to 30.*	*BRiG*
Other control setpoints	*Other control setpoints. E.g., heat*	*Object*	*JSON*	*Depends on the source and configuration*	*BRiG*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

	pump setpoint.			*of the BMS system.*	
Output Parameters					
Attribute/Para-meter	***Short Description***	***Data Type***	***Data Format***	***Value Range & Frequency***	***Data Sent To***
Indoor temperature	*Indoor room temperature as measured by the local thermostat or temperature sensor (the source varies from building to building).*	*Object*	*JSON*	*Depends on the source and configuration of the BMS system.* *Sample frequency varies, based on source and size of the overall project.*	*Depends on the hardware delivered in the specific BMS system.* • *Thermostat* • *Sensor connected to DDC controller* *All enquiries from BRiG will be returned to BRiG.*
CO2 value	*The indoor air quality measures in CO2 (ppm)*	*Object*	*JSON*	*0 -2000 PPM* *Sample frequency varies based on the source and size of the overall project*	*CO2 sensor, primarily 2-10V output read into a thermostat or DDC controller.*
Software Requirements/Development Language	*EMS Server (BMS Software)*				
Hardware Requirements	*Building Automation Server* *Industrial PC*				
Communications	*LAN*				
Status of the development of the component	*Developed.*				

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

	Can be revised or changed based on potential needs in the project.

7.1.6 Portal for access to multiple BMS

Table 15 Portal for access to multiple BMS

Name of New Component/Service:			*Portal for access to multiple BMS (EMPortal.no)* *Portal/Cloud solution for 1400 BMS systems delivered by EM Systemer AS.*		
Type:			*Software*		
Functionality:			*API for data exchange. Reading of temperature and air quality (CO2) levels from rooms and adjusting setpoint for room control or other loads in the building.*		
Input Connections & Interfaces: From which components it receives input			*The API will receive input from Central database NODERED based interface and deliver data to the Central server.*		
Output Connections & Interfaces: To which components it sends the results			*The result of the API query will be sent back to the Central server.*		
Relevant Use Cases			*UC01, UC02, UC03*		
Input Parameters					
Attribute/Para-meter	***Short Description***	***Data Type***	***Data Format***	***Value Range & Frequency***	***Data Received From***
GetMany/T	*Get all the analog objects defined in the given project/building*	*List of objects*	*JSON*	*Varies*	*Varies*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

GetMany/M	*Get all measurement zones defined in the give project/building.*	*List of objects*	*JSON*	*Varies*	*Varies*
GetGaugeData/0	*Get measures data from a given measurement zone.* *One measurement zone could for example be the AMS meter.*	*Object*	*JSON*	*Varies*	*Varies*
GetMany/A	*Get all the alarm status of the building. Also used to get status of motion sensors, window contacts and other digital variables*	*Object*	*JSON*	*Varies*	*Varies*
Output Parameters					
Attribute/Para-meter	***Short Description***	***Data Type***	***Data Format***	***Value Range & Frequency***	***Data Sent To***
Reply to given Query					*NTNU Server*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Software Requirements/Development Language	*EMPortal.no*
Hardware Requirements	*N/A*
Communications	*Internet access to www.EMPortal.no*
Status of the development of the component	*Done. Alterations and modification can be done based on COLLECTiEF project needs.*

7.1.7 APIs for setpoint control in small scale environment

Table 16 APIs for setpoint control in small scale environment

Name of New Component/Service:	*APIs for setpoint control in small scale environment*
Type:	*Software*
Functionality:	*APIs that allow the control from third party and the control inside the G2ELAB pilot site.* *Particularly, in the first case, the API permits the third party to send the new setpoints of comfort towards the intermediate plateform (called as SG-InterOp) by a third party. Essentially, the sent setpoints should be backed by a model predictive control developed by third party. SG-InterOp is an IOT based platform (with thingsboard at the backend) for storing and visualizing data.* *In the second case the API does the following tasks:* *Compare the setpoints of BMS and SG-InterOp and:* • *If values are different and latest values are found on SG-InterOp, it sets these values on BMS, log it, sends notifications to responsibles and stores the new values in influxDB database and csv file.* • *If values are different and latest values are found on BMS, it copes these values*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

	on SG-InterOp, log it, send notifications to responsibles and store the new values in influxDB database and csv file.
***Input Connections & Interfaces:* From which components it receives input**	• *Third party server* • *BMS G2ELab* • *SG-Interop G2ELab*
***Output Connections & Interfaces: To* which components it sends the results**	• *BMS G2ELab* • *SG-Interop G2ELab* • *Log* • *Pushbullet* • *Csv file* • *InfluxDB database*
Relevant Use Cases	*UC01, UC02, UC03*

Input Parameters

Attribute/Parameter	***Short Description***	***Data Type***	***Data Format***	***Value Range & Frequency***	***Data Received From***
Setpoints of BMS *Latest values of setpoints on SG-InterOp*	*The setpoints of both entities are compared for either remote modification on BMS or indicate a human interference*	*Float*	*List*	*Units;* *Thot = °C* *Tcold = °C* *CO2 = ppm* *Range:* *Temperature= 15°C to 30°C* *CO2 concentration = 300 to 1500 ppm*	*BMS G2ELab* *SG InterOp G2ELab*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Output Parameters					
Attribute/Para-meter	***Short Description***	***Data Type***	***Data Format***	***Value Range & Frequency***	***Data Sent To***
Setpoints of BMS *Setpoints on SG-InterOp*	-	*Float*	*HTTPS* *(SOAP Protocol for BMS)* *(Simple Post Request for SG InterOp)*	*Units;* *Thot = °C* *Tcold = °C* *CO2 = ppm* *Range:* *Temperature= 15°C to 30°C* *CO2 concentration = 300 to 1500 ppm*	*SG InterOp G2ELab* *BMS G2ELab*

Software Requirements/Development Language	*Programming language : Python* *Requirement : Pandas, Requests, Logging, SOAP Protocol, Pushbullet, InfluxDb*
Hardware Requirements	*Linux Server with crontab and InfluxDB*
Communications	*HTTPS (either SOAP or simple requests.post)*
Status of the development of the component	*Already developed*

7.2 Distributed Layer

7.2.1 Border Router + Edge Node MGT (iGateway)

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Table 17 Border Router + Edge Node MGT (iGateway)

