The BRIGHT project is co-founded by the EU's Horizon 2020 innovation programme under grant agreement No 957816

Boosting DR through increased communIty-level consumer engaGement by combining Data-driven and blockcHain technology Tools with social science approaches and multi-value service design

Deliverable D2.3 DR Technologies and Tools

Author(s): Denisa Ziu (ENG), Andrea Lazzeri (ENG), Vincenzo Croce(ENG)

Imprint

Title:	**DR Technologies and Tools**
Contractual Date of Delivery to the EC:	31.10.2021
Actual Date of Delivery to the EC:	31.10.2021
Author(s):	Denisa Ziu (ENG), Andrea Lazzeri (ENG), Vincenzo Croce (ENG)
Participant(s):	ENG, TUC, IMEC, DOMX, COM, CEL
Project:	Boosting DR through increased communIty-level consumer engaGement by combining Data-driven and blockcHain technology Tools with social science approaches and multi-value service design (BRIGHT)
Work Package:	WP2 – BRIGHT Technology and Novel Multi-Value Service Design
Task:	T2.2 – Functional Specification & Technology/Tools Design
Confidentiality:	Public
Version:	1.0

Table of Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

Table 1 List of Acronyms and Abbreviations

BRIGHT	Boosting DR through increased communIty-level consumer engaGement by combining Data-driven and blockcHain technology Tools with social science approaches and multi-value service design
DLT	Distributed Ledger Technology
DR	Demand-Response
GUI	Graphical User Interface
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
JIT	Just In Time
JTC	Joint Technical Committee
MAPE	Mean Absolute Percentage Error
ML	Machine Learning
MQTT	Message Queue Telemetry Transport
SGAM	Smart Grid Architecture Model
SRS	Software Requirements Specification
TDD	Test-Driven Development
VPP	Virtual Power Plant
WP	Work Package

Executive Summary

This document is part of the deliverables of the BRIGHT project, in the context of Work Package 2 (WP2) "BRIGHT Technology and Novel Multi-Value Service Design". Specifically, it presents the design of the DR technologies to be used in BRIGHT in terms of technical and functional specifications of the project's ICT components with an in-depth definition of the new tools and functionalities that will be developed in the technical WPs.

As a first step the methodology for the definition and the development of BRIGHT Software Requirements Specification (SRS) is presented. The requirements of the tools to be developed in BRIGHT have been built following the guidelines of the ISO/IEC/IEEE 29148:2018 standard. As only one version of this document exists, an agile approach, supported by online tools like Trello, will be taken into account for the refinement of the requirements. Furthermore, as the computational complexity of the BRIGHT techniques and algorithms can be non-trivial, special attention has been paid to the technologies that follow the principles of the Reactive Manifesto, namely being Responsive, Resilient, Elastic and Non-Blocking, to guarantee scalable performance.

Subsequently, the high-level software requirements for each technical WP are presented. For each group of components, an assets section describes the scope and overview of the set of tools developed in the WP as a whole, and a Tools section, which describes the functional and non-functional requirements, the technology stack, and the validation process for each of the specific tools that will be developed.

As a final step, the actors, the design of the overall architecture of the BRIGHT ecosystem, the interactions between the tools, and the technological aspects are presented. Particular attention is given to the interoperability layer, essential for a common way of communication between the tools. Scalability, flexibility, reliability, and high performance are the key characteristics taken into consideration for the technologies adopted in BRIGHT project.

1 Introduction

1.1 Purpose

This document (D2.3 *DR technologies and tools)* reports the results of the activities performed in Task 2.2 *Functional Specification & Technology/Tools Design*. The task aims to define the technical and functional specifications of BRIGHT technology and tools that will be developed in WP4, WP5, and WP6. To this purpose two templates have been sent to the partners for the collection of technical information. The information retrieved follows the construct, attributes, and characteristics structure defined by the IEEE guidelines (ISO/IEC/IEEE 29148:2018 standard). Another outcome of this deliverable is the design of BRIGHT's high-level architecture and how all its components are organized and interact with each other. Before providing the architecture model, the actors of the system and the datasets will be reported.

1.2 Relation to Other Activities

Task 2.2 provides functional specification and technology/tools design for the BRIGHT project. The output of deliverable D2.3 is the complete set of technologies/tools requirements specification of BRIGHT's ICT components. This output differs from that of T2.4, contained in D2.2, which instead focused on non-functional requirements, spefically those related to privacy, cybersecurity, ethics, and legal dimensions. Ultimately, outputs of T2.2 and T2.4 should be considered as complimentary.

The output of this deliverable is particularly relevant for WP4, WP5, WP6 that represent the BRIGHT tools and services development. The list of functional and non functional requirements is also important for WP7 for the validation of the defined BRIGHT's tools in the pilots.

1.3 Structure of the Document

Deliverable D2.3 is structured as follows:

- Chapter 1: Introduction to the objective of the document and its structure.
- Chapter 2: Methodology used for the definition of the BRIGHT software requirements specification. Agile approach model and reactive systems are the key concepts for the development of BRIGHT technology. Requirements such as functional requirements, performance requirements, interface requirements, design constraints, software system attributes, technology stack and validation process identified for each tool.
- Chapter 3: Description of the architecture model, actors, datasets, technology aspects for the realization of the BRIGHT system.
- Chapter 4: Document conclusions.

2 BRIGHT Software requirements specification

2.1 Methodology

2.1.1 IEEE guidelines for business and user requirements

The International Organisation for Standardization (ISO) and the International Electrotechnical Commission (IEC) work together in technical commitees for the development of international standards in fields of mutual interest, together with other international organisations and governamental bodies. The ISO/IEC Joint Technical Committee (JTC) 1 is a joint technical committee in the field of information technology.
The ISO/IEC/IEEE 29148:2018 is the second edition of the ISO/IEC/IEEE 29148:2011 standard, prepared by ISO/IEC JTC 1 in cooperation with the Systems and Software Engineering Standards Committee of the IEEE Computer Society.
The document specifies the required processes implemented in requirements engineering for systems and software through their lifecycle, provides guidelines for applying the related processes, and specifies the required information items and their contents for the implementation of the processes. The standard is meant for anyone performing requirements engineering activities related to software and hardware products and related services and can be applied regardless of the project scope, size, and complexity.

According to the standard, the primary result of the requirements engineering process is a set of requirements:

- With reference to a defined software
- Enabling understanding between stakeholders
- Validated against real-world needs
- Able to be implemented
- Providing a reference for verification.

The requirements definition begins with stakeholder needs. Initial stakeholder needs may lack definition, consistency, and feasibility so they are refined to obtain valid requirements and the same process is also applied recursively to lower levels of the system structure.

Well-formed requirements contribute to the process of their validation and ensure that stakeholders' needs are accurately captured. A well-formed requirement should:

- be met or possessed by a system to address a stakeholder concern
- be measurable
- be bounded by constraints
- define the performance of the system, but not a capability of the user
- be verifiable

If expressed in the form of a natural language, the requirement should include a subject and a verb, state the subject of the requirement, what shall be done, or a constraint on the system. Figure 1 shows some requirements examples.

[Condition] [Subject] [Action] [Object] [Constraint of Action]

When signal X is received, the system shall set the signal X received bit within 2 seconds

At sea state 1, the radar system shall detect targets at ranges out to 100 nautical miles

The invoice system shall display pending customer invoices in ascending order of invoice due date

Figure 1 Functional requirements examples (from ISO/IEC/IEEE 29148:2018)

The standard also suggests some best practices on the specific keywords and terms to be used, as follows:

- Requirements are mandatory binding provisions and use 'shall'
- Non-requirements use verbs such as 'are', 'is', and 'was'. Avoid the term 'must', due to potential misinterpretation
- 'Will' is used in non-mandatory, non-binding provisions or to establish context or limitations
- Desired, non-mandatory, non-binding provisions use 'should'
- Suggestions or allowances use 'may'
- Avoid negative requirements, such as 'shall not'
- Avoid using passive voice, such as 'it is required that'
- Avoid using terms such as 'shall be able to'

It is important to agree in advance on all terms specific to requirements and apply them consistently.

Interfaces to existing systems, pre-existing technologies, physical limitations, laws, budget, and additional limitations that restrict the design or the implementation of the solution are indicated as contraints.

Finally, requirements should be ranked to indicate priority.

In BRIGHT, the ISO/IEC/IEEE 29148:2018 standard has been used for the definition of the software requirements specification. Specifically, two templates have been prepared and sent to the technical partners for the collection of BRIGHT's tools requirements. Templates are related to each technical WP and consist of:

- Assets SRS Template: to define the scope and overview of the set of tools developed in the WP as a whole.
- Tools SRS Template: to define the functional and non functional requirements for each of the specific tools to be developed.

The templates specification can be found in the Annex of this document.

2.1.2 Agile Requirements Modeling

Agile software development describes an iterative approach in which work is delivered in small increments to be able to respond to changes quickly. Due to its iterative nature, Agile works best for innovative projects and when working in an industry that changes quickly and, for this reason, requirements, plans, and results are evaluated continuously allowing the teams to respond to early feedback instead of relying on extensive upfront planning.

A common practice is to identify early in the project some high-level system requirements to help defining the scope of the project, identify business goals, and develop a common vision. Such requirements will be refined periodically as the project progressess, benefiting from the increased level of knowledge achieved in later stages of the project and exploiting additional information not available at the beginning of the activity.

Agile Model Driven Development (Figure 2) includes an initial effort for requirements envisioning during the initial phase of the project, often called “Iteration 0”.

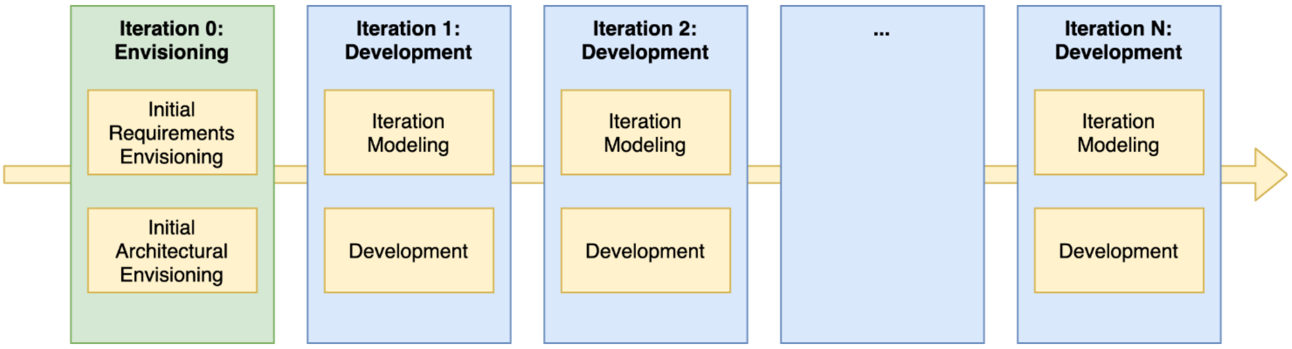
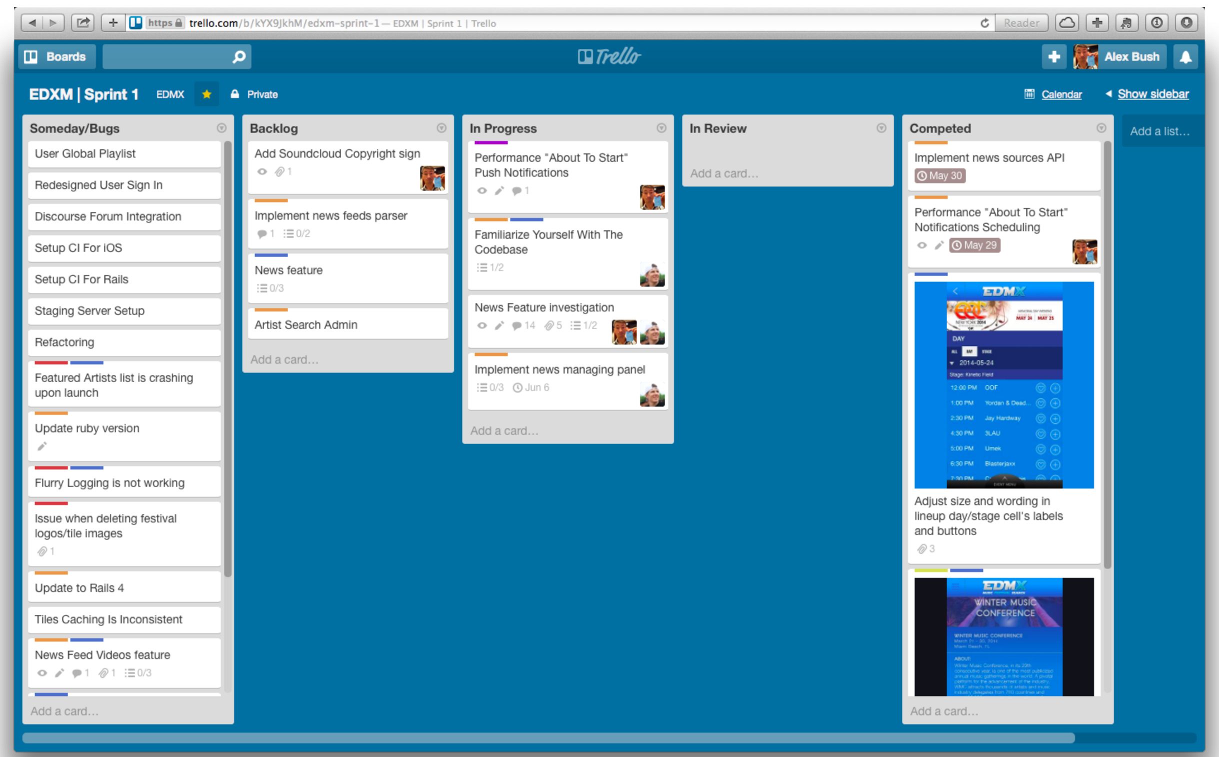


Figure 2 Agile Model Driven Development Lifecycle

The goal of this initial activity is not to create a detailed specification of requirements upfront like a traditional project but, instead, to identify the high-level scope, the initial requirements stack, and the architectural vision.

Actual modeling is part of each iteration effort and helps planning the work for the iteration. Each incremental steps focuses on specific issues following a Just In Time (JIT) approach and the active participation of stakeholders is encouraged. Modelling focuses only on what is needed at the time of each iteration, allowing requirements to evolve during the project lifecycle and, in some cases such as in Test Driven Development (TDD), capturing them in the form of operative specifications [1].

While agile encourages the usage of physical tools like boards and backlogs, a number of different online tools (Figure 3) allow to manage projects following agile methodologies even when the usage of physical tools is unpractical due to the team size or its distribution in different geographical areas.


trello.com/b/kYX9JkhM/edxm-sprint-1 — EDXM | Sprint 1 | Trello

Boards · Trello · Alex Bush

EDXM | Sprint 1 · EDMX · Private · Calendar · Show sidebar

Someday/Bugs
- User Global Playlist
- Redesigned User Sign In
- Discourse Forum Integration
- Setup CI For iOS
- Setup CI For Rails
- Staging Server Setup
- Refactoring
- Featured Artists list is crashing upon launch
- Update ruby version
- Flurry Logging is not working
- Issue when deleting festival logos/tile images
- Update to Rails 4
- Tiles Caching Is Inconsistent
- News Feed Videos feature
- Add a card…

Backlog
- Add Soundcloud Copyright sign
- Implement news feeds parser
- News feature
- Artist Search Admin
- Add a card…

In Progress
- Performance "About To Start" Push Notifications
- Familiarize Yourself With The Codebase
- News Feature investigation
- Implement news managing panel
- Add a card…

In Review
- Add a card…

Competed
- Implement news sources API · May 30
- Performance "About To Start" Notifications Scheduling · May 29
- Adjust size and wording in lineup day/stage cell's labels and buttons
- Add a card…

Add a list…

Figure 3 Example of Agile Board with Trello

Another advantage of using online tools is that a single product can be used to bundle the capabilities of different tools such as agile boards, backlogs, roadmaps and reports and can usually be integrated with existing services that help track and manage activities.

