Horizon Europe Programme
Research and Innovation Action

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No°101096946

Flexibility services based on Connected and Interoperable Hybrid Energy Storage Systems

FlexCHESS

Start date of project: December 1st 2022

Duration: 36 months

D1.4 report on the technical requirements

Deliverable details	
Deliverable Number	**D1.4**
Revision Number	**V5**
Author(s)	**Habib NASSER (RDIUP), Dah DIARRA (RDIUP), Nihed HUNT (RDIUP), HAMZA ELKHADAR (RDIUP), Seifeddine BENELGHALI (AMU), Tim FARNHAM (Toshiba), Stefano BIANCHI (ALWA), Emre KARABEK (Arçelik) and Yue ZHOU (CU)**
Due Date	**31-05-2023**
Delivered Date	**19-06-2023**
Reviewed by	**Seifeddine BENGHALI (AMU), Emre KARABEK (ARC)**
Dissemination level	**PU**
Contact person EC	**Antonios MARINOPOULOS**

		Contributing partners
1.	AMU	AIX MARSEILLE UNIVERSITY – France (Coordinator)
2.	ALWA	ALGOWATT SPA – Italy
3.	RDIUP	RDIUP – France
4.	ARC	ARCELIK A.S. – Turkey
5.	EL	ELEKTRO LJUBLJANA PODJETJE ZADISTRIBUCIJO – Slovenia
6.	CIP	C.I.P. CITIZENS IN POWER – Cyprus
7.	LASOLAR	LA SOLAR ENERGIA SOCIEDAD COOPERATIVA – Spain
8.	MIW	My Energia ONER, S.L. – Spain
9.	IREN	IREN SPA – Italy
9.1.	IEN	IREN ENERGIA SPA – Italy
10.	UNIGE	UNIVERSITA DEGLI STUDI DI GENOVA – Italy
11.	UEDAS	ULUDAG ELEKTRIK DAGITIM ANONIM SIRKKETI – Turkey

Associated partners	
12.	TOSHIBA EUROPE LIMITED – UK
13.	CARDIFF UNIVERSITY – UK

Technical requirements

List of figures

List of tables

Abbreviations	
FSQ: Flexibility Service Quantifier	**DSS**: Decision Support System
GUIs: Graphical User Interfaces	**KB**: Knowledge Base
BLE: Bluetooth Low Energy	**EMS**: Energy Management System
DT: Digital Twin	**PoS**: Proof of Stack
FS: Flexibility Service	**DLT**: Distributed Ledger Technology
FT: Forecasting tool	**VESS**: Virtual Energy Storage System
CHESS: Connected Hybrid Energy Storage System	

Executive summary

The deliverable D1.4 presents the technical and functional requirements that must be considered by all work packages in order to define the tailored solutions in FlexCHESS. As a whole, this document will:

- Specify the functionalities to be provided by FlexCHESS.
- Define the required variables for the use cases.
- Detail the specifications of the frontend and backend modules.
- Determine the specifications of the hardware and
- Clarify the protocols and communication layer.

In this direction, various potential discussions and brainstorming meetings were carried out between key technical contributors (AMU, Toshiba, ALWA, RDIUP, ARC and CU) and pilot sites to define the high-level functionalities provided by these different tools.

Concretely, the deliverable D1.4 reports on the two key layer (FlexPlatform and CHESS) developed for the aggregation of the distributed connected HESS and provision of the flexibility services. Also, it will consider the clean energy and digital transition in demo sites. Also, this document aims to elaborate best practices and identify lessons learnt for an effective integration and interoperability early by-design.

The structure of the D1.4 includes: Firstly, identifying the interaction between partners involved in each task and providing a design and architecture of the FlexCHESS. and providing a design and architecture of the FlexCHESS. Then, we provide the different functionalities and services in the high-level way. Then we detail the hardware and physical layer of CHESS. Finally, conclusions will be provided in the last section of this deliverable.

1. Introduction

This document delivers a comprehensive overview of the major components of the cloud-based software and hardware systems in the FlexCHESS, as well as their connections and how they will interact between each other. A structured solution will be defined to meet technical, functional, and operational requirements of the following main components. The essential goal of this document D1.4 is describing aspects and features that define the general structure of the solution, thus reducing risks associated with aligning technical and functional requirements. Additionally, the design and architecture for each different module will be showcased, attending to functional and non-functional specifications as well as data flow definitions.

Moreover, this deliverable D1.4 exploited the results and outcomes of the **T1.1** to gather and provide the requirements of interoperability levels, address identified needs and underlying standardised data exchange between FlexCHESS's flexibility service providers, operators and commercial and technical stakeholders. Therefore, the technical specifications will feed the development of FlexCHESS solutions in WP2 and WP3 and address the unified interactions of all the interested stakeholders, based on the FlexCHESS service design. In particular, **deliverable 1.4**:

- Provides the methodology of technical requirements collection.
- Provides the features and the cloud-based services that will be designed in T3.6 and implemented in the FlexPlatform
- Defines the resources to be modelled in the digital twin.
- Details the user interfaces and design to be used for both the digital twin and FlexPlatform
- Specify the interoperability layer and data management.

Also, in D1.4, the technical choices of the software and hardware will be provided. Regarding the hardware part, the skycamera, CHESS plug, gateway and lab-level will be showcased and linked requirements will be explained.

2. Requirement and specifications collection

During the previous months, RDIUP has organized general bi-weekly meetings and separate meetings with technical partners in order to better understand each module in FlexCHESS and collect the different technical specifications and functional requirements.