Name of New Component/Service:	*BRiG Device (Border Router + Edge Node MGT (iGateway))*
Type:	*Firmware*
Functionality:	*Field sensors measurement data reception and local storage.* *Field actuators setting by Boolean status or continuous (percentage) value.* *Field devices (sensors and actuators) diagnostics reception and local storage.* *Service commands for radio network testing and maintenance.* *BrIG diagnostic status related to communication performance with field devices (signal quality, communication failures), local system performance (memory, CPU load); local storage of these information.* *Registry definition of all devices related to the specific BRiG (to which they are connected), including type (sensor, actuator), product model, communication parameter (protocol type, address, timings, etc.), local unique system identifier, user assigned name and description, etc.* *Local data retrieval capability specifying the time interval and data unique identifier.*
Input Connections & Interfaces:** **From which components it receives input	*iGateway (Edge Node) software subsystem inside BRiG through MQTT messages and JSON payloads.*
Output Connections & Interfaces:** **To which components it sends the results	*iGateway software subsystem inside BRiG through MQTT messages and JSON payloads.*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Relevant Use Cases			*UC01, UC02, UC03*		
Input Parameters					
Attribute/Parameter	***Short Description***	***Data Type***	***Data Format***	***Value Range & Frequency***	***Data Received From***
Sphensor measures value	*BRiG receives measurement data transmitted by Sphensor; all measurements are grouped into a single message*	*UTC DateTime of measurement instant.* *Floating point array*	*Binary*	*Value range: dependent on measure type. Frequency: asynchronous (push mode), dependent on configured tx rate, default 60"*	*Sphensor device*
Sphensor diagnostic information	*BRiG receives diagnostic and status information*	*Information kind dependent*	*Binary*	*Information kind dependent*	*Sphensor device*
Set BRiG device configuration (command)	*BRiG work mode is dependent of the parameters here specified. This includes i.e.: local network configuration, device name, external MQTT broker.*	*Command message unique identifier.* *Structure of BRiG configuration data*	*JSON*	*Information kind dependent; data is upgraded only in system/field configuration time*	*iGateway subsystem or other origin through MQTT message*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Read BRiG device configuration (command)	*Read the BRiG configuration set by the relative command.*	*Command message unique identifier.* *Command only specification*	*JSON*	--	*iGateway subsystem or other origin through MQTT message*
Add (set) field device (command)	*Each device managed from BRiG is appropriately configured to be managed from BRiG*	*Command message unique identifier.* *Structure of device information including communication information, device registry communication mode (data pushed from device, sampling rate, etc.)*	*JSON*	*Device kind dependent; data is upgraded only in system/field configuration time*	*iGateway subsystem or other origin through MQTT message*
Remove field device (command)	*Remove from the internal field device list the one with the specified*	*Command message unique identifier.* *Field device*	*JSON*	--	*iGateway subsystem or other origin through MQTT message*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

	unique identifier	*unique identifier*			
Read all field device list (command)	*Read a list of all field devices previously configured inside BRiG*	*Command message unique identifier.* *Unique device identifier;* *Device user assigned name*	*JSON*	*--*	*iGateway subsystem or other origin through MQTT message*
Set output status on field device (command)	*BRiG sends a command to a specific field device to set its output to the defined state*	*Command message unique identifier.* *Device unique id; device output channel; Boolean output status value*	*JSON*	*Device unique identifier must correspond to one defined in the device configuration list.* *Status value is boolean (off/on or 0/1).* *Channel number is dependent of the characteristics of the field device*	*iGateway subsystem or other origin through MQTT message*
Read output status on field device (command)	*BRiG reads the current output status from a*	*Command message unique identifier.*	*JSON*	*Device unique identifier must correspond to one defined in the device*	*iGateway subsystem or other origin through MQTT message*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

	specific field device	*Device unique id; device output channel*		*configuration list.* *Channel number is dependent of the characteristics of the field device*	
Set output value on Field device (command)	*BRiG sends a command to a specific field device to set its output to the defined proportional value*	*Command message unique identifier.* *Device unique id; device output channel; percentage of the full scale output value*	*JSON*	*Device unique identifier must correspond to one defined in the device configuration list.* *Proportional value is a percentage (0..100%) of the full analog output scale.* *Channel number is dependent of the characteristics of the field device*	*iGateway subsystem or other origin through MQTT message*
Read output value from field device (command)	*BRiG reads the current output proportional value from a specific field device*	*Command message unique identifier.* *Device unique id; device*	*JSON*	*Device unique identifier must correspond to one defined in the device configuration list.*	*iGateway subsystem or other origin through MQTT message*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

		output channel		*Channel number is dependent of the characteristics of the field device*	
Read measured value from field device (command)	*BRiG reads the current measured value from a specific field device*	*Command message unique identifier.* *Device unique id; device measure channel*	*JSON*	*Device unique identifier must correspond to one defined in the device configuration list.* *Channel number is dependent of the characteristics of the field device*	*iGateway subsystem or other origin through MQTT message*
Output Parameters					
Attribute/Para-meter	***Short Description***	***Data Type***	***Data Format***	***Value Range & Frequency***	***Data Sent To***
Sphensor measures value	*BRiG sends measurement data received from Sphensor; all measurements are grouped into a single message*	*UTC DateTime of measurement instant.* *Floating point array*	*JSON*	*Value range: dependent on measure type. Frequency: asynchronous (push mode), dependent on configured tx rate, default 60"*	*iGateway subsystem or other destination through MQTT message*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Sphensor diagnostic information	*BRiG sends diagnostic and status information received from Sphensor or elaborated by BRiG depending of the communication status with each Sphensor*	*Information kind dependent*	*Binary*	*Information kind dependent*	*iGateway subsystem or other destination through MQTT message*
Set BRiG device configuration confirmation status	*Result of the set BRiG configuration command*	*Command message unique identifier.* *Result code of the operation*	*JSON*	*Result ok, invalid data, general failure, etc.*	*iGateway subsystem or other destination through MQTT message*
Read BRiG device configuration response	*Answer of the read BRiG configuration, set by the relative command*	*Command message unique identifier.* *Structure of device configuration data.*	*JSON*	--	*iGateway subsystem or other destination through MQTT message*
Add (set) field device confirmation status	*Answer with the result of the add field device command*	*Command message unique identifier.* *Result code of*	*JSON*	*Result ok, invalid data, general failure, etc.*	*iGateway subsystem or other destination through MQTT message*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

		the operation			
Remove field device confirmation status	*Answer this the result of the remove field device command*	*Command message unique identifier.* *Result code of the operation*	*JSON*	*Result ok, invalid data, general failure, etc.*	*iGateway subsystem or other destination through MQTT message*
Read all field device list command response	*Answer with the resulting list of field devices previously configured inside BRiG*	*Command message unique identifier.* *List of all devices user assigned name and unique identifiers*	*JSON*	*--*	*iGateway subsystem or other destination through MQTT message*
Field device set output status command response	*Answer with the result of the set output status command*	*Command message unique identifier.* *Result code of the operation*	*JSON*	*Result ok, invalid data, general failure, etc.*	*iGateway subsystem or other destination through MQTT message*
Field device read output status command response	*Answer with the status of the field device read output status command*	*Command message unique identifier.* *Device unique id;*	*JSON*	*Device unique identifier must correspond to one defined in the device*	*iGateway subsystem or other destination through MQTT message*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

		device output channel. Field device output status value		configuration list. Channel number is dependent of the characteristics of the field device	
Field device set output value command response	*Answer with the result of the set output value command*	*Command message unique identifier.* *Result code of the operation*	*JSON*	*Result ok, invalid data, general failure, etc.*	*iGateway subsystem or other destination through MQTT message*
Field device read output value command response	*Answer with the status of the field device read output value command*	*Command message unique identifier.* *Device unique id; device output channel.* *Field device output value*	*JSON*	*Device unique identifier must correspond to one defined in the device configuration list.* *Channel number is dependent of the characteristics of the field device*	*iGateway subsystem or other destination through MQTT message*
Field device read measured value command response	*Answer with the status of the field device read measured*	*Command message unique identifier.*	*JSON*	*Device unique identifier must correspond to one defined in the device*	*iGateway subsystem or other destination through MQTT message*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

	value command	*Device unique id; device output channel.* *Field device measured value*		*configuration list.* *Channel number is dependent of the characteristics of the field device*	