A very popular visual work management tool is Trello [2]. Trello is a collaborative work management app designed to track team projects, highlight tasks underway, show who they are assigned to, and detail progress towards completion. Trello relies on the principles of Kanban project boards to visualize workflows, providing managers and team members with a simple overview of a project from start to finish. Trello main components are boards, lists, and cards. A board represents a place to keep track of information — often for large projects, teams, or workflows. Within each board, several lists can be created to indicate the progress of a project; "to do," "in progress," and "done" lists are common examples. Individual cards within the lists hold information on a specific task and can be moved from list to list as needed (such as when a task is completed).
Trello will be used in BRIGHT to track the refinement of requirements. This document contains the first version of BRIGHT 's requirements. During the project, the requirements will be reviewed and changed as necessary following an agile approach.

2.1.3 Reactive Manifesto

In the last years, software application requirements have changed dramatically. The impact of cloud technology on IT increased demand for speed, flexibility, and innovation and the globalization of the market economy puts new pressures to increase speed and lower costs. Workforce automation is changing how businesses think about IT operations, monitoring, and management and new services built-on-the-web and app-enabled may be used to bring disruptive products on the market in a fraction of the time it would take a traditional system to respond [3].

In such a scenario, yesterday's software architectures are unable to met today's demands and, for this reason, the authors of the *Reactive Manifesto* [4] defined the concept of reactive systems, systems designed to be responsive, resilient, elastic, and message driven to be more failure tolerant, highly responsive, flexible, loosely-coupled, easy to develop and maintain, and scalable.
The architecture of a reactive system enables multiple microservices to work as a single unit, providing greater elesticity under changing workloads and resiliency in case of failure.

Following the above definition, reactive systems are:

- Responsive: able to respond rapidly and consistently.
- Resilient: remaining responsive in case of failure. This can be achieved thanks to replication, containment, isolation, and delegation.
- Elastic: remaining responsive under changing workload, thanks to live performance measures, scaling algorithms, and avoiding bottlenecks.
- Message Driven: relying on asynchronous messages queues. Non blocking communication reduces system overhead allowing recipients to consume resources only when active.

Implementing reactive systems using Apache Kafka
Apache Kafka can be considered the de-facto asynchronous messaging technology for reactive systems. Kafka is an open-source, distributed streaming platform to handle streams of events, able to work in a reactive and highly responsive manner.
While Kafka alone already offers built-in resiliency and scalability, it is required to configure the system appropriately and to consider how applications integrate with Kafka through producers and consumers for achieving full reactivity [5].
Reactive systems achieve loose-coupling and isolation establishing boundary between components thanks to non-blocking asynchronous messages. Kafka enables the asynchronous message-passing that makes up the backbone of a reactive system. By configuring Kafka specifically for resilience and elasticity, applications may be responsive to events and therefore reactive.
Kafka already uses a combination of multiple distributed brokers and replicates records between them. To achieve end-to-end resiliency of records, it is possible to configure acknowledgements, retry policies, and offset commit strategies.
Kafka already provides a certain degree of scalability, allowing brokers and partitions to be scaled out creating a system elastic enough to deal with fluctuating load. When developing elastic producer applications it is important to scale producers so that they don't produce duplicate messages.
It is also possible to scale consumer applications to collaborate using consumer groups: in this way, each consumer in the group receives a subset of the records on a specific topic.

In BRIGHT Kafka will be used for the implementation of the Message Queue system as part of the interoperability layer. BRIGHT's tools communicate with each other using Kafka as an intermediary. This is achievable thanks to Kafka's publish-subscribe approach for handling record writing and reading. The publish-subscribe model (pub-sub) is a communication strategy in which the sender sends events — whenever events are available — and each receiver chooses which events to receive asynchronously.

2.2 High-Level Requirements

2.2.1 Community and Customer Digital Twin Models (WP4)

2.2.1.1 Assets

Scope	
Name	Community and Customer Digital Twin Models
Description	The tools enable collection and process of the data from the pilot sites. The modelled data is subsequentialy used by Machine Learning (ML) techniques and algorithms in order to predict both production and consumption of energy by customers, communities, etc. The tools are the following: <u>Multi-purpose clustering tool for thermal loads and assets</u> The tool enables the automated clustering in household heating scenarios to identify consumers/buildings/heating systems characterized by similar behaviour/performance. <u>Dynamic simulator for modelling thermal loads and assets</u> The tool enables the automated performance simulation of thermal assets for household heating scenarios, by generating the digital twin representation of their core components (heating system, building, occupants). <u>Energy Forecasting Tool</u> The tool allows the users to: • visualize historical energy data on consumption or production by selecting a specific day in the past. • visualize the energy prediction results for specific prosumer or energy asset. <u>Physics-informed Modeling Framework</u> This tool focuses on control-oriented modeling of building thermal dynamics. The tool will allow users to: • Model the thermal behavior of a building • Predict future states of this building • Perform what-if analysis for this building.
Goals	The tools aim to achieve the WP objectives by developing models for data, implementing Digital Twins, developing techniques and forecasting algorithms able to predict energy consumption and usage. <u>Multi-purpose clustering tool for thermal loads and assets</u>

	The tool delivers as output the calculated clusters, which are used as input by algorithms/services executed at subsequent stages towards improving their performance. It is particularly useful for improving the performance of generic algorithms (e.g. demand forecasting) that cannot be optimized to model the performance of each unique asset (e.g. specific boiler/user), but can be calibrated for assets featuring similar characteristics (e.g. boilers of the same technology/consumers with similar heating habits). <u>Dynamic simulator for modelling thermal loads and assets</u> The tool generates synthetic time-series that reflect the real-world behaviour of the considered set of thermal assets (heating system, building, occupants), under different configurations. It is particularly useful for comparing the performance of the considered set of thermal assets (heating system, building, occupants), based on real historical data versus the digital twin model when employing different configurations, towards quantifying the potential improvement when modifying the set of assets, for instance by replacing the heating system with one of improved characteristics. <u>Energy Forecasting Tool</u> The tool predicts the energy demand, generation and flexibility of prosumers and energy assets. The tool runs on demand considering historical energy data as well as near real time monitored data. <u>Physics-informed Modeling Framework</u> The tool should periodically learn the behavior of the building using available/new data and clusters, returning a trained model of the building

Product overview	
Product perspective	<u>Multi-purpose clustering tool for thermal loads and assets</u> The tool runs on the server side and is executed on demand, taking as input historical data collected from: • heating system data (water temperature, etc.) and energy consumption data as captured from the attached domX heating controller • user heating preferences as captured by the DomX smartphone application and the room thermostat • indoor and outdoor environmental data (temperature, humidity, etc.) as captured by various connected sensors and web services

	The tool periodically analyzes the collected data through ML multi-dimensional clustering algorithms to deliver as output clusters of assets with similar characteristics in terms of: • user behavior • demand patterns • offered flexibility • building performance • heating system performance The clustering results are stored in a database and are accessible for integration with other components/services through a web API. The output is also offered in human readable reports in html and pdf formats. <u>Dynamic simulator for modelling thermal loads and assets</u> The tool runs on demand, taking as input historical data collected from: • heating system data (water temperature, etc.) and energy consumption data as captured from the attached domX heating controller • user heating preferences as captured by the domX smartphone application and the room thermostat • indoor and outdoor environmental data (temperature, humidity, etc.) as captured by various connected sensors and web services The tool analyzes the collected time-series to output a digital twin model of the given set of thermal assets (heating system, building, occupants), by generating synthetic time-series that reflect the real-world behaviour under different configurations. The tool employs: • mathematical models to simulate linear behaviors • neural networks to learn the non-linear behaviour out of energy (e.g. boiler consumption) and non-energy data (boiler operating temperature). The generated synthetic time-series are stored in a time-series database and are accessible for integration with other components through a web API. The output can also be integrated with a custom dashboard for visualizing the performance difference of the different monitored parameters (e.g. indoor temperature, boiler consumption) across time, between the actual and synthetic

	datasets. <u>Energy Forecasting Tool</u> • The tool runs on the server side while the tool front end will be shown on client side in a browser. • The energy forecasting tool Graphical User Interface (GUI) provides menus, toolbars, buttons, panes, containers, grids allowing for easy control by a mouse. • The tool considers the historical energy data as well as of the monitored energy data that were previously saved in a database. • The tool stores the prediction results in a database which will provide an API for integration with other components. • The tool considers both energy features as well as contextual features in the prediction process. <u>Physics-informed Modeling Framework</u> The application works on a local machine as python script files. The inputs can be given directly in the script file or provided in the form of JSON input files. Training data needs to be provided as .csv files.
Product functions	<u>Multi-purpose clustering tool for thermal loads and assets</u> The tool analyzes household heating data collected from various data sources (heating controller, smartphone application, climate sensors) and delivers automated clustering results identifying consumers/buildings/heating systems characterized by similar behaviour/performance. <u>Dynamic simulator for modelling thermal loads and assets</u> The tool analyzes household thermal data collected from various data sources (heating controller, smartphone application, climate sensors) and outputs synthetic time-series that reflect the real-world behaviour of the considered set of thermal assets (heating system, building, occupants), under different configurations. <u>Energy Forecasting Tool</u> The tool will predict the energy demand / generation or flexibility of prosumers and energy assets. <u>Physics-informed Modeling Framework</u> The tool will allow users to: • Model the thermal behavior of a building • Predict future states of this building

	• Perform what-if analyses for this building
User characteristics	Multi-purpose clustering tool for thermal loads and assets The tool is intended for integration with other components/services through a web API and field experts for studying the human readable exported reports. Dynamic simulator for modelling thermal loads and assets The tool is intended for integration with other other components/services through a web API and field experts for simulating/visualizing the performance of thermal assets under different configurations. Energy Forecasting Tool The software is intended for general users with no previous experience and no specific technical expertise required. Physics-informed Modeling Framework The software is intended for general users with little previous experience. Technical expertise is required to analyse the trained models.

2.2.1.2 Tools

2.2.1.2.1 Multi-purpose clustering framework for thermal loads and assets (TLC)

Functional requirements

Functional Requirements		
ID	**Requirement**	**Related Use Case**
TLC-1	The tool should consider energy data stored in a database and should store the clustering results in a database	LLUC4_1 LLUC4_2 LLUC4_3
TLC-2	The tool shall be able to analyze thermal data collected from various data sources (heating controller, smartphone application, climate sensors)	
TLC-3	The tool shall allow integration with other components/services through a REST API	
TLC-4	The tool shall provide clustering results to be exported in human readable reports (html/pdf)	
TLC-5	The tool shall provide clustering results identifying heating systems/buildings/users characterized by similar behaviour/performance	

Performance requirements

Performance Requirements	
ID	**Requirement**
TLC-6	The tool should generate a clustering report whenever new data is added (i.e., new household). As an example, data collected for 100 households with historical information of less than a year, should be clustered in less than 30 min, excluding the time needed for data fetching.

Interface requirements

Interface Requirements	
ID	**Requirement**
TLC-4	The tool shall provide clustering results to be exported in human readable reports (html/pdf).
TLC-2	The tool shall be able to analyze thermal data collected from various data sources (time-series DBs, relational DBs, message brokers).
TLC-3	The tool shall allow integration with other components/services through a REST API.

Design constraints
No such constraints have been identified at this point

Software system attributes

Software System Attributes		
ID	**Requirement**	**Software Attribute**
TLC-10	Each clustering output should produce clusters with cohesion, characterized by a positive Silhouette score at minimum. Optimally, over 50% of clustering outputs should have a Silhouette coefficient over 0.2.	Reliability
TLC-11	The tool should ensure data privacy, so that no personal data are able to be collected through the REST API exposing data from the DBs	Privacy
TLC-12	The tool should be only accessible by certified users that have been granted a valid JWT token.	Security

Technology stack

A custom web-based graphical user interface will be developed by combining the Vue.js and Grafana frameworks for visualizing the outputs of the thermal load clustering tool. Real-time data collected from the heating controllers are integrated using the MQTT message broker and stored in the Influx time-series DB. Pandas, SciKit Learn and TensorFlow will be used to handle the data and train the clustering models. The clustering results are stored in MySQL DB exposed over a REST API while at the same time employing JWT tokens for guaranteeing secure data exchange with other services.

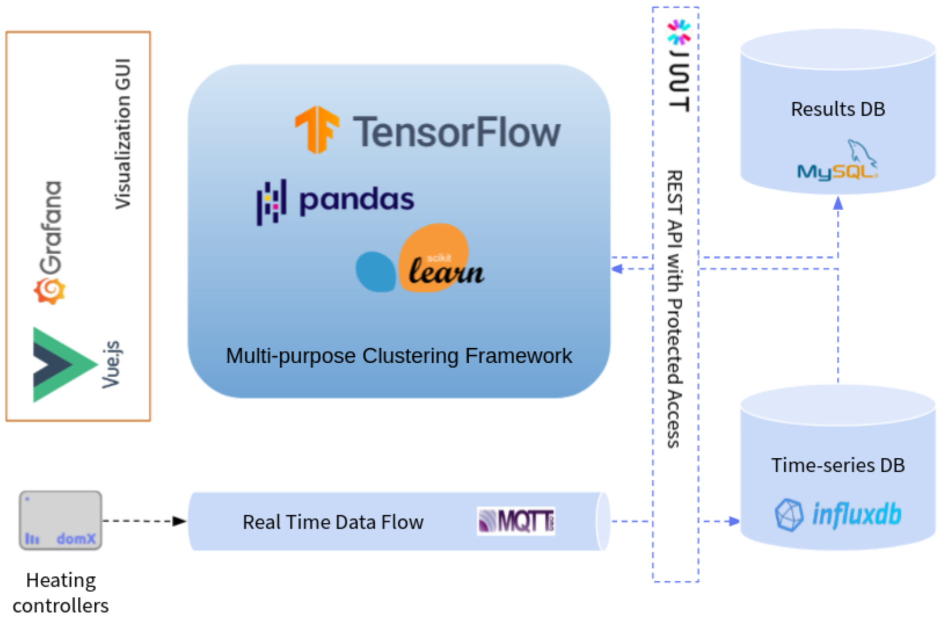


Figure 4 Multi-purpose clustering framework for thermal loads and assets (TLC) Technology Stack

Verification

Verification of the attributes will be performed through unit tests and integration tests.

2.2.1.2.2 Dynamic simulator for modelling thermal loads and assets (SLC)

Functional requirements

Functional Requirements		
ID	**Requirement**	**Related Use Case**
SLC-1	The tool should consider energy data stored in a database and should store the simulation results in a database	LLUC4_1 LLUC4_2 LLUC4_3
SLC-2	The tool shall be able to analyze thermal	

	data collected from various data sources (heating controller, smartphone application, climate sensors)	
SLC-3	The tool shall generate synthetic time-series results that reflect the real-world behaviour under different configurations	
SLC-4	The tool shall allow integration with other components/services through a REST API	
SLC-5	The tool shall simulate thermal assets for household heating scenarios, by generating the digital twin representation of their core components (heating system, building, occupants).	

Performance requirements

Performance Requirements	
ID	**Requirement**
SLC-6	The tool should generate the simulation results for one household and by using the forecasting horizon of 15 minutes in less than 30s.

Interface requirements

Interface Requirements	
ID	**Requirement**
SLC-2	The tool shall be able to analyze thermal data collected from various data sources (time-series DBs, relational DBs, message brokers).
SLC-4	The tool shall allow integration with other components/services through a REST API.
SLC-9	The tool should be able to visualize the performance difference of the different monitored parameters (e.g. indoor temperature, boiler consumption) across time, between the actual and simulated datasets.