Software Requirements/Development Language	*Command and configuration messages are transmitted through MQTT protocol using the BRiG internal MQTT broker or, for debugging purpose, any external MQTT broker; this is a configurable option.* *Data storage inside BRiG uses Linux file system and a light SQL database file as SQLite or Inflex.* *Main program language is Python. Specific library code for hi-performance need can be written in C/C++, if required.*
Hardware Requirements	*The RPi4 main memory storage can be a bottleneck for heavy data save & retrieval; an additional memory (i.e. USB stick) can be evaluated as additional mass storage.*
Communications	*BRiG uses these communication channels:* • *Thread protocol with a specific internal radio module for Sphensor communication.* • *Wi-Fi protocol with RPi4 internal radio module for smart-plug/valve and generic LAN/WAN communication (through a local router acting as a Wi-Fi Access Point).* • *Ethernet alternative of parallel to Wi-Fi for LAN/WAN communication (through a local router).* *MQTT protocol for internal BR-iG subsystems command and data exchange.*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Status of the development of the component	*Partially developed, only for Sphensor data communication.*

7.2.2 CI lightweight edge algorithms for the management of the local flexibility

Table 18 CI lightweight edge algorithms for the management of the local flexibility (Edge Node)

Name of New Component/Service:	*CI lightweight edge algorithms for the management of the local flexibility (Edge Node)*
Type:	*Software*
Functionality:	*An algorithm for computing the room air temperature set point for ensuring thermal comfort and health of occupants and for enabling energy flexibility and climate resilience.* 1. *Adaptive thermal comfort models (ASHRAE, EN)* 2. *Circadian Rhythm* 3. *Pre-cooling/pre-heating based on the flexibility demand signal* 4. *Pre-cooling/pre-heating based on forecasting of extreme weather conditions*
Input Connections & Interfaces: From which components it receives input	*Open-source Weather Databases (e.g., Meteostat)-API / Local Weather Station – API* *Cluster Node - JSON* *Sphensor Gateway - JSON* *Brig – JSON*
Output Connections & Interfaces: To which components it sends the results	*IoT Control Device (smart valves, smart thermostats, smart plugs) - API* *BMS - API*
Relevant Use Cases	*UC01, UC02, UC03*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Input Parameters					
Attribute/Para-meter	*Short Description*	*Data Type*	*Data Format*	*Value Range & Frequency*	*Data Received From*
Historical data of outdoor air temperature (for at least the last 30 days)	*A daily calculation of the Running mean average outdoor air temperature (Tave)*	*Float64, Time series (Dataframe)*	*JSON*	*Celsius (°C), [-30,60], one sample per day*	*Open Source Weather API using building coordinates / Local Weather Station*
Flexibility Signal		*int*	*XML/JSON*	*[0-5],* *4 samples per hour (every 15 minutes)*	*Cluster Node*
Current value of each zone Air Temperature	*PRMPB0402*	*Float64*	*JSON*	*Celsius (°C), [-30,60], 60 samples per hour (every 1 minute)*	• *Sphensor Gateway –* • *BRIG*
Current value of each zone Relative Humidity	*PRMPB0402*	*Float64*		*%RH, [0,100], 60 samples per hour (every 1 minute)*	• *Sphensor Gateway -* • *BRIG*
Current value of each zone Atmospheric pressure	*PRMPB0402*	*Float64*		*hPa, [600,1100], 60 samples per hour (every 1 minute)*	• *Sphensor Gateway -* • *BRIG*
Current value of each zone Illumination	*PRMPB0402*	*Float64*		*klx, [0.1,90], 60 samples per hour*	• *Sphensor Gateway -* • *BRIG*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

				(every 1 minute)	
Current value of each zone CO2, concertation	*PRMPA0423*	*Float64*		*ppm, [0,5000], 60 samples per hour (every 1 minute)*	• *Sphensor Gateway -* • *BRIG*
Current value of each zone VOC	*PRMPA0423*	*Float64*		*ppm, [0,1000], 60 samples per hour (every 1 minute)*	• *Sphensor Gateway -* • *BRIG*
Current value of each zone PM1	*PRMPA0423*	*Float64*		*µg/m3, [0,1000], 60 samples per hour (every 1 minute)*	• *Sphensor Gateway -* • *BRIG*
Current value of each zone PM2.5	*PRMPA0423*	*Float64*		*µg/m3, [0,1000], 60 samples per hour (every 1 minute)*	• *Sphensor Gateway -* • *BRIG*
Current value of each zone PM4	*PRMPA0423*	*Float64*		*µg/m3, [0,1000], 60 samples per hour (every 1 minute)*	• *Sphensor Gateway -* • *BRIG*
Current value of each zone PM10	*PRMPA0423*	*Float64*		*µg/m3, [0,1000], 60 samples per hour (every 1 minute)*	• *Sphensor Gateway -* • *BRIG*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Historical data of state of each zone		*Float64 Time series (Dataframe)*	*JSON*	*60 samples per hour (every 1 minute)*	*Central Database*
Historical data of set point temperature in each zone		*Float64 Time series (Datafram e*	*JSON*	*60 samples per hour (every 1 minute)*	*Central Database*
Output Parameters					
Attribute/Para-meter	***Short Description***	***Data Type***	***Data Format***	***Value Range & Frequency***	***Data Sent To***
Set point temperature in each zone		*Float64*	*JSON*	*Celsius, [18,30], 4 samples per hour (every 15 minutes)*	*IoT Control Device (smart valves, smart thermostats, smart plugs) – API*
Set point temperature in each zone		*Float64*	*JSON*	*Celsius, [18,30], 4 samples per hour (every 15 minutes)*	*BMS - API*

Software Requirements/Development Language	*Python*
Hardware Requirements	*None*
Communications	*None*
Status of the development of the component	*partially developed*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

7.2.3 Edge Node: Building Thermal Optimization

Table 19 Edge Node: Building Thermal Optimization

Name of New Component/Service:	*Edge Node: Building Thermal Optimization*
Type:	*Software solution.*
Functionality:	*Optimise building-wise heat demand with respect to a price signal (modulo a positive scalar) subject to constraints on flexibility.*
Input Connections & Interfaces: From which components it receives input	*Time series of (timestamp, value) with a slight variation depending on the scenario (electricity, or heat).* *1. A 24 hour and hourly or more frequent weather forecast of the relevant outdoor temperature.* *2. Hourly or more frequent building outdoor temperature.* *2.1. Heat pump source temperature and secondary side supply temperature set point, and electricity demand.* *2.2. District heating secondary side temperature and the corresponding secondary side temperature set point, and heat demand.* *3. Hourly or more frequent building indoor temperatures and the corresponding thermostat temperature set points.* *4. Cluster Node price signal.*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Output Connections & Interfaces: ***To which components it sends the results***	*Time series of (timestamp, value) with a slight variation depending on the scenario (electricity, or heat).* *1. Demand forecast.* *2. Temperature set point signals.*
Relevant Use Cases	*UC01, UC02, UC03*