Design constraints
No such constraints have been identified at this point

Software system attributes

Software System Attributes		
ID	**Requirement**	**Software Attribute**
SLC-10	The tool should forecast the desired metrics with a Mean Absolute Percentage Error (MAPE) below 10 % for a forecasting horizon of 15 min.	Reliability

SLC-11	The tool should ensure data privacy, so that no personal data are able to be collected through the REST API exposing data from the DBs	Privacy
SLC-12	The tool should be only accessible by certified users	Security

Technology stack

A custom web-based graphical user interface will be developed by combining the Vue.js and Grafana frameworks for visualizing the outputs of the thermal load simulation tool. Real-time data collected from the heating controllers are integrated using the MQTT message broker and stored in the Influx time-series DB. Pandas, Statsmodels and TensorFlow will be used to handle the data and train the forecasting models. The simulation results are stored in the Influx time-series DB and MySQL DB, exposed over a REST API while at the same time employing JWT tokens for guaranteeing secure data exchange with other services.

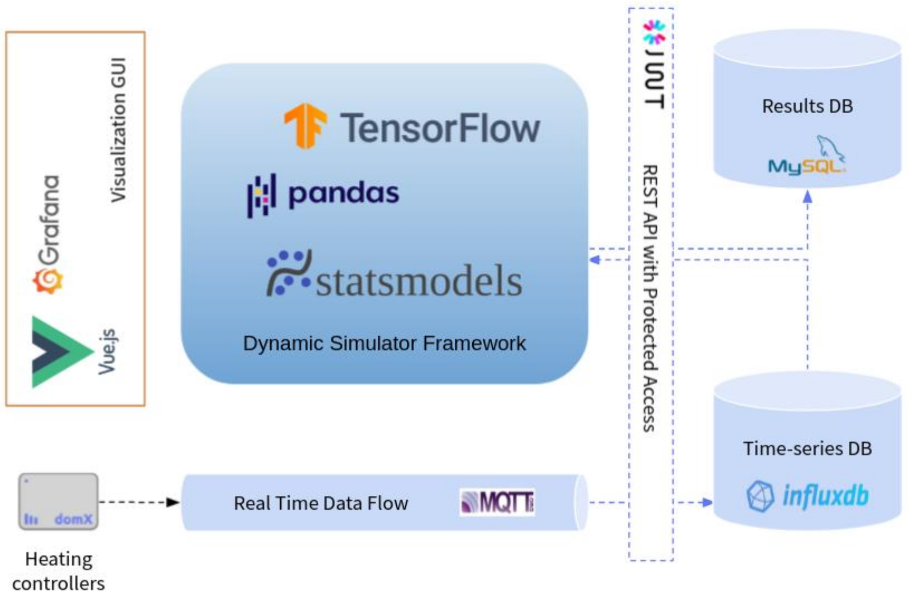


Figure 5 Dynamic simulator for modelling thermal loads and assets (SLC) Technology Stack

Verification

Verification of the attributes will be performed through unit tests and integration tests.

2.2.1.2.3 Energy Forecasting Tool (EFT)

Functional requirements

Functional Requirements		
ID	**Requirement**	**Related Use Case**
EFT-1	The tool shall allow users to login.	LLUC1_1 LLUC1_2 LLUC2_1 LLUC2_2 LLUC2_3 LLUC3_2 LLUC3_3 LLUC4_1 LLUC4_2 LLUC4_3
EFT-2	The tool shall allow users to visualize their energy consumption or production	
EFT-3	The tool shall allow users to visualize their forecasted energy consumption / production or flexibility	
EFT-4	The tool should consider energy data stored in a database and should save the prediction results in the database	
EFT-5	The tool shall forecast the energy consumption, production, and flexibility	

Performance requirements

Performance Requirements	
ID	**Requirement**
EFT-6	The tool should display the forecasting results within 60s after the prediction process is initiated.

Interface requirements

Interface Requirements	
ID	**Requirement**
EFT-7	The tool should provide a web interfaces for users featuring menus, toolbars, buttons, panes, containers, grids allowing for easy control by a keyboard and a mouse.
EFT-8	The tool should provide web inferences to visualize the historical energy data as well as the energy predictions of demand, production, and flexibility.
EFT-9	The tool should store the energy prediction results in a database allowing other systems/components to get data using rest APIs.
EFT-10	The tool should get the energy data used in forecasting from a database using rest APIs.

Design constraints
No such constrains have been identified at this point

Software system attributes

Software System Attributes		
ID	**Requirement**	**Software Attribute**
EFT-11	The tool energy prediction error should be below 15%.	Reliability
EFT-12	The tool shall allow users the users to see only their data after login	Security

Technology stack

React JS is used to provide a web-based graphical user interface for the forecasting tool. MySQL will be the database of the tool and a REST API is used to get / store the energy data and prediction results. SciKit Learn, KERAS and TensorFlow will be used to define and train the energy prediction models. The smart meters data will be integrated using queuing systems such as RabbitMQ.

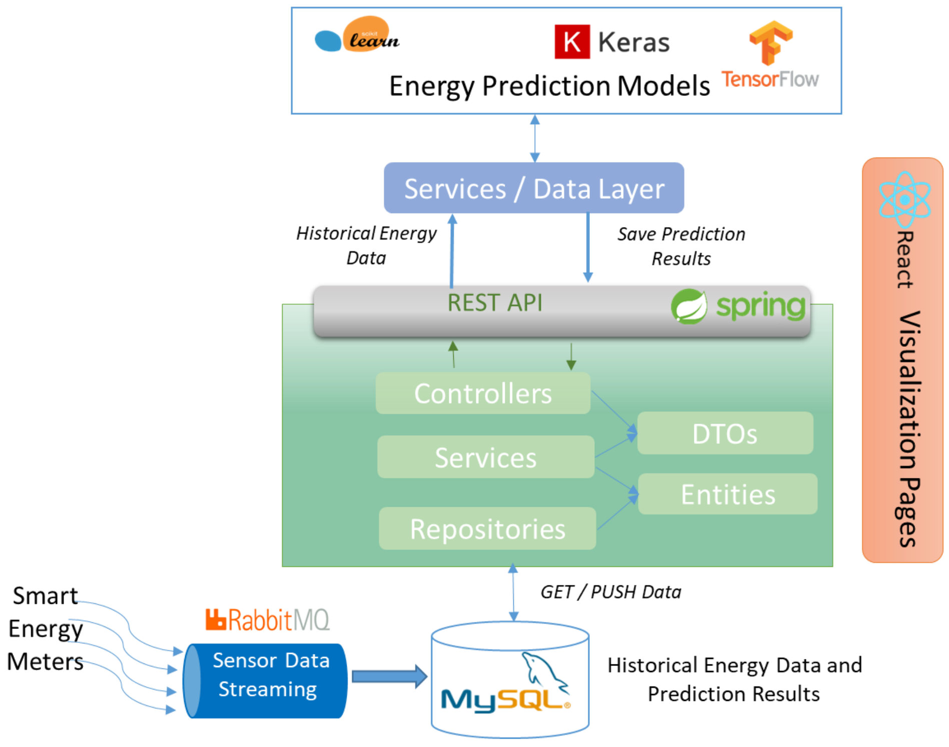


Figure 6 Energy Forecasting Tool Technology Stack

Verification

Verification of the attributes will be performed through unit tests and integration tests.

2.2.1.2.4 Physics Informed Modeling Framework

Functional requirements

Functional Requirements		
ID	**Requirement**	**Related Use Case**
PIMF-1	The tool shall allow users to model the thermal behavior of a single household.	LLUC1_1 LLUC2_1 LLUC3_1 LLUC4_1 LLUC4_2 LLUC4_3
PIMF-2	After training, the model generated by this tool can be used for predicting future states and perform what-if analyses	

Performance requirements

Performance Requirements	
ID	**Requirement**
PIMF-3	The tool should give predictions within a range of ±1.5°C.

Interface requirements
Not decided yet.

Design constraints
Not decided yet.

Software system attributes

Software System Attributes		
ID	**Requirement**	**Software Attribute**
PIMF-4	The tool performance will be benchmarked with validation data and only used if this condition is satisfied.	Reliability
PIMF-5	The tool will have built-in warm start models for aiding training. Each training will be followed by a model checkpoint.	Availability

Technology stack

This tool uses Python 3.7 with standard data science and machine learning packages: numpy, pandas, pytorch, along the standard python packages. The training data needs to be loaded in the form of .csv files or dataframes. The trained models are saved using built-in pytorch functions and as serialized objects based on Python's pickle module.

Verification

Verification of the attributes will be performed through unit tests and integration tests. Further verification would include model testing on an in-house simulator.

2.2.2 Digital-Twin enabled Flexibility and information valorisation (WP5)

2.2.2.1 Assets

Scope	
Name	Digital-Twin enabled Flexibility and information valorisation
Description	The tools produced will leverage on WP4 digital models in combination with ML techniques and AI algorightms in order to offer new flexibility services. <u>Reinforcement Learning based Control Framework</u> This tool focuses on developing optimal control policies using reinforcement learning based algorithms, leveraging the models developed in WP4. The tool will allow users to: • Generate control policies that optimize assets to the desired objective / cost function. • Perform comparison of different control policies for a specific system/ building/ asset group. <u>Integrated energy and non-energy smart home service manager</u> This tool is enabled through the integration of different smart home sensors able to characterize both energy (electricity consumption, boiler modulation) and non-energy parameters (climate comfort, indoor air quality, etc.). The integrated service manager enables home users to specify their personal preferences (room target temperature and humidity) and goals (e.g. prioritize comfort or economy). <u>Cross Sector Services Flexibility Optimization</u> The tool will allow the combination of flexibility services from different sectors to increase the amount of flexibility mobilized at the energy community level. The tool will allow the users to: • visualize the levels of energy flexibility of each individual service. • visualize the target / goal of the flexibility combination

	optimization. • visualize the optimal combination of services or control variables to meet the goal.
Goals	The tools use algorithms and techniques implemented in the previous WP in order to offer many different services. For instance, ML based algorithms and AI techniques are used for learning from the current data to produce control policies and flexibility services. <u>Reinforcement Learning based Control Framework</u> This tool learns from available data, leveraging models created in WP4, to compute optimum control policies that can be used for demand response applications. <u>Integrated energy and non-energy smart home service manager</u> The main goals of the tool are to: • offer smart management of IoT-enabled home appliances, devices and sensors • improve energy management by considering both energy and non-energy parameters • improve the end user comfort and living by adapting the environment to the preferences of home residents. <u>Cross Sector Services Flexibility Optimization</u> The tool should work offline and should decide on the optimal combination of flexibility services to meet the goal. The combinations should be stored in a database.

Product overview	
Product perspective	<u>Reinforcement Learning based Control Framework</u> The framework includes two components: • An application that works on a local machine as python script files. The inputs can be given directly in the script file or provided in the form of JSON input files. Training data needs to be provided as .csv file • An application that will be deployed on managed cloud services and integrated in the aggregator's backend for executing the field demonstrations. <u>Integrated energy and non-energy smart home service manager</u> The tool is delivered as a service executed locally on the DomX home-IoT gateway, which acts as the central component collecting measurements from home devices able to characterize both energy and non-energy parameters and controlling the operation of multiple appliances.

	The tool considers as input multiple energy parameters through the following devices: • smart electricity meter (aggregate home electricity power-energy monitor) • smart electricity plug (device electricity power-energy) • smart heating controller (instant boiler modulation, energy consumption for space heating, DHW preparation) The tool considers as input multiple **non-energy parameters** through the following devices: • climate sensors able to characterize user comfort (temperature, humidity, light) • security sensors (door contacts, human presence) • indoor air quality sensors (PM2.5, CO2, VOC, etc.) The tool can **control the operation of multiple device types** through the following actuators: • smart relay (phase level device control ON/OFF) • smart electricity plug (plug level device control ON/OFF) • smart heating controller (modulated boiler control, target temperature control both for space heating and DHW preparation) Indicative scenarios that can be offered through the integrated service manager include the: • configuration of personal climate comfort thresholds (temperature 23°C, humidity 50%) and consumption profile (economy)) that continuously adapt the operation of connected appliances belonging to different vectors (gas boiler for temperature control and dehumidifier for humidity control). • delivery of advice to end users when the indoor air quality has fallen below certain thresholds and manual action needs to be taken (e.g., natural ventilation). <u>Cross Sector Services Flexibility Optimization</u> • The tool is server side and has as clients the latest version of the Chrome browser. • The tool GUI provides menus, toolbars, buttons, panes, containers, grids allowing for easy control by a keyboard and a mouse. • The tool connects to a database for taking flexibility data and stores the results to be used by 3rd party applications.

Product functions	Reinforcement Learning based Control Framework The tools will allow users to: • Generate control policies that optimize assets to the desired objective • Perform comparison of different control policies for a specific system/ building/ asset group. Integrated energy and non-energy smart home service manager The tool provides the following functions: • improve energy management by considering both energy and non-energy parameters • configure user preferences and consumption profiles • improve the end user comfort and living by adapting the environment to the preferences of home residents Cross Sector Services Flexibility Optimization The tool allows its users to visualize the levels of energy flexibility of each individual service, the target / goal of the flexibility combination optimization and the optimal combination of services or control variables to meet the goal. The tool decides on the optimal combination of flexibility services to meet the goal.
User characteristics	Reinforcement Learning based Control Framework The software is intended for general users with little previous coding experience. Technical expertise is required to analyse the computed policies. Integrated energy and non-energy smart home service manager The software is intended for general users with no previous experience and no specific technical expertise required. Cross Sector Services Flexibility Optimization The tool is intended for users with experience in the management of smart grid and flexibility services.

2.2.2.2 Tools

2.2.2.2.1 Reinforcement Learning based Control Framework

Functional requirements

Functional Requirements		
ID	**Requirement**	**Related Use Case**
RLCF-1	The tool shall allow users to generate control policies that	LLUC1_2 LLUC2_1

	optimize according to the pre-defined objective and which is adaptable to different use cases / cost functions.	

Performance requirements

Performance Requirements	
ID	**Requirement**
RLCF-2	The tool should generate control policies better than business-as-usual policies.
RLCF-3	Avoiding violations of comfort constraints

Interface requirements

Interface Requirements	
ID	**Requirement**
RLCF-4	Device-cloud integration (proprietary interface)
RLCF-5	Use of Energy market and meteo data via proprietary interface

Design constraints

- Satisfy comfort constraints

Software system attributes

Software System Attributes		
ID	**Requirement**	**Software Attribute**
RLCF-7	The tool performance will be benchmarked with validation data and compared with other control algorithms.	Reliability
RLCF-8	The tool will have built-in warm start models for aiding the training phase. Each training phase will be followed by a model checkpoint.	Availability

Technology stack

This tool uses Python 3 with standard data science and machine learning packages: numpy, pandas, pytorch, along the standard python packages. The training data needs to be loaded in the form of .csv files, dataframes or via a proprietary data pipeline in the Flexpond platform. The trained models are saved using built-in pytorch functions and as serialized objects based on Python's pickle module.

Verification

Verification of the attributes will be performed through unit tests and integration tests. Further verification would include model testing via an in-house simulator and lab testing where appropriate.

2.2.2.2.2 Integrated energy and non-energy Smart home service Manager (ISM)

Functional Requirements

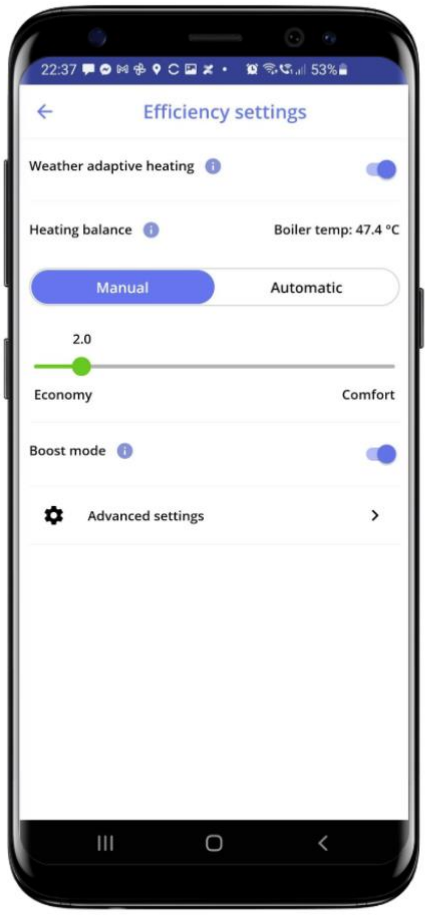
Functional Requirements		
ID	**Requirement**	**Related Use Case**
ISM-1	The tool should support different smart home sensors able to characterize energy (electricity consumption, boiler modulation) parameters and non-energy parameters (climate comfort, indoor air quality, etc.)	LLUC4_1 LLUC4_2 LLUC4_3
ISM-2	The tool should support different smart home sensors able to characterize non-energy parameters (climate comfort, indoor air quality, etc.)	
ISM-3	The tool should enable home users to specify their personal preferences (room target temperature and humidity) and goals (e.g. prioritize comfort or economy).	
ISM-4	The tool shall enable the translation of user preferences into commands able to control the operation of multiple appliances towards adapting the home environment to the user needs.	

Performance requirements

Performance Requirements	
ID	**Requirement**
ISM-5	The tool should enable the analysis of energy and non-energy parameters to be periodically executed at least every 60s and be directly translated into

	control commands, with acceptable delays lower than 1s.

Interface requirements

Interface Requirements	
ID	**Requirement**
ISM-6	The tool shall provide home users the ability to specify their personalized preferences and goals through an easy to use smartphone application screen.
ISM-7	The tool shall provide home users the ability to control the operation of multiple home devices through a transparent set of rules that do not vary based on the type of controlled devices (gas boiler for temperature control and dehumidifier for humidity control).  Efficiency settings; Weather adaptive heating; Heating balance; Boiler temp: 47.4 °C; Manual; Automatic; 2.0; Economy; Comfort; Boost mode; Advanced settings *Figure 7 Smartphone application screen for setting personalized preferences*

Design constraints
No such constraints have been identified at this point

Software system attributes

Software System Attributes		
ID	**Requirement**	**Software Attribute**
ISM-8	The tool should ensure data privacy, so that no personal data can be exposed without the user's consent	Privacy
ISM-9	The tool should enable the integration of new device types/sensor measurements with ease.	Maintainability

Technology stack

The Integrated energy and non-energy Smart home service Manager is executed locally as a local agent on the Interoperable home-IoT gateway that is based on the Raspbery Pi device, directly supporting the Wi-Fi, Bluetooth and ZigBee protocols. The user can specify personalized goals that directly activate the rule engine of Openhab and can automatically adapt the operation of multiple device types, such as smart relay plugs, heating controllers, etc.

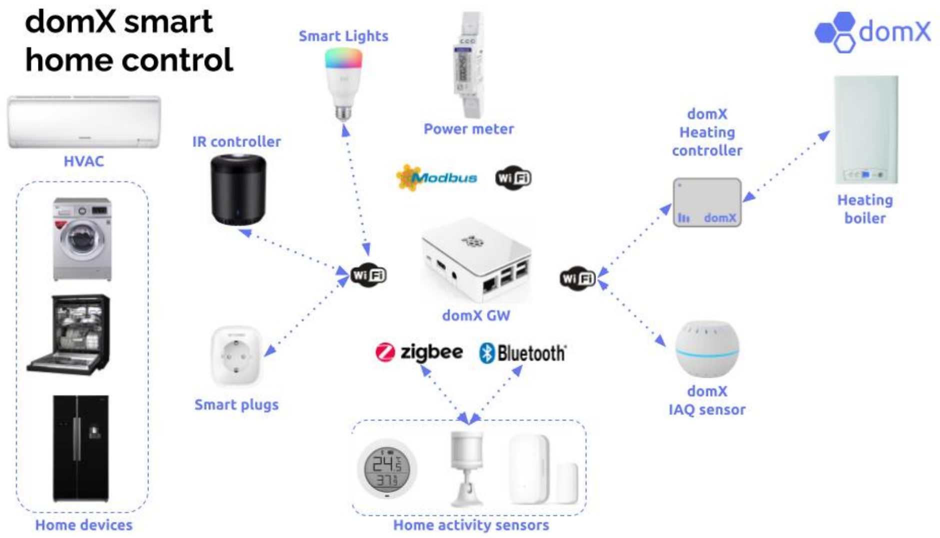


Figure 8 Device types considered by the Integrated energy and non-energy Smart home service Manager (ISM)

Verification

Verification of the attributes will be performed through unit tests and integration tests.

2.2.2.2.3 Cross Sector Services Flexibility Optimization Tool (CSSFO)

Functional requirements

Functional Requirements		
ID	**Requirement**	**Related Use Case**
CSSFO-1	The tool shall allow users to visualize the levels of energy flexibility of each individual service	LLUC1_1 LLUC1_2 LLUC2_1 LLUC3_1 LLUC4_1 LLUC4_2 LLUC4_3
CSSFO-2	The tool shall allow users to visualize the target / goal of the flexibility combination optimization	
CSSFO-3	The tool shall allow users to visualize the optimal combination of services or control variables to meet the goal	
CSSFO-4	The tool shall decide on the optimal combination of flexibility services to meet the goal	
CSSFO-5	The tool should get flexibility data and target goal form a database	
CSSFO-6	The tool should store the optimal combination result in a database	

Performance requirements

Performance Requirements	
ID	**Requirement**
CSSFO-7	The tool should take an optimization decision and display the results in less than 5 minutes.

Interface requirements

Performance Requirements	
ID	**Requirement**
CSSFO-8	The tool should provide a web interface for users featuring menus, toolbars, buttons, panes, containers, grids allowing for easy control by a keyboard and a mouse.
CSSFO-9	The tool should provide web inferences to visualize the levels of energy flexibility of each individual service, visualize the target / goal of the flexibility combination optimization and to visualize the optimal combination of services or control variables to meet the goal.
CSSFO-10	The tool should store the flexibility combination optimization results in a database allowing other systems to get data using rest APIs
CSSFO-11	The tool should get the cross-sector flexibility services from a database using rest APIs.

Design constraints

No such constrains have been identified at this point

Software system attributes

Software System Attributes		
ID	**Requirement**	**Software Attribute**
CSSFO-12	The tool should have high usability allowing the users to complete the task with a minimum number of steps	Usability

Technology stack

React JS is used to provide a web-based graphical user interface for the tool. MySQL will be the database of the tool and REST API is used to get / store the data. A custom solver based on Java is used to implement and run the heuristic for deciding on the optimal combination of flexibility services to meet the goal.

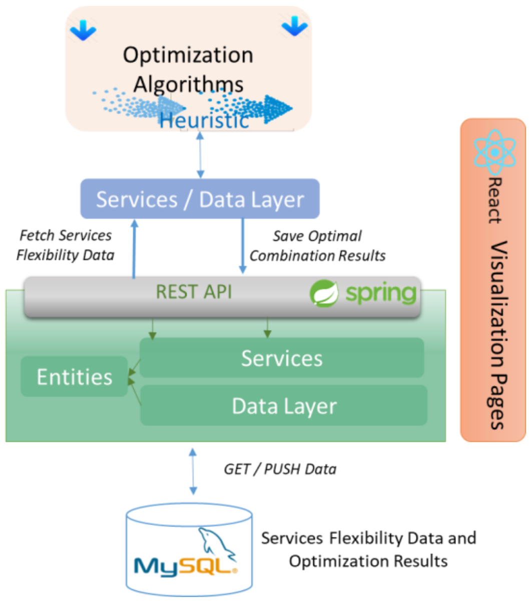

Optimization Algorithms
Heuristic
Services / Data Layer
Fetch Services Flexibility Data
Save Optimal Combination Results
REST API
spring
Entities
Services
Data Layer
React Visualization Pages
GET / PUSH Data
MySQL
Services Flexibility Data and Optimization Results

Figure 9 Cross Sector Services Flexibility Optimization Tool Technology Stack

Verification

Verification of the attributes will be performed through unit tests and integration tests.

2.2.3 DLT Enablers for Decentralized VPP (WP6)

2.2.3.1 Assets

Scope

Name	DLT Enablers for Decentralized VPP
Description	The tools created in the WP are used to integrate and uniform heterogeonous data and manipulate it. This data is used to enable a flexibility marketplace and flexbility tools. Specifically, the products are the following: <u>Interoperable Home automation Gateway</u> The home-IoT Gateway of domX provides for monitoring and control of the home environment, integrating various home sensors/controllers/appliances belonging to different vendor ecosystems and making them interoperable. <u>Decentralized Management of Energy Communities Product</u> The product provides 4 integrated software tools for the management of energy communities: • Community level Blockchain based Flexibility Marketplace: application allowing the users from a community to trade their energy flexibility in a P2P manner. Different types of flexibilities are considered such as electrical, thermal, comfort services, etc. • Blockchain Management and Settlement of Flexibility driven DR: tool which allows the injection of energy goals in smart contracts, the tracking of flexibility delivery and energy and financial settlement • Community Self-governance to Deliver Flexibility Services Tool: application which allows the creation and decentralized management of coalitions in a community to deliver flexibility services on demand to the main grid • Edge Metering Infrastructure and Interoperable Gateway: tool which allows monitoring of energy data and integration with smart contracts for flexibility actions automation.
Goals	<u>Interoperable Home automation Gateway</u> The main goals of the interoperable domX home automation gateway are to: • Support cost-effective interoperable smart home automation across vendor ecosystems • Support the dominant wireless protocols of the smart home domain • Offer smart management of IoT-enabled home appliances, devices and sensors. <u>Decentralized Management of Energy Communities Product</u> • Community level energy balance, unlocking the flexibility potential to meet local sustainability goals via P2P trading of local energy flexibility. • Enable communities to provide flexibility services to the

	main grid via decentralized management and self-governance.

Product overview	
Product perspective	<u>Interoperable Home automation Gateway</u> The domX home-IoT gateway builds on the Raspberry Pi device and runs the OpenHAB open home automation protocol, supporting the integration of a multitude of devices and technologies into: • an overarching gateway device • a uniform user interface and • a common approach to automation rules across the entire system, regardless of the number of manufacturers and sub-systems involved. The domX gateway supports the dominant wireless protocols of the smart home domain: • Wi-Fi • Bluetooth • ZigBee The gateway supports a wide variety of sensors, meters, controllers and home appliances, including: • smart electricity meters • smart plugs to control white goods • heating controllers to control legacy heaters and boilers • IR emulators to control legacy ACs • remotely controlled relays to manage the operation of heavy consuming appliances, such as electric water heaters/space heaters • climate sensors able to characterize indoor conditions (temperature, humidity, light) • indoor air quality sensors (PM2.5, CO2, VOC, etc.) • security sensors (door contacts, human presence) <u>Decentralized Management of Energy Communities Product</u> • The product runs on the blockchain using smart contracts while the tool front end will be shown on client side in a browser. • The product GUI provides menus, toolbars, buttons, panes, containers, grids allowing for easy control by a mouse. • The product considers the prediction of energy demand, generation and/or flexibility. • The matching between the flexibility bids and offers is done

	using services integrated with the Blockchain via Oracles providing both cooperative and competitive (price-driven) trading models for matching. • The optimal coalitions for community level energy flexibility services delivery are provided using services with the Blockchain via Oracles. • The product stores the energy flexibility transactions on the blockchain and injects the agreed levels of flexibility into the smart contracts. • The product considers the edge-monitored energy data for tracking the energy flexibility actual delivery according to the transactions' values • The product provides flexibility management actions to meet the target goals to be executed via an interoperable gateway for automation • The product deals with the energy and financial settlement of energy flexibility transactions.
Product functions	<u>Interoperable Home automation Gateway</u> The domX home-IoT gateway provides the following functions: • smart management of home appliances, devices and sensors • control of all home assets through a uniform user interface • setup of automation rules through a common approach <u>Decentralized Management of Energy Communities Product</u> • The product shall allow users to register their flexibility assets and flexibility management preferences (e.g., via the flexibility marketplace or self-governance of coalitions) • The product shall allow users to login and visualize their energy predictions, market session information and to submit flexibility bids and offers. • The product will match and pair the flexibility bids and offers and will generate and store flexibility transactions on the blockchain. • The product shall allow the construction of community levels coalitions and their self-governance to meet the constraints and deliver on demand flexibility services to the main grid. • The product shall track the energy flexibility delivery of users according to the transactions values and monitored data. • The product shall assess deviations from traded values and shall do the energy and financial settlement of flexibility transactions.
User characteristics	<u>Interoperable Home automation Gateway</u>

	The software is intended for general users with no previous experience and no specific technical expertise required. Decentralized Management of Energy Communities Product The software is intended for small scale prosumers or aggregators belonging to an energy community, or for local distribution system operators.

2.2.3.2 Tools

2.2.3.2.1 Community level Blockchain based Flexibility Marketplace (CBFM)

Functional requirements

Functional Requirements		
ID	**Requirement**	**Related Use Case**
CBFM-1	The application shall allow users to register to participate in the trading sessions.	LLUC1_1 LLUC1_2 LLUC2_1 LLUC3_1 LLUC4_1 LLUC4_2 LLUC4_3
CBFM-2	The application shall allow users to login.	
CBFM-3	The application shall allow users to visualize their forecast of energy consumption, production, or flexibility	
CBFM-4	The application shall allow users to visualize market session information	
CBFM-5	The application shall allow users to place bids for buying energy flexibility	
CBFM-6	The application shall allow users to place offers for selling energy flexibility	
CBFM-7	The application shall allow the matching of flexibility bids and offers and generation of energy transactions on the blockchain	
CBFM-8	The application should provide either cooperative or competitive flexibility trading model setups	
CBFM-9	The application shall allow the users to visualize their energy transactions	
CBFM-10	The application shall allow the injection of flexibility target levels into the users' smart contracts	

Performance requirements

Performance Requirements	
ID	**Requirement**
CBFM-11	The application should do the matching and display the flexibility transactions results within 60s after the process is initiated.

Interface requirements

Interface Requirements	
ID	**Requirement**

CBFM-12	The application should provide a web interface for users featuring menus, toolbars, buttons, panes, containers, grids allowing for easy control by a keyboard and a mouse.
CBFM-13	The application should provide web inferences to visualize the predicted energy data, blockchain energy transactions and market session information.
CBFM-14	The application should provide web interfaces to submit flexibility bids and offers.
CBFM-15	The application should store the energy flexibility bids and offers submitted on a trading session as well as the energy transactions on the blockchain allowing other systems/components to get data using rest APIs.
CBFM-16	The application should get the energy prediction data as well as the monitored energy data from a database using rest APIs.

Design constraints

No such constrains have been identified at this point

Software system attributes

Software System Attributes		
ID	**Requirement**	**Software Attribute**
CBFM-17	The application shall allow the registration of flexibility bids and offers in a tamper proof manner on the blockchain	Security

Technology stack

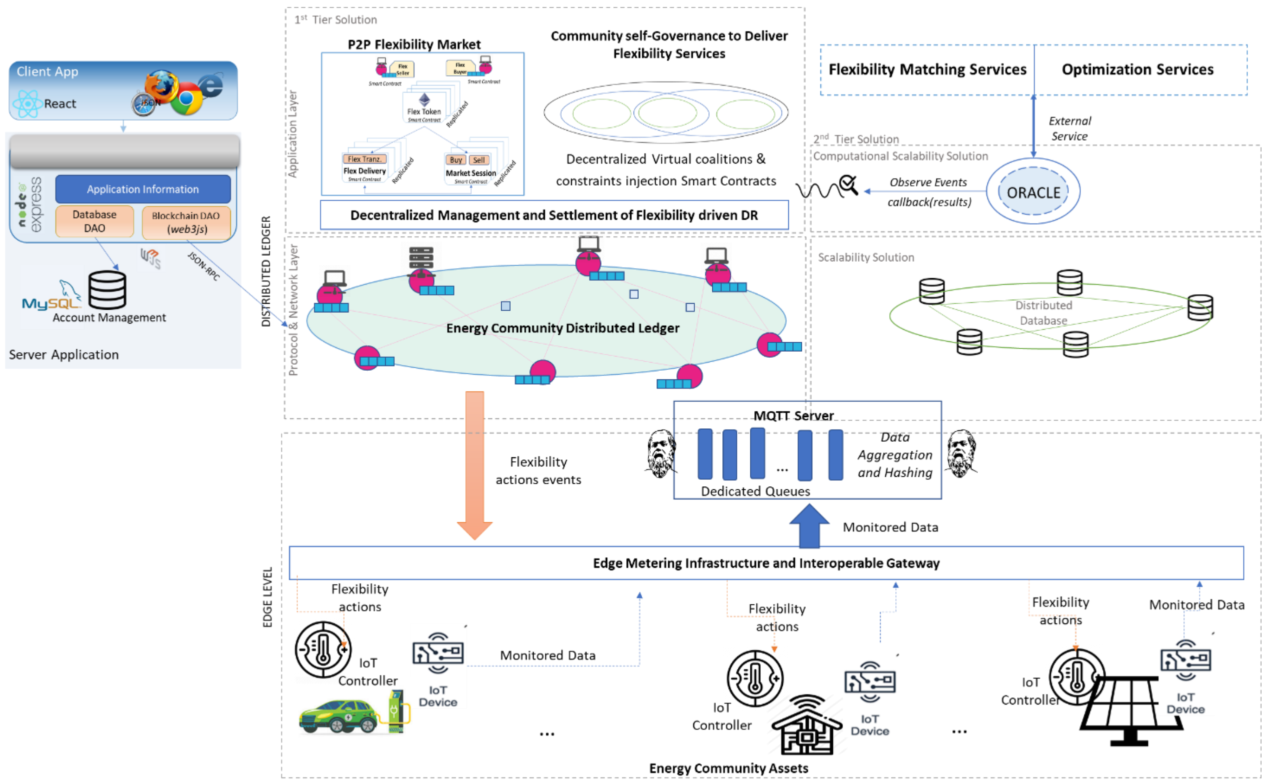


Figure 10 Technology Stack - DLT Enablers for Decentralized VPP

Verification

Verification of the attributes will be performed through unit tests and integration tests.