Input Parameters

Attribute/Para-meter	***Short Description***	***Data Type***	***Data Format***	***Value Range & Frequency***	***Data Received From***
1.	*A 24 hour and hourly or more frequent weather forecast of the relevant outdoor temperature.*	*Atime series of (timestamp, value).*	*JSON and/or local SQLite database*	*Temperatures (K, resolution 10^{-3}) with a frequency between 12/h and 1/h.*	*Central database.*
2.	*Hourly or more frequent building outdoor temperature.*	*A time series of (timestamp, value).*	*JSON and/or local SQLite database*	*Temperatures (K, resolution 10^{-3}) with a frequency between 12/h and 1/h.*	*Edge Node sensors.*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

2.1.	*Heat pump source temperature and secondary side supply temperature set point, and electricity demand.*	*A collection of time series of (timestamp, value).*	*JSON and/or local SQLite database*	*Temperatures (K, resolution 10^{-3}) with a frequency between 12/h and 1/h, and mean power values (W, resolution 10^{0}) with a frequency between 12/h and 1/h.*	*Edge Node sensors.*
2.2.	*District heating secondary side temperature and the corresponding secondary side temperature set point, and heat demand.*	*A collection of time series of (timestamp, value).*	*JSON and/or local SQLite database*	*Temperatures (K, resolution 10^{-3}) with a frequency between 12/h and 1/h, and mean power (W, resolution 10^{0}) with a frequency between 12/h and 1/h.*	*Edge Node sensors.*
3.	*Hourly or more frequent building indoor temperatures and the corresponding thermostat temperature set points.*	*A collection of time series of (timestamp, value).*	*JSON and/or local SQLite database*	*Temperatures (K, resolution 10^{-3}) with a frequency between 12/h and 1/h.*	*Edge Node sensors.*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

4.	*Cluster Node price signal.*	*A time series of (timestamp, value).*	*JSON and/or local SQLite database*	*Unit (1, resolution 10^{-6}) with a frequency between 12/h and 1/h.*	*Cluster Node broadcast.*
Output Parameters					
Attribute/Para-meter	***Short Description***	***Data Type***	***Data Format***	***Value Range & Frequency***	***Data Sent To***
1.	*Demand forecast.*	*A time series of (timestamp, value).*	*JSON and/or local SQLite database*	*Mean power values (W, resolution 10^{0}) with a frequency between 12/h and 1/h.*	*The Central database, and the Cluster Node Fleet Management*
2.	*Temperature set point signals.*	*Time series of (timestamp, value).*	*JSON and/or local SQLite database*	*Unit (1, resolution 10^{-6}) with a frequency between 12/h and 1/h.*	*Edge Node actuators.*
Software Requirements/Development Language			*Python, SQLite, and a compatible MQTT solution, perhaps [paho-mqtt](*https://pypi.org/project/paho-mqtt/*).*		
Hardware Requirements			Resources for running the software solution.		
Communications			Access to a dedicated MQTT broker for cluster/edge and edge/edge communication, and a REST API to access the Central database to the end of populating		

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

	the local databases, and to access continuous weather and price forecasts.
Status of the development of the component	Started.

7.3 Cluster Layer

7.3.1 Cluster Node Collective Privacy

Table 20 Cluster Node Collective Privacy

Name of New Component/Service:	*Cluster Node Collective Privacy*				
Type:	*Architectural principle.*				
Functionality:	*Restrict the solutions to never communicate private data and require the Central database API to implement differential privacy.*				
Input Connections & Interfaces: From which components it receives input	*Edge Node, Central Database*				
Output Connections & Interfaces: To which components it sends the results	*Cluster Node*				
Relevant Use Cases	*UC03, UC04*				
Input Parameters					
Attribute/Parameter	***Short Description***	***Data Type***	***Data Format***	***Value Range & Frequency***	***Data Received From***

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

N/A	N/A	N/A	N/A	N/A	N/A
Output Parameters					
Attribute/Par a-meter	**Short Description**	**Data Type**	**Data Format**	**Value Range & Frequency**	**Data Sent To**
N/A	N/A	N/A	N/A	N/A	N/A
Software Requirements/Development Language	N/A				
Hardware Requirements	N/A				
Communications	N/A				
Status of the development of the component	to be developed from scratch				

7.3.2 Cluster Node Fleet Manager

Table 21 Cluster Node Fleet Manager

Name of New Component/Service:	*Cluster Node Fleet Manager*
Type:	*Software solution.*
Functionality:	*Subject to the constraints dictated by Collective Privacy, use MQTT to collect self-reported cluster/edge and edge/edge communication, and report on the system health and performance in a way that facilitates strategic decisions on fleet operation and maintenance.*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

***Input Connections & Interfaces:* From which components it receives input**			*Access to a dedicated MQTT broker for cluster/edge and edge/edge communication.*		
***Output Connections & Interfaces:* To which components it sends the results**			*A continuously populated SQLite database of self-reported cluster/edge and edge/edge communication and additional key performance indicators.*		
Relevant Use Cases			*UC01, UC03, UC04*		
Input Parameters					
Attribute/Para-meter	***Short Description***	***Data Type***	***Data Format***	***Value Range & Frequency***	***Data Received From***
Everything communicated through the MQTT broker.					
Output Parameters					
Attribute/Par a-meter	***Short Description***	***Data Type***	***Data Format***	***Value Range & Frequency***	***Data Sent To***
Everything communicated through the MQTT broker and additional key performance indicators.					
Software Requirements/Development Language			*Python, SQLite, and a compatible MQTT solution, perhaps [paho-mqtt]*(https://pypi.org/project/paho-mqtt/ *).*		
Hardware Requirements			Resources for running the software solution.		
Communications			Access to a dedicated MQTT broker for cluster/edge and edge/edge communication.		
Status of the development of the component			Not started.		

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

7.3.3 Cluster Node: Network Thermal Optimization

Table 22 Cluster Node: Network Thermal Optimization

Name of New Component/Service:	*Cluster Node: Network Thermal Optimization*
Type:	*Software solution.*
Functionality:	*Optimise network-wise heat demand by constructing a price signal (modulo a positive scalar) to the end of advising the edge nodes of when to consume more and when to consume less.*
Input Connections & Interfaces: From which components it receives input	*Time series of (timestamp, value) with a slight variation depending on the scenario (electricity, or heat).* *1. A 24 hour and hourly or more frequent weather forecast of the relevant outdoor temperature.* *2. A 24 hour and hourly or more frequent price forecast for the modulated resource (electricity or heat), or a closed form price signal (modulo a positive scalar) tailored to the scenario.* *3. The hourly or more frequent report of the total demand of the modulated resource (electricity, or heat) for the selected building stock. Note that, to achieve the desired outcome, the selected building stock need to reflect the demand to be modulated, which can be different from the demand of the modulated buildings, i.e., the demand of unmodulated buildings are likely to affect the location of the peek demand.*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Output Connections & Interfaces: To which components it sends the results	*Time series of (timestamp, value) with a slight variation depending on the scenario (electricity, or heat).* *1. Demand forecast.* *2. Price signal (modulo a positive scalar).*
Relevant Use Cases	*UC01, UC03, UC04*

Input Parameters					
Attribute/Parameter	***Short Description***	***Data Type***	***Data Format***	***Value Range & Frequency***	***Data Received From***
1.	*A 24 hour and hourly or more frequent weather forecast of the relevant outdoor temperature.*	*A time series of (timestamp, value).*	*JSON and/or local SQLite database*	*Temperatures (K, resolution 10^{-3}) with a frequency between 12/h and 1/h.*	*The Central database.*
2.	*A 24 hour and hourly or more frequent price forecast for the modulated resource (electricity or heat), or a closed form price signal (modulo a positive scalar) tailored to the scenario.*	*A time series of (timestamp, value).*	*JSON and/or local SQLite database*	*Unit (1, resolution 10^{-6}) with a frequency between 12/h and 1/h, or a formula on closed form.*	*The Central database, or local configuration.*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

3.	*The hourly or more frequent report of the total demand of the modulated resource (electricity, or heat) for the selected building stock.*	*A time series of (timestamp, value).*	*JSON and/or local SQLite database*	*Mean power values (W, resolution 10^{0}) with a frequency between 12/h and 1/h.*	*The Central database.*
Output Parameters					
Attribute/Para-meter	***Short Description***	***Data Type***	***Data Format***	***Value Range & Frequency***	***Data Sent To***
1.	*Demand forecast.*	*A time series of (timestamp, value).*	*JSON and/or local SQLite database*	*Mean power values (W, resolution 10^{0}) with a frequency between 12/h and 1/h.*	*Cluster Node fleet management, Edge Node: Building Thermal Optimization*
2.	*Price signal (modul, a positive scalar).*	*A time series of (timestamp, value).*	*JSON and/or local SQLite database*	*Unit (1, resolution 10^{-6}) with a frequency between 12/h and 1/h.*	*Cluster Node fleet management, Edge Node: Building Thermal Optimization*
Software Requirements/Development Language	*Python, SQLite, and a compatible MQTT solution, perhaps [paho-mqtt](https://pypi.org/project/paho-mqtt/).*				
Hardware Requirements	*Resources for running the software solution.*				

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Communications	*Access to a dedicated MQTT broker for cluster/edge and edge/edge communication, and a REST API to access the Central database to the end of populating the local databases, and to access continuous weather and price forecasts.*
Status of the development of the component	*Started.*

7.4 Application Layer

7.4.1 Human-Building Interface (Edge Node)

Table 23 Human-Building Interface (Edge Node)

Name of New Component/Service:	*Human-Building Interface (Edge Node)*
Type:	*Software – Run on the BRIG*
Functionality:	*The interface will give the ability to the user on the Edge Node (i.e., Building Occupant, Manager, Owner) to* • ***Visualise*** *data and analytics related to the indoor environmental conditions (i.e., thermal comfort, indoor air quality, and light), energy efficiency, cost savings, and data and trends meaningful and valuable to the each user category.* • ***Interact*** *with edge node CI-based building control and management algorithms for setting up user preferences related to the operation of the control field devices i.e., smart valves, smart thermostats, existing BMS, such as schedules for allowing or not flexibility, setting points to the control filed devices, etc.* • ***Suggest*** *optimization strategies for the optimal control of energy and environmental, notify alarms, collect user feedback*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

***Input Connections & Interfaces:* From which components it receives input**	*Border Rooter (for data from the field devices)* *CI-based building control and management algorithms* *Building thermal optimization (Indoor Environmental Quality KPIs)* *User (i.e., Building Occupant, Manager, Owner)*
***Output Connections & Interfaces:* To which components it sends the results**	*Edge Node* • *CI-based building control and management algorithms* • *User (i.e., Building Occupant, Manager, Owner)* *Cluster Node:* • *Network Optimization, thermal network optimization and aggregated flexibility management* • *Collective Intelligence based control system*
Relevant Use Cases	*UC01, UC02*

Input Parameters					
Attribute/Para-meter	***Short Description***	***Data Type***	***Data Format***	***Value Range & Frequency***	***Data Received From***
N/A	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*

Output Parameters					
Attribute/Para-meter	***Short Description***	***Data Type***	***Data Format***	***Value Range & Frequency***	***Data Sent To***
N/A	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Software Requirements/Development Language	*Python, Influx DB and/or SQLite DB, MQTT based communication*
Hardware Requirements	*Sensors/Actuator data, RPi 4 on which deploy the SW*
Communications	*REST API, MQTT based communication*
Status of the development of the component	*This software component is going to be developed from scratch mainly by VIRTUAL in collaboration with the rest of the partners involved in T3.4. At the moment we are in the process of deriving the concept of the Human-Building interface based on the literature, existing Human-Building interfaces used in the industry, and a Questionnaire (see preliminary version of the Questionnaire in Annex 3/Figure 22) developed by the partners of the T3.4. The Questionnaire aims to conceptualize the exact functionalities of the interface. Specifically, it aims to identify the content of the HB-interface based on user category, using information form the list of KPIs (provided by D5.1.), the descriptive data of the units (room, apartment, building, cluster of buildings), and the output data of the COLLECTiEF's software components developed at both the Edge and Cluster Node. Moreover, the Questionnaire aims to extract the way the data will be visualized to each user category in order to be user-friendly and valuable.*

7.4.2 Fully Integrated Dashboard (Cluster Node)

Table 24 Fully Integrated Dashboard (Cluster Node)

Name of New Component/Service:	*Fully Integrated Dashboard (Cluster Node)*
Type:	*Software – Run on a Cloud or Server*
Functionality:	*Fully Integrated dashboard aims to visualize information at the level of cluster of buildings to*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