2.2.3.2.2 Decentralized Management and Settlement of Flexibility driven DR (DMSFDR)

Functional requirements

Functional Requirements		
ID	**Requirement**	**Related Use Case**
DMSFDR-1	The tool shall allow users to visualize their monitored energy data compared with flexibility values promised in the agreed transactions	LLUC1_1 LLUC1_2 LLUC2_1 LLUC3_1 LLUC4_1 LLUC4_2 LLUC4_3
DMSFDR-2	The tool shall allow users to visualize potential deviations from promised flexibility values	
DMSFDR-3	The tool shall allow users to visualize the penalties and rewards established based on their conformance levels	
DMSFDR-4	The tool shall allow the usage of non-financial schemes for rewarding or penalizing the users that are not delivering as expected	
DMSFDR-5	The tool shall settle the users accounts based on registered transactions and monitored energy data	
DMSFDR-6	The tool shall replace the users that are not delivering the expected levels of energy flexibility	

Performance requirements

Performance Requirements	
ID	**Requirement**
DMSFDR-7	The tool should do the settlement as close as possible to real time (about 30 minutes).

Interface requirements

Performance Requirements	
ID	**Requirement**
DMSFDR-8	The application should provide a web interface for users featuring menus, toolbars, buttons, panes, containers, grids allowing for easy control by a keyboard and a mouse.
DMSFDR-9	The application should provide web inferences to visualize the monitored energy data, expected levels of flexibility, potential deviation, penalties/rewards, users' wallets, etc.
DMSFDR-10	The application should get the monitored energy data from a database using rest APIs.

Design constraints
No such constrains have been identified at this point

Software system attributes

Software System Attributes		
ID	**Requirement**	**Software Attribute**
DMSFDR-SRS-11	The tool shall assure the privacy of users' energy data	Security

Technology stack

React JS is used to provide a web-based graphical user interface for the forecasting tool. MySQL will be the database of the tool and a REST API is used to get / store the energy data and prediction results. SciKit Learn, KERAS and TensorFlow will be used to define and train the energy prediction models. The smart meters data will be integrated using queuing systems such as RabbitMQ.

Verification

Verification of the attributes will be performed through unit tests and integration tests.

2.2.3.2.3 Community self-Governance to Deliver Flexibility Services Tool (CGDFS Tool)

Functional requirements

Functional Requirements		
ID	**Requirement**	**Related Use Case**
CGDFS-1	The application shall allow users to login.	LLUC1_1 LLUC1_2 LLUC2_1 LLUC3_1 LLUC4_1 LLUC4_2 LLUC4_3
CGDFS-2	The application shall allow users to visualize their forecast of energy consumption, production, or flexibility	
CGDFS-3	The application shall allow users to visualize the requirements and constraints of the flexibility service to be delivered	
CGDFS-4	The application shall allow users to self-organize in coalitions inside the community for meeting the flexibility service constraints	
CGDFS-5	The application shall allow the injection of flexibility target levels into the users' smart contracts	
CGDFS-6	The application shall allow users to visualize the coalition and their monitored energy values compared to the agreed ones	
CGDFS-7	The application shall allow the storage of energy flexibility transactions on the chain in a tamper proof manner	

Performance requirements

Performance Requirements	
ID	**Requirement**
CGDFS-8	The application should determine the coalition organizations for flexibility service delivery and display the results within 60s after the process is initiated.

Interface requirements

Interface Requirements	
ID	**Requirement**
CGDFS-9	The application should provide a web interface for users featuring menus, toolbars, buttons, panes, containers, grids allowing for easy control by a keyboard and a mouse.
CGDFS-10	The application should provide web inferences to visualize the predicted energy data, blockchain energy transactions and community organization in virtual coalitions.
CGDFS-11	The application should store the virtual coalitions, the flexibility services constraint as well as the energy transactions on the

	blockchain allowing other systems/components to get data using rest APIs.
CGDFS-12	The application should get the energy prediction data as well as the monitored energy data from a database using rest APIs.

Design constraints
No such constrains have been identified at this point

Software system attributes

Software System Attributes		
ID	**Requirement**	**Software Attribute**
CGDFS-13	The application should determine community virtual coalitions for meeting flexibility services constraints with at most 10% tolerance level	Reliability

Technology stack

React JS is used to provide a web-based graphical user interface for the forecasting tool. MySQL will be the database of the tool and REST API is used to get / store the energy data and prediction results. SciKit Learn, KERAS and TensorFlow will be used to define and train the energy prediction models. The smart meters data will be integrated using queuing systems such as RabbitMQ.

Verification

Verification of the attributes will be performed through unit tests and integration tests.

2.2.3.2.4 Interoperable Home automation Gateway (IHG)

Functional requirements

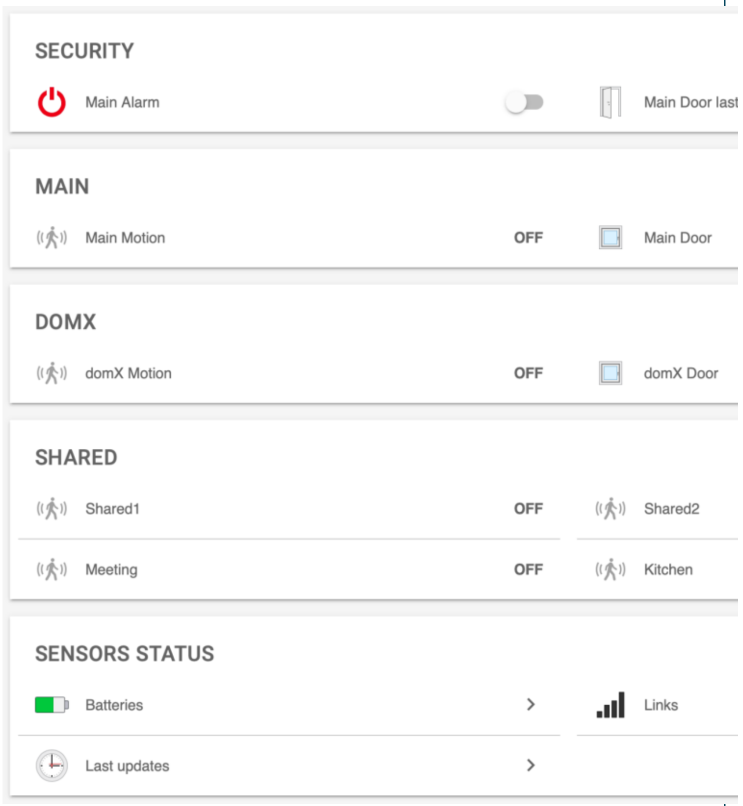
Functional Requirements		
ID	**Requirement**	**Related Use Case**
IHG-1	The tool should support the dominant wireless protocols of the smart home domain (Wi-Fi, Bluetooth, ZigBee, etc.)	LLUC4_1 LLUC4_2 LLUC4_3
IHG-2	The tool should offer interoperable smart management of a wide variety of sensors, meters, controllers and home appliances commonly found in the home environment.	
IHG-3	The tool should support interoperable smart home automation across vendor ecosystems	

IHG-4	The tool shall enable the setup of automation rules through a common approach.	
IHG-5	The tool shall provide control of all home assets through a uniform user interface.	

Performance requirements

Performance Requirements	
ID	**Requirement**
IHG-6	The tool should provide real-time updates about the status of each connected appliance/meter/sensor, with acceptable delays lower than 30 s.

Interface requirements

Performance Requirements	
ID	**Requirement**
IHG-7	The tool shall provide home users to use all connected appliances through a uniform user interface. (smartphone application/ user dashboard)
IHG-8	The tool shall allow integration with other components/services over MQTT and a REST API  SECURITY Main Alarm Main Door last MAIN Main Motion OFF Main Door DOMX domX Motion OFF domX Door SHARED Shared1 OFF Shared2 Meeting OFF Kitchen SENSORS STATUS Batteries Links Last updates *Figure 11 User dashboard for monitoring/controlling the Interoperable Gateway for Home Automation (IHG)*

Design constraints
No such constraints have been identified at this point

Software system attributes

Software System Attributes		
ID	**Requirement**	**Software Attribute**
IHG-9	The tool should ensure data privacy, so that no personal data can be exposed without the user's consent	Privacy
IHG-10	The tool should ensure security, so that data exchange between the gateway and external services is always encrypted	Security
IHG-11	The tool should enable the integration of new protocols/device types with ease.	Maintainability

Technology stack

The gateway is based on the Raspbery Pi device, directly supporting the Wi-Fi and Bluetooth protocols and the ZigBee protocol through an additional wireless interface. The gateway runs the OpenHAB home automation framework, enabling monitoring and control of a wide variety of devices/sensors/appliances across vendor ecosystems. The gateway runs a local instance of the Mosquitto MQTT broker that is used for communicating with external services and offloading of locally collected data to a cloud hosted Influx time-series DB. End users use the OpenHAB smartphone application and dashboard for interacting with connected devices in real-time. Access to historical data is enabled through a Grafana dashboard.

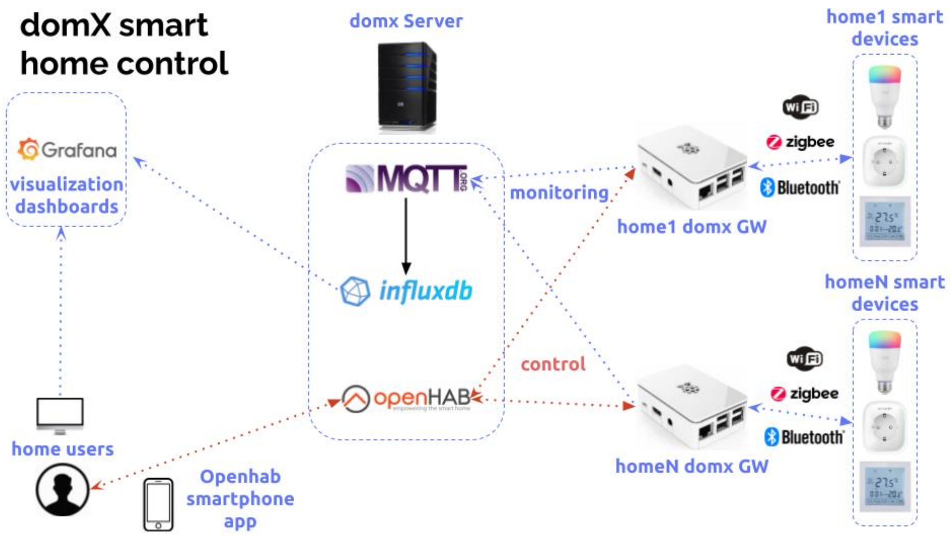


Figure 12 Interoperable Gateway for Home Automation (IHG) Technology Stack

Verification

Verification of the attributes will be performed through unit tests and integration tests.

3 BRIGHT Technologies specifications

3.1 Actors

The identification of actors is a crucial aspect in defining a software architecture. In this context, an actor is defined as a human user, external hardware, or a software application that interacts with the BRIGHT system. They often do not represent specific real-world entities but rather a role or a model of them. For instance, the same real-world entity might play different roles and hence different actors in BRIGHT. Their interactions with the system can take different forms: for example, human actors might interact with the application through the web browser or a smartphone, while a hardware component might interact by exchanging data or requesting the execution of a specific task.

In this specific context, actors can be grouped and detailed according to the area they belong to. In the following table, actors are listed and described using the three-dimensional system defined by SGAM - Smart Grid Reference Architecture Framework (Figure 13); for each dimension, the most meaningful value has been selected. The three views offered by the framework are interoperability, domains, and zones. Interoperability is defined as the ability of two or more systems to be able to exchange information and use it to work in a cooperative way. Each layer, in the interoperability view, describes in which zones of information management interactions between domains take place. Domains are related to the physical grid and are identified by each one of the steps of the electrical energy conversion chain, from the generation to the customer premise. Finally, zones represent the power system management with the use of hierarchical levels.

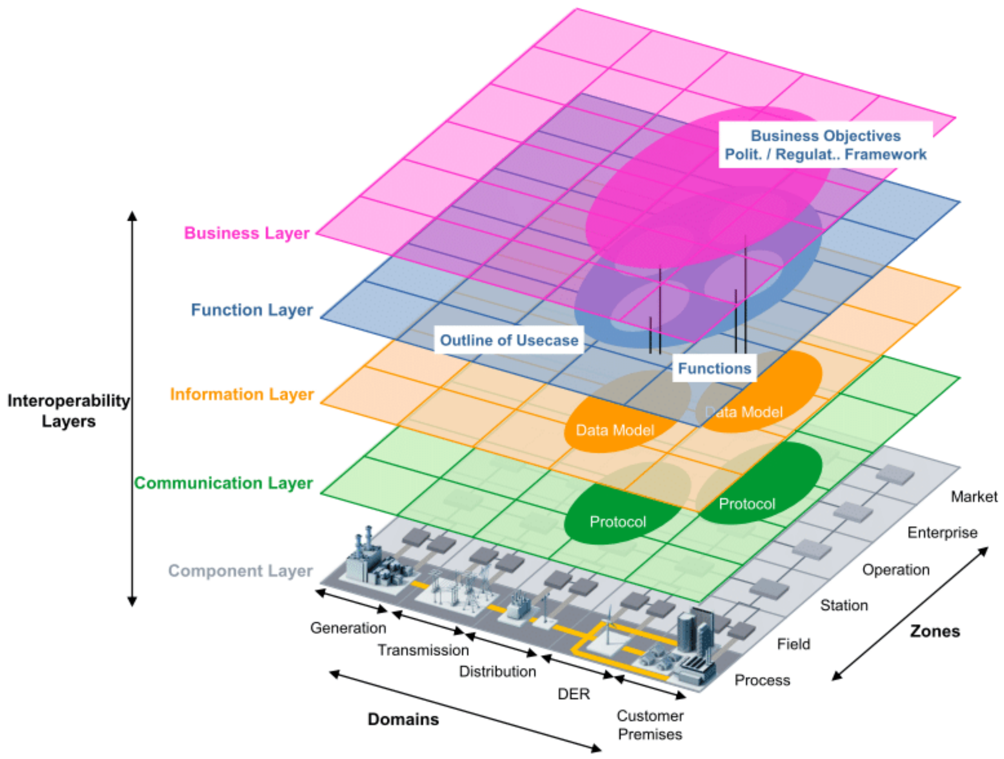


Figure 13 The Smart Grid Architecture Model

The identified actors in the system, according to the use cases, are the following:

Table 2 List of identified BRIGHT actors

Actor	Description	Interoperability	Domain	Zone
Distribution System Operator (DSO)	Company that owns and manages the energy networks. Responsible for engaging the users, and providing energy to consumers and LECs when needed, for example during shortage of local renewable energy. Benefits from the energy flexibility provided by BRIGHT.	Component	Distribution	Enterprise
Transmission System Operator (TSO)	Company that owns and manages the energy networks. Responsible for engaging the users, and providing energy to consumers and LECs when needed, for example during	Component	Transmission	Enterprise

	shortage of local renewable energy. Benefits from the energy flexibility provided by BRIGHT.			
Energy service provider (ESP)	Company responsible for providing energy services and for the communication with district residents.	Function	Distribution	Enterprise
Service provider	Company that provides the IoT hardware, user interfaces and energy services to consumers.	Function	Distribution	Enterprise
Technology provider	Company that is responsible for providing assets, software and IoT hardware for asset control, and other energy management services for enabling energy efficiency and consumer participation to flexibility services. Responsible also for the user engagement.	Function	Distribution	Enterprise
Utility provider	Company that carries out billing operation and it sells energy to the customers. Provides electricity and/or natural gas to consumers.	Business	Distribution	Enterprise
Consumer	Company, community, or person that is connected to the distribution grid and that consumes electric energy and/or natural gas. Engaged to DR schemes.	Business	Customer premise	Market
Energy user	Company, community, or person that uses electricity and/or natural gas. Pays bills to the energy service provider.	Business	Customer premise	Market
EV owner	Company, community, or person that owns an electric vehicle that is being charged.	Business	Customer premise	Market
Local energy supplier	Company, community, or person that is responsible for providing additional local renewable energy to the LEC energy service provider.	Business	DER/Customer premise	Market
Not domestic consumer	Company, community, or person that compared with a domestic user might have a wider range of flexibility to be	Business	Customer premise	Market

	offered.			
Producers of smart appliances	Company, community, or person that produces appliances that could increase flexibility at the energy user level.	Business	Customer premise	Market
Prosumer	Company, community, or person that is connected to the distribution grid and that can either consume or produce electric energy.	Business	Customer premise	Market
Citizen	Person that is part of a local energy community. It is not the owner of the energy contract, but its needs must be considered for the DR campaign, for example as a member of a family or a user of public buildings.	Business	Customer premise	Market
Domestic consumer	Person to be engaged by the DSO and to be involved in DR campaigns.	Business	Customer premise	Market
Energy communities	Community to be engaged by the DSO and to be involved in DR campaigns.	Business	Customer premise	Market
Aggregator	Company, or person that is responsible for engaging energy users and coordinate them to act as a single entity for providing flexibility services to the DSO.	Business	Customer premise	Market
Forecast provider	Provider of weather-forecast related data.	Function	Customer premise	Operation
P2P energy marketplace operator	Company, or person that works as an operator of the P2P energy marketplace.	Business	Customer premise	Market
Charging station	Charging stations for EVs	Component	Customer premise	Process
Device	Hardware device involved in the communication between assets, data acquisition, and the control algorithms.	Component	Customer premise	Process
EVs, flexible loads and storage system	Deployed devices suitable for DR campaign involvement	Component	Customer premise	Process
Flexible device	Controlled device	Component	Customer	Process

			premise	
IoT Device	IoT devices provided by the service provider	Component	Customer premise	Process
Legacy gas boiler	Legacy heating and DHW gas boiler	Component	Customer premise	Process
Automatic metering infrastructure (AMI)	System that measures, collects, and analyzes energy usage, often in real-time. Communicates with metering devices, like smart meters, either on request or on a schedule	Communication	Customer premise	Operation
Control algorithm	Responsible for calculating and sending control signals to the assets for their optimal operation	Communication	Customer premise	Operation
Energy Management System (EMS)	A system that computes and forwards optimal control actions from model provider actor to assets	Communication	Customer premise	Operation
Flexibility marketplace	Marketplace designed to enable P2P energy trading	Communication	Customer premise	Market
Asset	An electric or heat energy-related asset	Information	DER/Customer premise	Process
Communal asset	Electricity-related assets on the communal level	Information	DER/Customer premise	Station
Private asset	Private electricity-related asset that allows flexibility	Information	Customer premise	Operation
Data provider	Provides the data that can be shared for developing services	Function	Customer premise	Station
Service provider	Entity that creates and offers new services	Function	Customer premise	Enterprise
Service user	User of the services	Business	Customer premise	Market
Flexible asset	Devices actively involved in DR schemes	Component	Customer premise	Operation

Another approach for organizing the different actors is to identify how they are in relationship with each other. In the following table, actors are organized logically according to their role or type of interaction they have with the system. Types of relationships are specialization, which goes from left to right in the underlying table, or generalization, which goes in the opposite direction. The aggregations are:

- Group: represent the macro group for a type of actor. For example, all kind of devices such as IoT, EV or even legacy will be grouped together in the “Device” group.
- Type: represent the first logic aggregation for actors in the same group. For instance, every energy user regardless of further specifications, will be categorized together at this level.

- Subtype: it’s the last specification for an actor. If needed, gives another specialization to an already defined type. For example, the energy data provider is a special type of data provider.

Table 3 Relations among actors of the system

Group	Type	Subtype
Subject (Person, Company, Community)	DSO	
	TSO	
	Service Provider	Energy Service Provider (ESP)
		Technology Provider
		Utility Provider
	Service User	
	Energy Provider	Local Energy Supplier
		Prosumer
	Energy User	
		Citizen
		Domestic Consumer
		Not Domestic Consumer
		EV Owner
	Energy Communities	
	Aggregator	
	P2P Energy Marketplace Operator	
Device	Flexible Device	
	IoT Device	
	Charging Station	
	EV	
	Legacy Device	Legacy Gas Boiler
System	Automatic Metering Infrastructure (Ami)	
	Control Algorithm	
	Energy Management System (EMS)	
	Flexibility Marketplace	
Asset	Communal Asset	
	Private Asset	
Data	Data Provider	Energy Data Provider
		Forecast Data Provider

3.2 Datasets

To compute and schedule energy usage for DR purposes, BRIGHT needs to access data from many different sources. Charging stations data, energy usage during the day, sensors’ data and weather-related information are just some of them. Due to their natural heterogeneity and the different methodologies used to collect them, the data management process requires a unique and tailored strategy for each available source. As a matter of fact, each pilot can rely on a subset of all the available sources; furthermore, each IoT device or system used to collect data might have a

different manufacturer with possibly closed specifications. To address this issue, BRIGHT will define and share a common structure for each kind of energy-related data. The goal of the specification is to offer a unique interface for every and each typology of data (e.g., electric energy data, heat data, weather data). In that way, the application will be able to make use of the data regardless of the initial source or format. Furthermore, BRIGHT will be able to schedule energy exchanges with the available data, knowing that not every data type will be available at the same time and for the same pilot.
A major concern about the datasets regards private and sensitive information that is usually available alongside the energy consumption data. Some of them, especially those describing home sensors and home energy usage, might contain personal information about the physical subject that owns the contract, that uses or produces energy. To avoid disclosure of sensitive data, all the datasets used in BRIGHT will be either anonymized, or be in public domain, or not contain any private information at all.
The following table shows the main datasets defined in the deliverable D1.2.

Table 4 Identified datasets of BRIGHT system

Dataset Name	Description	Country
Charging stations	Total power used by the system, power, current, voltage, number of currently ongoing charging sessions, session data.	Belgium
Smart meter data	Electricity and heat demand for separate households.	Belgium
District heating data	Temperature, flow and energy for heat sources and heat sinks. Temperature in different buffer tanks. Temperature flows in individual heat exchangers. Thermostat state and room temperature in individual living units. Outside temperature.	Belgium
Residential smart meter data	Net consumption and injection measurements for houses with a 15-minute resolution.	Belgium
Residential flexibility data	Consumption from households, PV production profiles, flexibility information from smart appliances (mainly whitegoods).	Belgium
Building BMS & sensor data	BMS: Measurements for the HVAC system, weather info, context data within the building (temperatures, status of remotely controllable blinds and windows, air quality). IoT sensors: measurements on air quality, noise, radiator valve status, status of windows and doors. BIM model of the whole building.	Belgium
Building BMS & sensor data (sensor, actuator, and BIM data for residential living lab)	Historical and real-time data for environmental parameters (temperature, air quality, user presence, light intensity, weather info, detailed energy consumptions) and status info on building systems (HVAC, blinds, windows, doors, curtains, lights).	Belgium
Electricity submeter data	Electricity measures.	Greece
Indoor conditions data (temperature, humidity)	Visualization of indoor parameters (temperature, humidity).	Greece

Home usage patterns (door contacts, human presence)	Real time monitoring as Boolean values for home usage patterns.	Greece
Home automation and monitoring of energy consumption on appliance level (smart plug, smart relay)	Data analysis for describing electricity measures and provide remote control.	Greece
Residential space heating and DHW preparation data for gas boilers	Current boiler modulation level (as percentage of max boiler output, with most common value being 24 kW). Current boiler water temperature. Current domestic hot water temperature flame. Current boiler flame state - Shows whether the boiler is ignited. Current boiler heat state - Shows whether the boiler circulator is active. Current boiler water state - Shows whether the boiler DHW is active. Outdoor temperature - Input taken from the DomX GW temperature sensor (default) or by the boiler temperature probe if it exists. Current room temperature - Reported by the thermostat or the DomX indoor climate sensor. Current room humidity - Reported by the DomX indoor climate sensor. Desired boiler water temperature setting - Set by the thermostat or the DomX GW. Desired room temperature setting - Set by the thermostat or the DomX GW. Desired DHW temperature setting - Set by the thermostat or the DomX GW. Desired boiler heat setting - Set by the thermostat or the DomX GW (Enabled/Disabled). Desired boiler water setting - Set by the thermostat or the DomX GW (Enabled/Disabled). Weather compensation trade-off that adapts the aggressiveness of the heating control algorithm (Controls the MAX boiler water temperature to be set). Control the boiler temperature controller source (0: default, 1: weather compensation with user assigned, 2: fixed boiler temperature, 4: weather compensation with cloud controlled). Boiler temperature as calculated by the heating control algorithm.	Greece
Charging Station Data	Historical and real-time data related to charging stations involved.	Italy
Electric Vehicle Data	Historical and real-time data related to electric vehicles involved.	Italy

ASM dataset (apartment building level data)	Energy data regarding the IoT smart meters of a customer group of an apartment building.	Italy
ASM dataset (ASM headquarters data)	Data from energy units, like photovoltaic plant.	Italy
ASM dataset (users/prosumers)	Description data from customers, like energy, voltages, currents.	Italy
ELaad EV charging data	Recorded connection times, charging times, power consumption and location info.	Netherlands
ElaadNL Open Datasets for Electric Mobility Research	Overview of 10k random charging events including 15 minutes meter values per transaction.	Netherlands
BAG – Basisregistratie Addressen en Gebouwen (Key register of addresses and buildings)	Year of construction, surface area, purpose, and geographical coordinates of a building.	Netherlands
NEDU Energy Usage Profiles	Country-averaged electricity and gas usage profiles normalized over a year grouped into consumer types.	Netherlands
Service Sector and Urban-Scale Energy Demand	Service Sector and Urban-Scale Energy Demand.	Netherlands
EUROSTAT indicators (Macroeconomic indicators)	Used in descriptive analyses.	EU
Questionnaire responses from pilots	Used in descriptive analyses.	EU
Partner’s answers to Data Protection Questionnaire (Data protection questionnaire answers)	Data gathered were used to prepare D10.1 and D10.2 as well as to map the flow of personal data processing within the project.	EU

3.3 Architecture Overview

3.3.1 Architectural diagram

Errore. L'origine riferimento non è stata trovata. shows the BRIGHT's high-level architecture and ow all its components are organized and interact with each other. To better understand this structure, four different layers or groups can be identified:

- **Service or application layer:** this group contains the core services of the BRIGHT system. In this layer, all the tools and applications that enable the flexibility services operate. In the architectural diagram, the tools are further organized according to the WP they belong to.
- **Data layer:** data inside the system can be organized and shared among different tools. In order to achieve this, a set of data storage systems are available within the system for the tools to use. Every one of those containers need a proper connection to store and retrieve information.
- **External sources layer:** external sources are all the devices, gateways, metering infrastructure or storage systems that are not available inside the BRIGHT system but are required by it to work properly. Examples of external sources are IoT devices, smart meters, weather forecast data, EV stations, etc.

- **Interoperability layer:** tools and data storages need to have a common way to communicate. Since available tools and data depends on pilot and due to the necessity of creating a scalable environment, BRIGHT offers a single abstract communication system by implementing an interoperable layer. This layer's main components are the message queue system, which allows asynchronous communication, and the external source connector, that converts to standardized data that flows in and out the system.

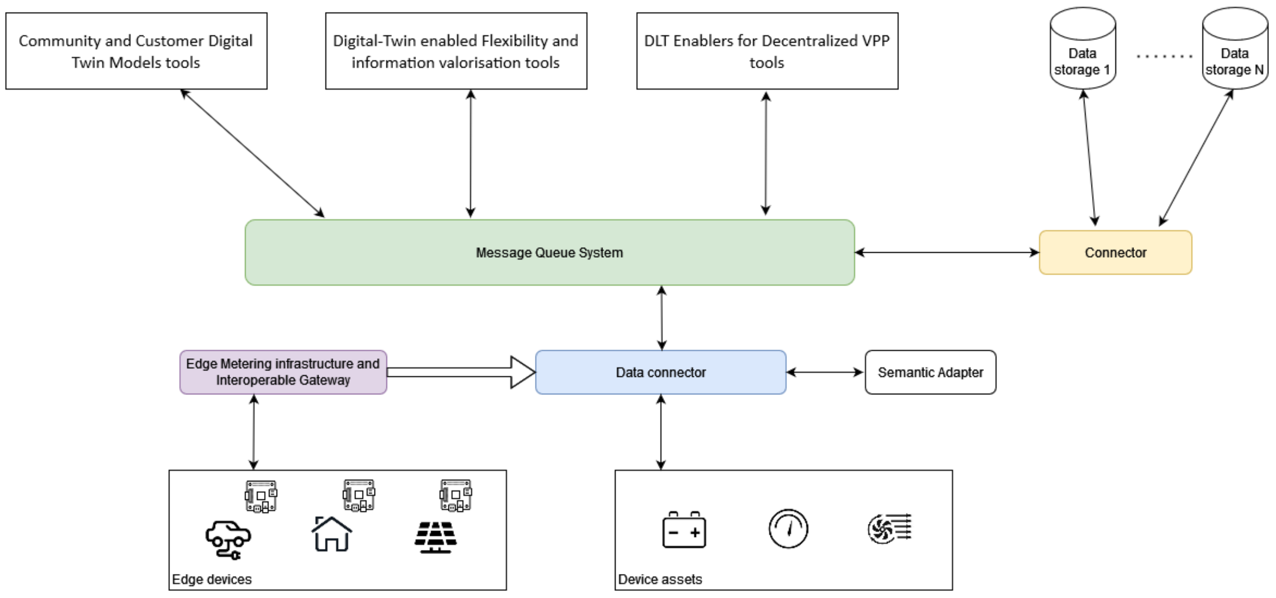

Community and Customer Digital Twin Models tools
Digital-Twin enabled Flexibility and information valorisation tools
DLT Enablers for Decentralized VPP tools
Data storage 1
....
Data storage N
Message Queue System
Connector
Edge Metering infrastructure and Interoperable Gateway
Data connector
Semantic Adapter
Edge devices
Device assets

Figure 14 Conceptual BRIGHT's Architecture

3.3.1.1 Architectural components

Service/Application Layer

Previous chapters showed that the BRIGHT system must face issues concerning the availability of components or the necessity of using different implementations for some of the modules with respect to the pilot needs. Furthermore, complex applications like BRIGHT need to be scalable and distributed, to better deal with system overloads.