<table>
<tr><td colspan="3"></td><td colspan="3">users such as property owners and managers and energy providers (DSO).</td></tr>
<tr><td colspan="3"><u>Input Connections & Interfaces:</u> From which components it receives input</td><td colspan="3">Manly it will receive input information from the following components:
Custer Node software components such as
Collective Fleet Management
Noda Network Optimization, thermal network optimization and aggregated flexibility management
Collective Intelligence based control system
Data and Trend analytics</td></tr>
<tr><td colspan="3"><u>Output Connections & Interfaces: </u>To which components it sends the results</td><td colspan="3">High Level data visualization for the end user</td></tr>
<tr><td colspan="3"><u>Relevant Use Cases</u></td><td colspan="3">UC04</td></tr>
<tr><td colspan="6"><u>Input Parameters</u></td></tr>
<tr><td>Attribute/Para-meter</td><td>Short Description</td><td>Data Type</td><td>Data Format</td><td>Value Range & Frequency</td><td>Data Received From</td></tr>
<tr><td>N/A</td><td>N/A</td><td>N/A</td><td>N/A</td><td>N/A</td><td>N/A</td></tr>
<tr><td colspan="6"><u>Output Parameters</u></td></tr>
<tr><td>Attribute/Para-meter</td><td>Short Description</td><td>Data Type</td><td>Data Format</td><td>Value Range & Frequency</td><td>Data Sent To</td></tr>
<tr><td>N/A</td><td>N/A</td><td>N/A</td><td>N/A</td><td>N/A</td><td>N/A</td></tr>
<tr><td colspan="3">Software Requirements/Development Language</td><td colspan="3">Python</td></tr>
<tr><td colspan="3">Hardware Requirements</td><td colspan="3">Cloud</td></tr>
<tr><td colspan="3">Communications</td><td colspan="3">REST API</td></tr>
</table>

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Status of the development of the component	*This software component is going to be developed from scratch mainly by VIRTUAL in collaboration with the rest of the partners involved in T3.4. The partners of the T3.4 are in the process of deriving the architecture of the Fully Integrated Dashboard based on a Questionnaire (see preliminary version of the Questionnaire in Annex 3/Figure 22). The Questionnaire aims to conceptualize the exact functionalities of the interface. Specifically, it aims to identify the content of the dashboard based on user category, using information form the list of KPIs (provided by D5.1.), the descriptive data of the units (cluster of buildings), and the output data of the COLLECTiEF's software components developed at the Cluster Node. Moreover, the Questionnaire aims to extract the way the data will be visualized to each user category in order to be user-friendly and valuable.*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

8 Conclusion

This report has presented the architectural design of the COLLECTiEF system along with the respective system specifications. The methodology followed to define the details of the architectural elements that make up the architecture of the COLLECTiEF system was initially described. The COLLECTiEF conceptual architecture was then analysed, as well as the structural, development, deployment and dynamics view of the system.

The architectural views and perspectives presented in this result will further guide design and implementation during the project's lifetime. UML diagrams illustrate the main components of the COLLECTiEF system, the key players and how they interact with the system. Moreover, the detailed description of the architectural elements provides a comprehensive view of the COLLECTiEF components focusing mainly on its major architectural elements and providing the first version of the project use-cases and sequence diagrams, which allow to reason and describe the dynamic behaviour of the system. The analysis presented for each architectural element, which includes data inputs and outputs, the interrelation between system entities and correlation with their respective use cases and key system requirements, will allow developers and component integrators to communicate about architectural problems in the most efficient and effective way.

Although no substantial changes are expected to the overall architecture of the COLLECTiEF system and its core components, this report can be considered as a living document that will address further refinements that might be necessary in case of new modifications that will come up during the implementation phase. It is worth mentioning that all key partners responsible for component development were involved in defining the architectural elements process. The involvement of developers in the architecture refinement process was significant because it led to a more coherent definition of the architecture (and its architectural elements), which also encompassed the point of view of the developers.

In summary, this result provides a sufficient basis for the technical developments of the COLLECTiEF system that will take place in WP3, while the actual architectural elements of each framework will be implemented (WP3) and validated (WP4).

.

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Annex 1: Use Case Description Template

Table 25 Use Case description template

UC Description	
UC Name	Use Case name, which uniquely identifies the UC (e.g. unique identifier), having an achievable goal
Version	To inform the user the stage a use case has reached.
Authors	Who created and who documented the Use Case
Last Update	Date of the last update
Brief Description	Description of the series of steps for the defined use case in a clear concise manner. Including what the COLLECTiEF system shall do for the involved actor to achieve a particular goal.
Assumptions and Pre-Conditions	The conditions that generally does not change during the execution and should be true to successfully terminate the use case. Moreover, pre-conditions define all the conditions that must be met (i.e., it describes the state of the system) to meaningfully cause the initiation of the use case.
Goal (Successful End Condition)	The ultimate aim and end condition(-s) of the Use Case
Post-Conditions	The state of the world upon successful completion
Involved Actors	Who are the actors involved in the use case? The same actor may play two different roles in the same use case. An actor may be a person, a device, another system or sub-system, or time. Actors represent different roles that something outside has in its relationship with the system, functional requirements of which are being specified.
UC Initiation	This refers to the potential triggers or events that could initiate the use case. The type of trigger can be temporal, internal or even in respond to an external event. Normally, the initiation of a UC shall take into account also the pre-conditions, e.g., checking them prior the execution of the UC.
Main Flow	Unconditional set of steps that describe how the use case goal can be achieved and all related stakeholder interests can be satisfied.

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Alternative Courses	Description of the alternative course of events.
Relationships with other UCs	Indication of connection with other use cases
Architectural Elements / Services Involved	Indication of COLLECTiEF elements involved
UML Sequence Diagram	
Diagram showing process interactions arranged in time sequence in the field of software engineering.	

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Annex 2: Architectural Specifications Template

Table 26 Architectural Components Detailed Specifications Template

Name of New Component/Service:	*< name of the architectural element e.g., Edge Node>*
Type:	*<Component, Software, Device etc.>*
Functionality:	*<please write here a short description of the operation of this module/component. A list of functions and operations could be valuable.>*
Input Connections & Interfaces: From which components it receives input	*<please write the components from which it receives input (input dependencies) along with the available connection interfaces e.g., API etc.>*
Output Connections & Interfaces: To which components it sends the results	*<please write the components to which it sends the results (output dependencies) and mention also the available interfaces e.g., API etc.>*
Relevant Use Cases	*<please specify the use case(s) where the module is participating, by writing the IDs of the respective use cases>*

Input Parameters

Attribute/Parameter	***Short Description***	***Data Type***	***Data Format***	***Value Range & Frequency***	***Data Received From***
<please mention the input parameters. Each row corresponds to a single parameter>	*<mention a short description of the input parameter if necessary>*	*<please mention the data type of this parameter (e.g., int, string, etc. or*	*<e.g. XML, JSON etc.>*	*<indicate measurement unit and range of values for this attribute/parameter and*	*<please mention the source component or module that provides input data to this parameter>*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

		complex type, e.g., list, object, etc.)		*frequency-sample rate>*	
Output Parameters					
Attribute/Para-meter	***Short Description***	***Data Type***	***Data Format***	***Value Range & Frequency***	***Data Sent To***
<please mention the input parameters. Each row corresponds to one parameter>	*<mention a short description of the input parameter if necessary>*	*<please mention the data type of this parameter (e.g., int, string, etc. or complex type, e.g., list, object, etc.)*	*<e.g. XML, JSON etc.>*	*<indicate measurement unit and range of values for this attribute/parameter and frequency-sample rate>*	*<please write the source component or module that provides input data to this parameter>*
Software Requirements/Development Language	*<specify the software requirements that are related to the architectural element, explain the Programming Language that is used during the development of the component>*				
Hardware Requirements	*<specify the hardware requirements of the module, giving specifications about the hardware requirements which are necessary for the best functionality of the component>*				

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

	In case of any extra sensors needed, which will also be included in the sensor specification template, it can be cross-referred here.
Communications	*<address specific communication requirements either for data input or for data output>*
Status of the development of the component	*<specify if the component is "already developed" or "partially developed" or "to be developed from scratch">*

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Annex 3: User-friendly human-building interface: the concept for COLLECTiEF Edge Node

Building interfaces and how their design, context (e.g., location), and underlying logic have a significant impact on their usability and occupants' perceived control, resulting comfort and energy performance of buildings. Human-building interface interactions are complex, more research is required to understand design, use, and characteristics. In the past, researchers attempted to explain occupants' interactions with buildings. A research paper [10] made an effort to model building-human interactions as illustrated in Figure 19.

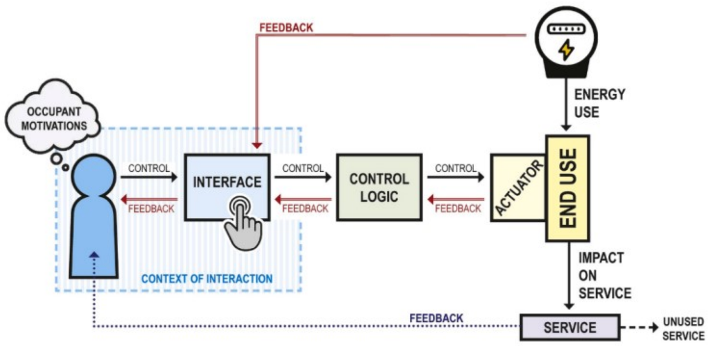


Figure 19 Conceptual model for understanding the occupant engagement with building interfaces

From the IEA-EBC Annex 79 [11] perspective, occupant-centric building design means to place occupant needs (comfort and health) as a priority and to use explicit occupant modelling to design building interfaces and recognize the bi-directional interaction between occupants and buildings. A paradigm shift is required, whereby practitioners transition from seeing occupants as sources of indoor heat gains and contaminants who are content with standardized indoor environmental conditions to understand that there is a complex and dynamic bi-directional interaction between occupants and buildings. This shift, which is presented in Figure 20, is supported by new knowledge, increased recognition of the value of healthy and comfortable environments, new enabling technologies and techniques (sensing and communication technology, analytical methods, computational power), and acknowledgement that occupants are an increasingly important factor for low- energy buildings.

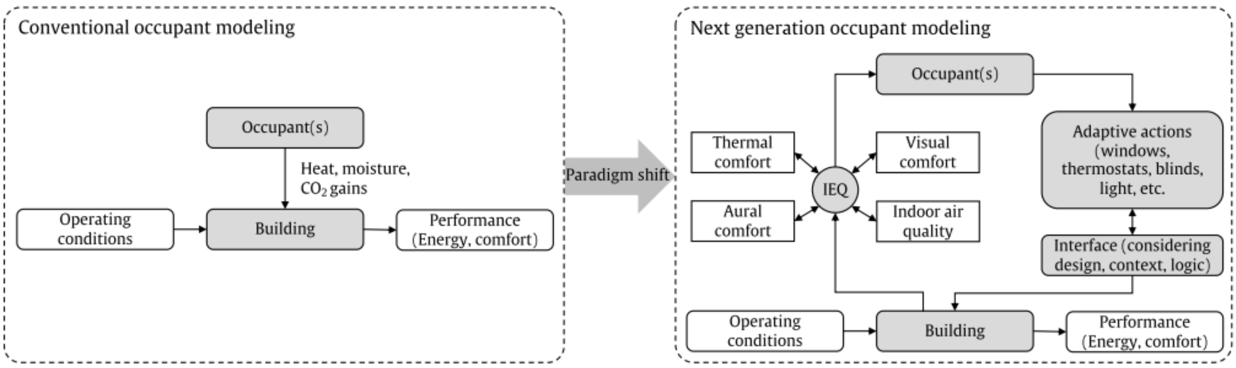


Figure 20 Paradigm shift from occupants and passive participants in buildings to active and dynamic elements in a complex two-way relationship

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Occupant-centric control (OCC) involves the sensing of actual indoor environmental quality, occupants' presence, and occupants' interactions with buildings and feeding this information directly back into control algorithms to achieve both high levels of energy efficiency and comfort, while maintaining usability and perceived control. The seven broad OCC categories listed in Figure 21 were defined. The first three of these categories rely on presence sensing technologies to modify building-level on-off or zone-level temperature setback schedules for HVAC and to switch off electric lighting in vacant lighting zones. The categories four and five modulate the airflow rates of HVAC systems (e.g., AHUs and VAV terminal units) at the building- and zone-level, respectively. The categories six and seven adapt the indoor temperature and illuminance setpoints during occupancy to preferred indoor climatic conditions. For each of the seven broad control categories, several experiments in different climatic conditions and building types will be carried out.

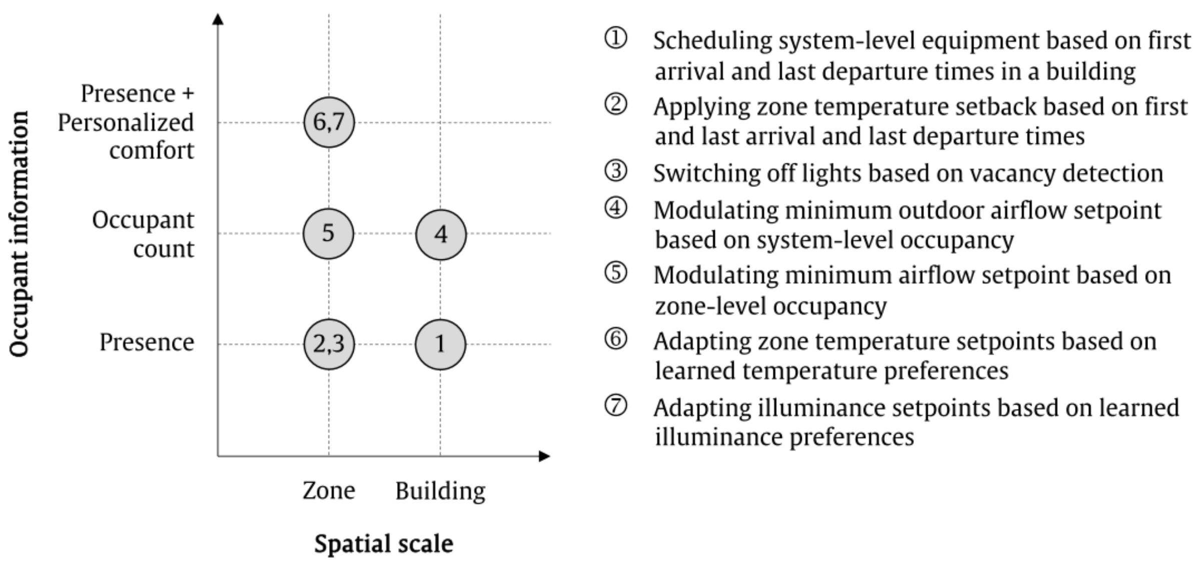


Figure 21 Case Study structure for occupant-centric control strategies

In conclusion, occupant-centricity also has to be an integral part of building operation and management. Consequently, occupant-centric control (OCC) strategies in order to detect occupants, learn about their comfort requirements, and train models on their behaviour have to be implemented into building management systems. Research has to be conducted to find the right balance between manual and automated controls and on how the design and configuration of control interfaces affect the performance of OCC algorithms. Further, the suitability of different OCC strategies for the respective building use has to be investigated.