To achieve that, the approach used to implement the service/application layer is the microservices architecture. In this way, every tool or component inside of the system will exists autonomously from the others and will focus on one specific task. If needed, each of them can be replicated to better handle requests and offer all the other scalability advantages. Furthermore, separated components and responsibilities allow each single tool to be implemented independently from the others, with specific technologies that are tailored to the need of the provided service.

Communication between tools is managed by the message queue system inside of the interoperability layer.

Data Layer

External sources such as IoT devices inject data into the BRIGHT architecture by using the message queue system. This information is received by the tools and used or further elaborated to achieve all the tool's specific goals. Both input and enriched data can be stored inside of the applications' storage system or in a common shared data storage. The tools are not compelled to use the data storage inside this layer, but it's a practical way to share data among them.

For each different type of storage, a connector will be available in order to access, save and retrieve information.

External Sources Layer
Energy usage measurements, IoT devices' data, and weather forecast are some of the information that flows inside the system. All of them are logically contained in a specific external sources layer. The entities inside might be not only data provider, but also consumers. For example, IoT devices need feedback to automatically adjust temperature according to all the flexibility services enabled by BRIGHT.
Due to each pilot needs and, in general, to different available sources that are specific for every environment, the entities inside of this layer can differ greatly. To standardize communication from and to the system, every single external source component that needs to connect to BRIGH must implement a proper interface. In the architectural schema, this interface is called "data connector".
A semantic adapter module will support this connector in homogenizing information according to a shared energy-related ontology such as SAREF.

Interoperability Layer
Communication between tools and all the components of the system is a very important aspect of BRIGHT. As explored in previous chapters, availability of data or services depends on the environment or the pilot preferences. Furthermore, data sources and new tools might be added or removed at any time according to specific necessities. For these reasons, communication in the system cannot be achieved by direct transmission between components.
The solution adopted is to use a messaging queue system as the only way for information exchange from, to and within BRIGHT. Every component can register on the bus as a data provider, such as for the data connector of external sources, and/or as a data consumer, such as the tools and applications available in the corresponding layer. This means that messages that travel inside of the queue will not only contain energy-related data but also information or service requests that are shared between the tools. Moreover, messages are handled asynchronously by the system, so communication is not meant to be in real time.
The messaging queue system implementation that has been selected for BRIGHT is Apache Kafka.

3.3.2 Dynamic view
There are several interactions and interrelation between tools that are going to be developed in the BRIGHT project. The dynamic view of the system focuses on describing the workflow and data exchange between the different components and actors of the BRIGHT system.
There is a particular connection between the tools developed in WP4 and those developed in WP5. The sequence diagram of Figure 15 shows the information flow among tools developed in T4.4, T4.5 and T5.1. The Clustering Block represents the clustering tool developed as a part of Task 4.4. The module uses input data from the pilots, processes this data to identify user clusters. This module then passes on these clusters to other tools, internal or external. The Modeling Block represents the digital twin model being developed as a part of Task 4.5. The module uses input data from the pilots and uses cluster information from the Clustering tool to model thermal behavior of different households. The trained model (neural network) represents the digital twin and is provided to the other tools. It also allows to generate cluster specific load forecasts (T4.2). The RL Control Block represents the reinforcement learning based controller that will be developed as a part of task 5.1. This module uses cluster information from the Clustering Block, to access cluster specific building training data and a model of that building to learn the optimum

control strategy for the building. This trained controller can then be used with pilot assets to test the performance in real-world deployments or by other partners for what-if analyses.

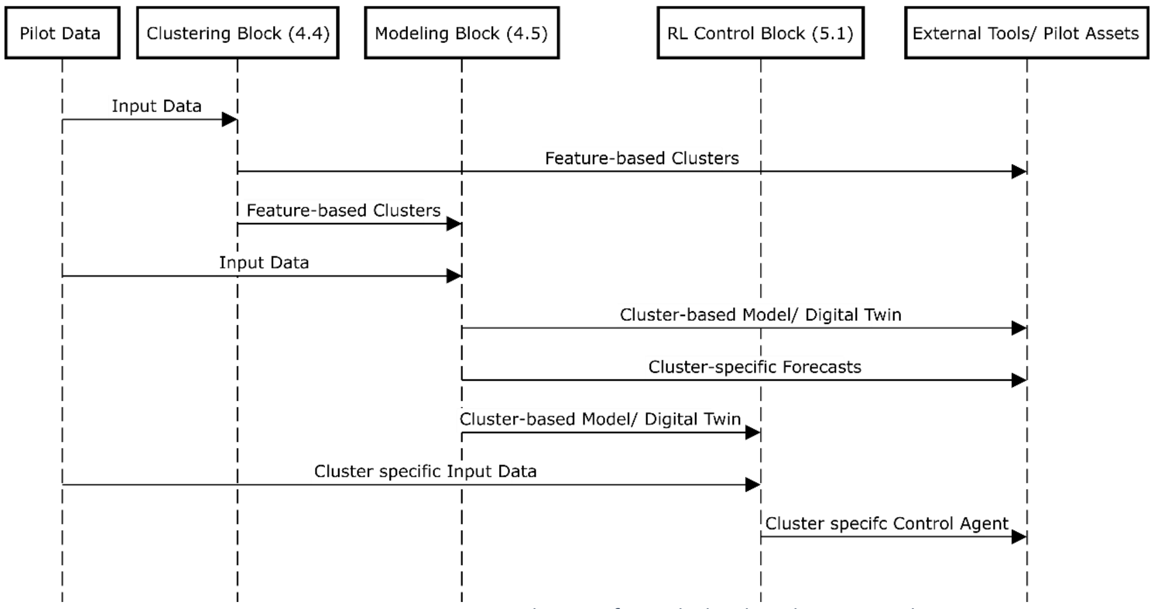
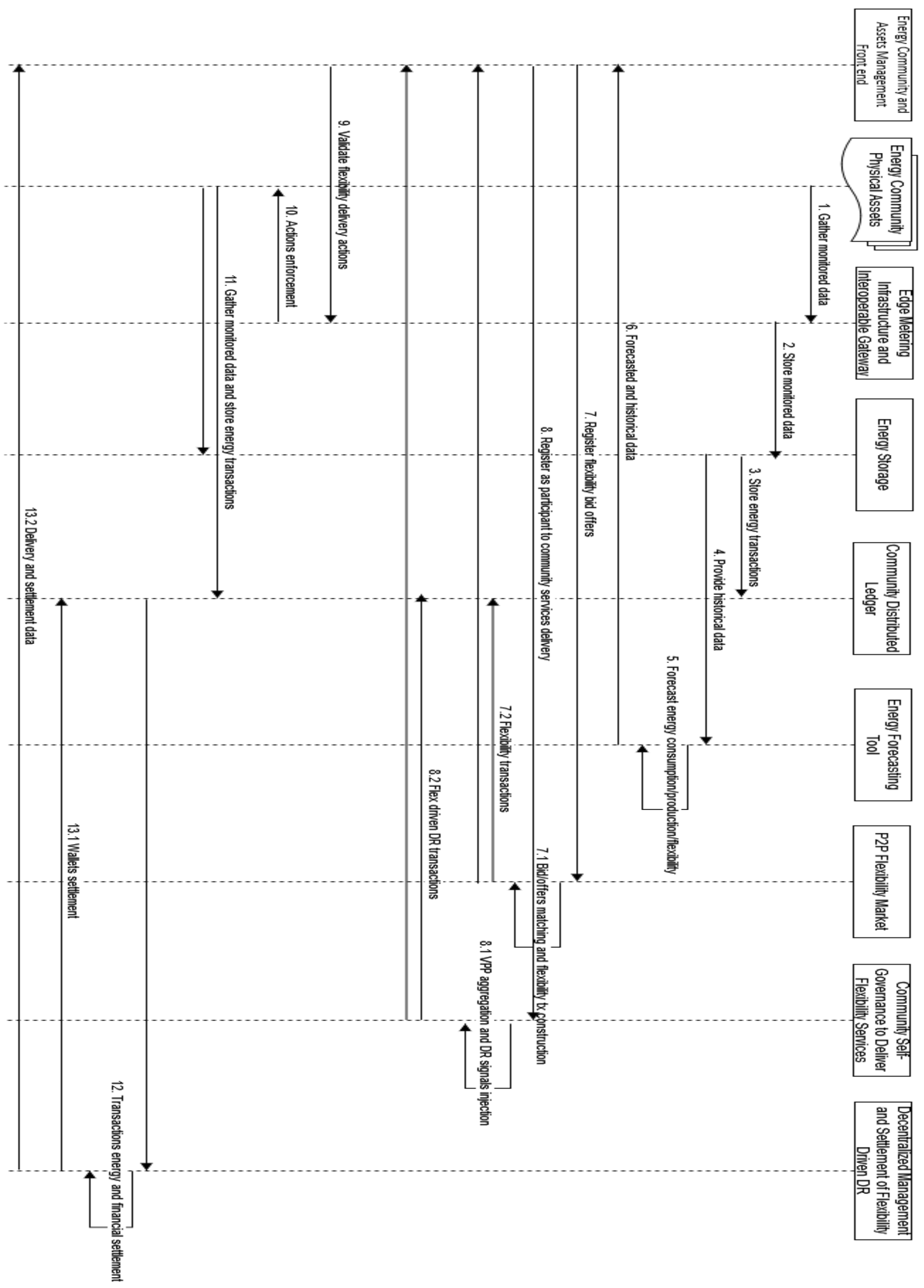

Pilot Data
Clustering Block (4.4)
Modeling Block (4.5)
RL Control Block (5.1)
External Tools/ Pilot Assets
Input Data
Feature-based Clusters
Feature-based Clusters
Input Data
Cluster-based Model/ Digital Twin
Cluster-specific Forecasts
Cluster-based Model/ Digital Twin
Cluster specific Input Data
Cluster specifc Control Agent

Figure 15 Sequence diagram for tools developed in WP4 and WP5

Another strong relation is presented between the tools developed as part of WP6. The sequence diagram of Figure 16 shows the information flow between WP6 components and the forecasting tool developed as part of WP4. The Edge Metering Infrastructure and Interoperable Gateway gathers the monitored data from the Energy Community Physical Assets. This data is stored in the Energy Data Storage and in the Distributed Ledger and can be used to forecast consumption, production, and flexibility. Based on the information of the forecast, users can decide to participate in the P2P Flexibility Market or can register to the Community Self-Governance to Deliver Flexibility Services, which allows the creation and decentralized management of coalitions in a community to deliver flexibility services on demand to the main grid. Once the flexibility delivery actions are validated, the energy transactions and financial settlement are managed by the Decentralized Management and Settlement of Flexibility Driven DR.

Figure 16 Information flow between tools developed in WP6 and the forecasting tool developed as part of WP4


Energy Community and Assets Management Front end
Energy Community Physical Assets
Edge Metering Infrastucture and Interoperable Gateway
Energy Storage
Community Distributed Ledger
Energy Forecasting Tool
P2P Flexibility Market
Community Self-Governance to Deliver Flexibility Services
Decentralized Management and Settlement of Flexibility Driven DR
1. Gather monitored data
2. Store monitored data
3. Store energy transactions
4. Provide historical data
5. Forecast energy consumption/production/flexibility
6. Forecasted and historical data
7. Register flexibility bid offers
7.1 Bid/offers matching and flexibility tx construction
7.2 Flexibility transactions
8. Register as participant to community services delivery
8.1 VPP aggregation and DR signals injection
8.2 Flex driven DR transactions
9. Validate flexibility delivery actions
10. Actions enforcement
11. Gather monitored data and store energy transactions
12. Transactions energy and financial settlement
13.1 Wallets settlement
13.2 Delivery and settlement data

3.3.3 Technologies

3.3.3.1 Microservices

Nowadays, modern applications are often distributed over the cloud and accessed from the Internet. The vast number of people that can use them forces applications to put particular attention towards availability of the services. To achieve that, software must be highly scalable, to handle system use, and also be distributed.

The common monolithic approach used to structure an application's architecture, can't really adapt, and address these requirements. The term monolithic architecture refers to the practice of designing an entire system or application within a single entity or artifact, hence the term monolithic. This approach usually simplifies some aspects of the development and delivery of the software, for instance a single code base makes it easier to reuse pieces of logic and the shared memory makes the application generally faster. According to the modern application scenario discussed earlier, a monolithic architecture has a lot of weaknesses. For example, horizontal scalability requires replication of the whole application and not only of the most stressed components; the same applies to application distribution.

A better approach to the system architecture design is offered by the microservices architecture. This methodology aims to address monolithic architecture's issues by adopting different and specific techniques, some of which are inherited from the more general Service Oriented Architecture (SOA). Microservices architecture suggests structuring the application as a collection of smaller services, each one being responsible for a specific logical and conceptual part of the system. The criteria might be around what's the purpose of the service or which domain's concept it is going to take care about. Since they don't share the same memory, the communication between them can be decided according to the system necessity or requirements. Commonly, the interaction between different services is done over an open network or protocol, such as Internet and HTTP. To avoid misconceptions, microservices are not a part of a monolithic architecture but rather a way of structuring an application. It is entirely possible, of course, that the implementation of a single microservice reflects the standard layered one.

Even if there isn't a widely accepted definition for the microservice architecture, and considering the strong independence required for each service, most sources agree that it has the following characteristics:

- Scalability: microservices are isolated one from another. This level of independence grants them the ability of being replaced and deployed separately from the others. Hence, each service scales singularly from the rest of the application, both horizontally and vertically, providing a strong benefit to resource and cost optimization when more computational power is required.
- Highly maintainable and testable: each service is specific for a domain or set of functionalities. This separation should make it small enough to be easily maintainable and testable, in contrast with monolithic applications that require test and development on the whole software for every improvement made.
- Loosely coupled: components are not aware of the presence of other ones nor the way the other entities are implemented. Each service can be considered as a single and separated module, that can be replaced or that can even not exist for a specific setting of the system.
- Integration with legacy systems: the communications between services is possible outside of the system and the memory is not shared between them. These advantages help with the integration of different or legacy systems that might be already available in some environments. Due to these level of abstractions, legacy applications can later be replaced with new ones, if necessary.

- Technology agnosticism: assuming that the communication between services is defined and agreed among them, each single service can be implemented with any technology or technique available. For instance, the owners of different services are not bounded to use the same programming language as long as they offer their services in the system in a shared way.
- Organized around business capabilities: each business capability, such as the management of energy related data or placement of a flexible asset order, is managed by a single or a set of services that are often maintained by the same team, hence simplifying interactions and development of the application.

3.3.3.2 Apache Kafka

Kafka is an event streaming platform, based on a distributed commit log. Born as a messaging queue and then evolved to a streaming platform, it is now maintained and updated by the Apache Software foundation.
In its core, it uses an immutable commit log to track every operation that is made within the system. Those operations, called events, are persisted in one or more storages that can be distributed across multiple nodes. Kafka is particularly efficient in managing great numbers of events and it retains large amounts of data with very little overhead. The complete log of all the events acts as source of the history of the system and enables the replay of each single event if needed. This is particularly useful to restore the system after a breakdown or to let newly added services be up to date to the current state.
Each service that wants to interact with the platform can be a producer (clients that publish the data), consumer (clients that use the data) or both. In Kafka, every node of a cluster is called Broker and can manage one or more different topics. Events that are submitted in the system are stored into topics, that act as a directory for messages. Topics can be further divided into partitions, that can be managed by different brokers. If necessary, the platform lets producers or consumers to interact with a specific topic-partition couple.
Kafka is particularly suitable for big data streaming. Every piece of information that flows inside of the system can be manipulated by the bundled libraries: they offer a set of extendable APIs that include common operations such as join, aggregation or windowing of data.
The platform is highly scalable, both vertically, by adding more resources such as CPU or memory, and horizontally, by creating another instance of Kafka as another cluster. Scaling can also be achieved by increasing the number of consumers and/or the number of brokers. The first approach addresses low performances when consumers are not fast enough in handling the too large quantity of messages received in the system. Instead, the second one is more advantageous when the brokers are not able to keep up with the messages' volume, but the consumers are fast enough.
Kafka adopts a publish + subscribe approach and a pull mechanism for the delivery of messages: each consumer pulls the data from a topic instead of having it pushed by the broker. This characteristic is particularly suited for scenarios in which the data received is remarkable, but consumers don't operate all at the same speed. In this case, each consumer can decide how to operate and pull the data in real-time or batch mode, according to its own necessity and capabilities.