The COLLECTiEF project will take into consideration the aforementioned studies on the modelling of the human-building interfaces and the need for occupant-centric control strategies, for designing a human-building interface for the Edge Node. The objective is to define a concept that is capable of receiving feedback from the user and using machine-learning techniques to adapt the indoor environmental condition based on the user preference but without excluding the needs for energy flexibility. In order, first to identify the key functionalities of the human-building interface, a Questionnaire is developed see Figure 22. This preliminary version of the Questionnaire aims to identify the content of the HB-interface based on user category, using information from the list of KPIs (provided by D5.1.), the descriptive data of the units (room, apartment, building, cluster of buildings), and the output data of the COLLECTiEF's software components developed at both the

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

Edge and Cluster Node. Moreover, the Questionnaire aims to extract the way the data will be visualized to each user category in order to be user-friendly and valuable.

					User categories										Data Visualization									
					Edge Node Users					Cluster Node Users					Single value		Time series						Bar Chart	Heatmap
			Data availablity		Occupant	Building Owner		Building Manager		Property Manager		Property Owner		DSO			Last 12 hours							
			Yes/No	Available temporal resolution		Consumer	Prosumer	Consumer	Prosumer	Consumer	Prosumer	Consumer	Prosumer		Current Value	Mean value	average/1 hour	average/12 hours	24 hours	Week	Month	Year		
Enviromental parameters	Indoor	1. Air temperature	Yes	1 sample/min																				
		2. Mean radiant temperature																						
		3. Globe temperature																						
		4. Operative temperature	Yes	1 sample/min																				
		5. Surface temperatures																						
		6. Air velocity																						
		7. Metabolic rate	Yes (POE q	twice a week																				
		8. Clothing insulation	Yes (POE q	twice a week																				
		9. Absolute humidity of the air																						
		10. Relative humidity	Yes	1 sample/min																				
	Outdoor	1. Outdoor air temperature	Yes	1 sample/ 10 min																				
		2. Outdoor running mean temperature	Yes	1 sample/ 10 min																				
		3. Prevailing mean outdoor air temperature	Yes	1 sample/ 10 min																				
		4. Global horizontal solar irradiance	Yes	1 sample/ 10 min																				
		5. Global horizontal solar irradiation	Yes	1 sample/ 10 min																				
	KPI	1. PMV - Predicted Mean Vote	Yes (1)																					
		2. PPD - Predicted Percentage of Dissatisfied	Yes (1)																					
		3. ALD - ASHRAE Likelihood of Dissatisfaction																						
		4. NaOR - Nicol's et a.'s Overheating Risk																						
		5. Percentage of time outside a PMV range																						
		6. Percentage of time outside an operative temperature range																						
		7. Degree-hours																						
		8. PPD-weighted criterion																						
		9. Average PPD																						
		10. Sum PPD																						
	indoor air quality	1. Carbon dioxide (CO2)	Yes	1 sample/min																				
		2. Total volatic compounds (TVOCs)	Yes	1 sample/min																				
		3. Particulate matter (PM10, PM2.5)	Yes	1 sample/min																				
	Light	1. Illuminance	Yes	1 sample/min																				
		2. Daylight Autonomy (DA)																						
		3. Continuous Daylight Autonomy (DA_CON)																						
		4. Useful Daylight Illuminance (UDI)																						
		5. Illuminance Uniformity (U_0)																						
		6. Wienold et al.'s simplification of Discomfort Glare Probability																						
		7. Hviid et al.'s simplification of Discomfort Glare Probability																						
Energy Flexibility		1. Change of electric load on the grid																						
		2. Electric self-consumption change (in case of an electric local production)																						
		3. Primary energy balance																						
		Climate resilience and flexibility:																						
		1. Flexibility Factor (DFF)																						
		2. Agility Factor (AF)																						
Discriptive data of unit (taking into account data protection, knowledge, competences of each user category)	Room	Number of thermal zones			x																			
		Area			x																			
		Oriantation			x																			
	Appartment / Building	Number of rooms																						
		Area																						
		location																						
		Cluster of Buildings																						
		Power Distribution Level																						
Edge Node Componets	Edge DBMS																							
	Edge Node Resource Management																							
	Device identity Management																							
	CI lightweight edge algorithms for the management of the local flexibility																							
	Building thermal optimization																							
Cluster Node Componets	Persistent Data Storage and Management (DBMS such as PostgreSQL, MySQL, MongoDB or AWS)																							
	Collective privacy (to be applied to the central database)																							
	Collective Fleet Management																							
	Noda Network Optimization, thermal network optimization and aggregated flexibility management																							
	Collective Intelligence based control system																							
	Data and Trend analytics (visualization -> move to application layer)																							

Figure 22 A sample of the preliminary Questionnaire developed to conceptualize the exact functionalities of the Human-Building interface and the Fully Integrated Dashboard.

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683

References

[1]. Kruchten, P. B. (1995). The 4+ 1 view model of architecture. IEEE software, 12(6), 42-50.
[2]. MQTT protocol, https://mqtt.org/.
[3]. Modbus Protocol, https://modbus.org/.
[4]. Bacnet Communication Protocol, http://www.bacnet.org/.
[5]. NODA Self-host, https://github.com/self-host/self-host#documentation
[6]. IEC 62559-2:2015: UC methodology – Part 2: Definition of the Templates for UCs, Actor List, and Requirements List, 2015.
[7]. Agile Modelling, UML2 Use Case Diagrams, available under http://www.agilemodeling.com/artifacts/useCaseDiagram.htm (state on September 2013).
[8]. Carnegie Mellon community, UML Use Case Diagrams: Tips and FAQ, available under http://www.andrew.cmu.edu/course/90-754/umlucdfaq.html (state on September 2013).
[9]. StarUML ,Modelling with Class Diagram, available under http://staruml.sourceforge.net/docs/user-guide(en)/ch05_2.htm
[10]. J. K. Day et al., "A review of select human-building interfaces and their relationship to human behavior, energy use and occupant comfort," Build. Environ., vol. 178, Jul. 2020, doi: 10.1016/J.BUILDENV.2020.106920.
[11]. W. O'Brien et al., "Introducing IEA EBC annex 79: Key challenges and opportunities in the field of occupant-centric building design and operation," Build. Environ., vol. 178, p. 106738, Jul. 2020, doi: 10.1016/J.BUILDENV.2020.106738.

This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No 101033683