3.3.3.3 REST

REST stands for REpresentational State Transfer. It is an architectural style and guidelines widely used for services on the Internet due to their usefulness and adaptability for distributed systems.

In contrast with SOAP, REST is not a protocol, so no official standard exists. Despite this, many implementations of RESTful architectures make use of well-known standards to work such as HTTP, URI, and JSON. The key concept of REST services is the resource. A resource is a unique entity identified by a global URI, accessible or available in a network, in general the Internet. On a resource, different operations might be allowed; for instance, for the HTTP protocol the most common ones are: get, post, put and delete. In order, they allow a resource to be read, inserted, updated and deleted.
A request to a RESTful service always returns a response. The returned data is considered the representation of a resource or entity, that might differ from the original and stored one. This representation has a specific format; the most used ones are XML or JSON.
RESTful does not have strict specifications, but six guidelines or constraints are defined for a mature APIs implementation. They are:

- Client-server: client and server must have separated concerns and responsibilities. This allows portability of the client and scalability of the server components.
- Stateless: evert request must contain all the information necessary for the client, in isolation with other previous or following requests.
- Cacheable: the data contained in a response to a request, must be labelled as cacheable (or non-cacheable), either implicitly or explicitly.
- Uniform interface: with a uniform interface, the architecture is simplified and decoupled, allowing the system to evolve seamlessly for the client. To obtain a uniform interface, the following constraints must be valid:
    - Resource identification in requests: for example, by identifying resources using URIs.
    - Resource manipulation through representations: having a representation of a resource is sufficient to modify or delete its state
    - Self-descriptive messages: each message should include all the information needed to process it by a client
    - Hypermedia as the engine of application state (HATEOAS): with the access to a starting URI, the client should be able to navigate the other resources with the use of server provided hyperlinks. This allows changes in the server without the need for the client to have such information.
- Layered system: a client should not know where and how the data is retrieved. A layered system allows the integration of intermediate servers that are not aware of the network state themselves. Intermediate servers can be act for example as security filter or data aggregators.
- Code on demand (optional): client's functionalities can be extended with the use of applets and scripts that can be retrieved from a service. In this way, the client might have a simplified structure and add other features over time.

In the BRIGHT project most involved resources are actually being accessed through the REST API, since it is the most common architecture employed by the majority of vendors/developers/systems. Most of the pilot sites expose historical data through the REST API. The API has predictable, resource-oriented URLs and uses HTTP response codes to indicate API errors. The API uses HTTP authentication and HTTP verbs which are understood by standard off-the-shelf HTTP clients. All API requests are served by a response in JSON format which can be easily interpreted and converted to other formats inside the microservices and analytics components.

3.3.4 Cyber security and data privacy

The energy domain is particularly rich of information: IoT tools data, weather forecast, and energy usage data are just some of them. To offer flexibility services and energy usage optimization, BRIGHT needs to have access and be able to compute the available data. The great variety of it, that is injected in the system, includes not only the ones defined before, but also personal data of the citizens or users that access BRIGHT. While the energy related data doesn't contain sensible information and thus can be managed in a simpler way, private data require particular care, specifically to the privacy and the security of the data. Those and the other important aspects are deeply analysed in the deliverable D2.2 called "Privacy, Ethics and Legal Requirements".
As stated in the deliverable and with focus on the architecture, all the developed tools that act in the system need to be compliant with the described specifications, especially on two important aspects:

- Cyber Securty: the dimension that concern the protection of data against theft, damage or unwanted alteration. In the BRIGHT's context, this regards the energy grid as well.
- Data Privacy: the aspect of ensuring that data is properly handled. In practice, this revolve around how data is collected and stored from a legal perspective, if it's shared with third parties and regulatory systems such as the GDPR.

The following table defines more in depth all the requirements and what kind of rules and policies are implemented in each tool.

Table 5 List of guidelines for cybersecurity aspects

Requirement	Description	Rules and policies
Implementation of security measures (in general)	The IT infrastructure shall implement adequate and appropriate security measures able to protect the data to be ingested in the infrastructure as well as its functionalities. In this respect, such measures shall include either physical or technological measures, and in any case shall be designed applying a risk-based approach, which shall consider all the components and their interactions.	• It is recommended that ICT processes in the BRIGHT project consider for each component the definition of security test procedures, acceptance thresholds and reports in order to evaluate the addressing of all the defined threats, as well as to identify new potential and unforeseen threats. • It is recommended to release the BRIGHT components with relative test reports, in order to provide evidence of security level.
Notification system	This requirement entails that the infrastructure is able to (i) detect and to send a prompt warning notification/message in case of actual attacks or even potential to the most appropriate authority; (ii) send a notification	• It is recommended to promptly notify the parties (i.e. data subject and data controller) about the status of any event occurred in the system and that can directly or indirectly impact on them. • Notification system has to adopt

	message complete with all the necessary information to detect the threats and determine the countermeasures; and (iii) the same notification system shall also be designed and construed applying adequate and proportionate security measures.	appropriate measures in order to guarantee the authenticity and integrity of alerts themselves.
Confidentiality	The requirement of confidentiality aims at protecting both personal and nonpersonal information from un-authorized access and/or use.	• It is recommended to define, implement and test appropriate management of authorisations to access and/or use data. • It is recommended to continuously update the level of reputation of the entities involved to gather, collect, access and process data. Based on the updated information, authorisation to access and/or use data have to be accordingly revised.
Availability	Means that the information circulating within the smart grid are timely and reliably accessible in case of need.	• It is recommended to identify the reasonable level of security with respect to the time constraints. Lightweight hashing algorithms and performing encryption mechanisms should be considered at the design phase of the communication protocols and mechanisms of the architecture.
Integrity	Means that the information stored or in any case circulating within the IT infrastructure cannot be modified (nor be tampered or lost), and therefore is reliable and trustable. A good practice might be the implementation of a blockchain solution.	• It is recommended to adopt techniques of data integrity management such as hashing, EDCs, etc.
Accountability	Entails that the data and the operations made on certain data can be tracked and traced back to specific and pre-authorised individuals.	• It is recommended to ensure traceability of permissions, authorisations, reputations, events, and any vital information needed for providing evidence of system accountability.

4 Conclusions

This deliverable details the design of the DR technologies to be used in BRIGHT in terms of technical and functional specifications of the project's components with an in-depth definition of the new tools and functionalities that will be developed in WP4, WP5, and WP6. Guidelines based on the ISO/IEC/IEEE 29148:2018 standard have been applied for the collection of technical information. The templates circulated among the BRIGHT partners have proved to be a powerful instrument for identifying a first version of functional and non-functional requirements for the development of BRIGHT's tools. An agile approach and methodology were followed in the collection of the software requirements illustrated in this deliverable, since itrepresents the only instrument for said purpose within the project. The collected requirements will be refined periodically during the course of the project, benefiting from the increased level of knowledge achieved in later stages and exploiting additional information not available at the current stage. After defining the set of tools specifications, the high-level architecture model has also been included to show how all ICT components are organized and interact with each other and what kind of technologies will be used for the implementation of the BRIGHT system.

References

[1] «Requirements Envisioning: An Agile Core Practice,» [Online]. Available: http://agilemodeling.com/essays/initialRequirementsModeling.htm.

[2] «Trello,» [Online]. Available: https://trello.com/.

[3] PeopleCert, «DevOps Fundamentals - Study Guide».

[4] «The Reactive Manifesto,» [Online]. Available: https://www.reactivemanifesto.org/.

[5] «Configuring Kafka for reactive systems,» [Online]. Available: https://developer.ibm.com/components/kafka/articles/configuring-kafka-for-reactive-applications.

Annex 1 – Software Requirements Specification Templates

Assets SRS Template

Scope

Please fill in the following table describing the scope of the tools that will be developed in your WP by:

- identifying the software product(s) to be produced by name (e.g., Host DBMS, Report Generator, etc.);
- explaining what the software product(s) will do;
- describing the application of the software being specified, including relevant benefits, objectives and goals

Scope	
Name	*Example Software Requirements Specification (SRS)*
Description	*This document specifies requirements for a simple application for requirements management of software and system products.* *The application allows users to:* • *Capture requirements specifications* • *Manage requirements using custom attributes* • *Set up requirements traceability* • *Browse the requirements traceability matrix* • *Comment and review requirements* • *Filter and search requirements* • *Import requirements from MS Word or Excel* • *Export requirements to DOCX, XLSX, HTML, or CSV* • *Analyze requirements coverage and impact of changes* • *Print requirements specifications*
Goals	*The application stores documents as human readable files with open file format.* *The application runs offline without connection to any server.*

Product overview

Give a brief description of the product characteristics of the tools that will be developed in your WP by taking in consideration:

- Product perspective: Define the system's relationship to other related products. If the product is an element of a larger system, relate the requirements of that larger system to the functionality of the product covered by the SRS.
- Product functions: Provide a summary of the major functions that the software will perform.
- User characteristics: Describe those general characteristics of the intended groups of users of the product including characteristics that may influence usability, such as educational level, experience, disabilities and technical expertise.

Product overview	
Product perspective	*The application runs in the latest version of Chrome or Firefox browser on Windows, Linux, and Mac.* *The application GUI provides menus, toolbars, buttons, panes, containers, grids allowing for easy control by a keyboard and a mouse.* *The application allows import a structured MS Word document via HTML data format.* *The application allows populating a MS Word document with project data via HTML data format.* *The application allows import / export a list of requirements from / to MS Excel sheet via CSV data format.* *The application stores project data in JSON format to enable easy integration with 3rd party applications.*
Product functions	• *The application shall allow users to create a new empty document.* • *If the current document contains unsaved changes then the application shall allow users to save the changes before closing the document.* • *The application shall allow users to open a document from a chosen file.* • *[….]* • *The application shall allow users to export requirements to CSV.*
User characteristics	*The software is intended for general users with no previous experience and no specific technical expertise required.*

TOOLS SRS Template

[TOOL_NAME] Requirements

Functions

Define the fundamental actions that have to take place for each tool that will be developed in your WP.

Fill in the following table with a list of requirements for each tool, taking in consideration these guidelines for their definition:

[Condition] [Subject] [Action] [Object] [Constraint of Action]

When signal X is received, the system shall set the signal X received bit within 2 seconds

At sea state 1, the radar system shall detect targets at ranges out to 100 nautical miles

The invoice system shall display pending customer invoices in ascending order of invoice due date

Functional Requirements		
ID	**Requirement**	**Related Use Case**
SAMPLE-SRS-1	*The application shall allow users to create a new empty document.*	*UC-1 (Create Document)*
SAMPLE-SRS-2	*If the current document contains unsaved changes then the application shall allow users to save the changes before closing the document.*	*UC-1 (Create Document)*
SAMPLE-SRS-3	*The application shall allow users to open a document from a chosen file*	*UC-2 (Open File)*
SAMPLE-SRS-4	*The application shall allow users to save the opened document into a file.*	*UC-3 (Save File)*
SAMPLE-SRS-5	*The application shall allow users to create a document template file from the opened document.*	*UC-4 (Document Template)*
SAMPLE-SRS-6	*Document templates shall store structure of document sections and definition and values of requirement attributes.*	*UC-4 (Document Template)*
SAMPLE-SRS-7	*The application shall allow users to create a new document from a chosen document template file preserving the structure of document sections and the definition and values of requirement attributes.*	*UC-4 (Document Template)*
…	…	…

Performance requirements
Specify both the static and the dynamic numerical requirements placed on the software or on human interaction with the software as a whole.
For example: *95 % of the transactions shall be processed in less than 1 s.*

Performance Requirements	
ID	**Requirement**

SAMPLE-SRS-8	*The application should display the opened document within 10s after it is started.*
SAMPLE-SRS-9	*The application shall allow users to open documents up to 10000 objects and 100 file attachments with total size up to 100MB.*
…	…

Interface requirements
Specify requirements for interfaces among system elements and with external entities. Interfaces among system elements should include interfaces with the human element. Interfaces with external
entities should include other systems.
Define any interdependencies or constraints associated with the interfaces (e.g., communication protocols, special devices, standards, fixed formats). Each interface may represent a bidirectional flow of information. A graphic representation of the interfaces can be used when appropriate for the sake of clarity.

e.g. The application stores project data in JSON format to enable easy integration with 3rd party applications.

Design constraints
Specify constraints on the system design imposed by external standards, regulatory requirements or project limitations.

e.g. The amount of System memory occupied by the application must be no more than 20 MB; The application's need of hard drive space must not exceed 10 GB; […]

Software system attributes
Fill in the table below, specifing the required attributes of the software product for each tool that will be developed in your WP. The following is a list of examples of software attributes that you can use:

a) Reliability - specify the factors required to establish the required reliability of the software system at the time of delivery.
b) Availability - specify the factors required to guarantee a defined availability level for the entire system such as checkpoint, recovery and restart.
c) Security - specify the requirements to protect the software from accidental or malicious access, use modification, destruction or disclosure. Specific requirements in this area could include the need to:
   1) utilize certain cryptographic techniques;
   2) keep specific log or history data sets;
   3) assign certain functions to different modules
   4) restrict communications between some areas of the programme
   5) check data integrity for critical variables
   6) assure data privacy
d) Maintainability - specify attributes of software that relate to the ease of maintenance of the software itself. These may include requirements for certain modularity, interfaces or complexity limitation. Requirements should not be placed here just because they are thought to be good design practices.

e) Portability - specify attributes of software that relate to the ease of porting the software to other host machines and/or operating systems.

Software System Attributes		
ID	**Requirement**	**Software Attribute**
SAMPLE-SRS-10	*The application shall not send any project data to the Internet.*	*Security*
SAMPLE-SRS-11	*The application shall sanitize any data input or imported by users.*	*Security*

Technology stack

Provide a brief description of the technology stack you will use to implement the tool functionalities.

e.g. the application is developed in Node.JS version X.Y and uses MongoDB X.Y as database. The API requests are handled using the Express web application framework. The application frontend is developed using Angular version X.Y.

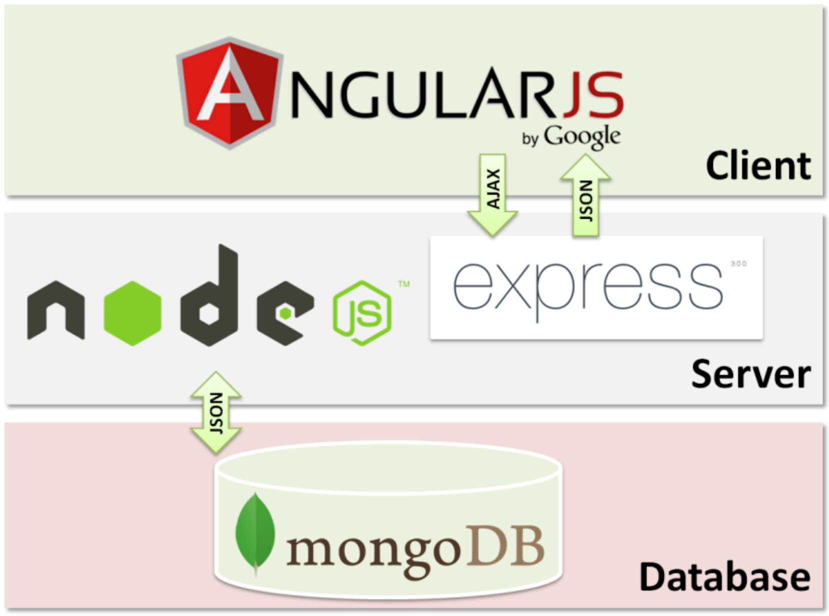


Figure X TOOL_NAME Technology Stack

Verification

Provide the verification approaches and methods planned to qualify the software tools.

e.g. Verification of the attributes will be performed through unit tests and integration tests.