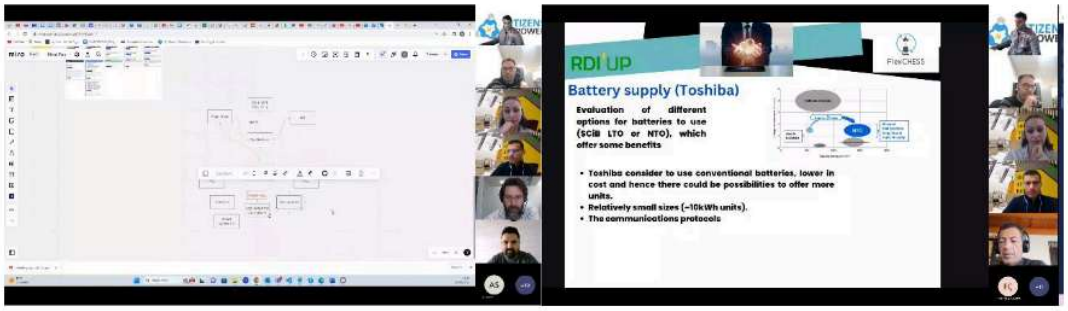


Figure 1: Bi-weekly meeting and collaborative MIRO board

A Specific MIRO board has been created to provide and collect requirements in a collaborative way. This whiteboard has facilitated the sharing and highlighting of ideas and insights to be considered in the development phases.

According to the DoA, this document aimed at gathering the requirements of interoperability levels, address identified needs and underlying standardized data exchange between FlexCHESS's flexibility service providers and commercial and technical stakeholders. Therefore, D1.4 provides technical specifications for the FlexCHESS solutions (WP2 and WP3), addressing the unified interactions of all the interested stakeholders, based on the FlexCHESS service design.

3. FlexCHESS functionalities and cloud-based services

In this section, the functionalities and the cloud-based services that will be designed in WP3 and implemented in FlexPlatform are presented.

3.1. FlexCHESS workflows and functionalities

In FlexPlatform, two key workflows will be offered as follows:

- In high-level and offline workflow: for optimization process simplified models will be used to configure the existing portfolio, simulate it and use objective functions (single or multiple) and then provide optimized HESS in an iterative manner. All elaborated knowledge will be stored in the shared KB.
- In real-time and operational workflow: in RT-DT received data from the field, interact with the forecasting tools to simulate the storage capabilities based on the needs of the FSP. The FSP engine will define the optimal flexibility service that can be provided and define the global set points for the CHESS Node. The EMS (including the Flexibility service quantifier (FSQ)) will

ensure the control and aggregation at low-level and in real-time of the different CHESSs while considering the grid constraints.

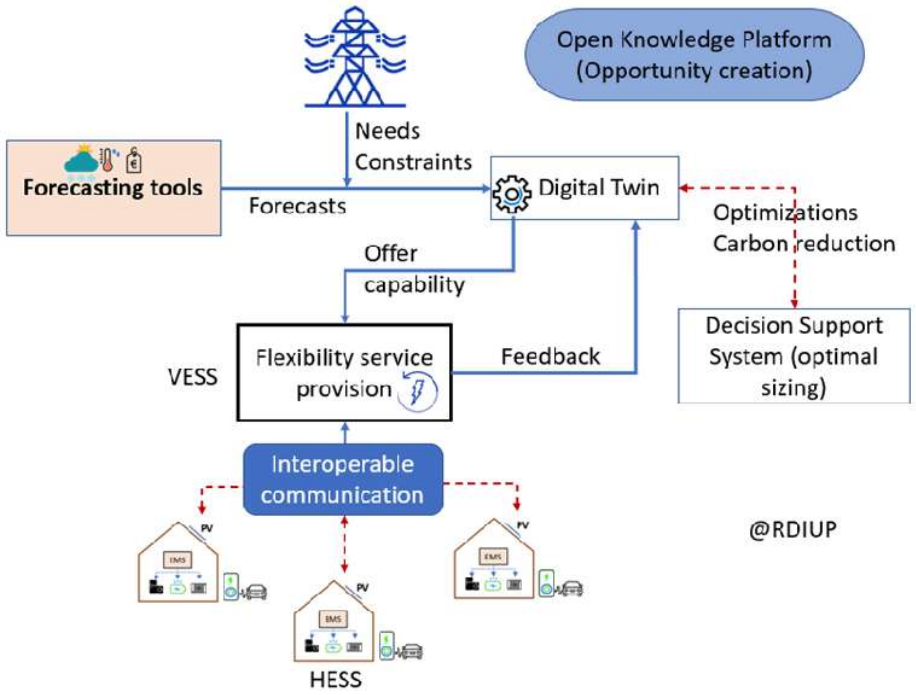


Figure 2: Global architecture of FlexCHESS solutions

As illustrated in the table below, the FlexPlatform is composed of the frontend and backend.

- **The front-end application** (mainly the GUIs) will be the component in charge of visualization, redirection, styling, notification, settings and reporting of events, providing the functional interface for the actors and visitors (index pages). Adequate UX and interfaces will be designed to easily implement an HMI that will empower the user acceptance in terms of real-time operations and ease use. The FlexPlatform will offer integration with notification, visualization, dashboard, and metrics tools that will help actors on the daily operation by ensure better understanding and explanation of the situation while providing monitoring for the energy, health, and maintenance of their CHESSs. It will provide a complete real-time situation awareness and will allow a friendly interaction between the operators and the FlexCHESS system. Additionally, the front-end application will also be in charge of the interface with the internal components and backend, such as the DT and the DSS.

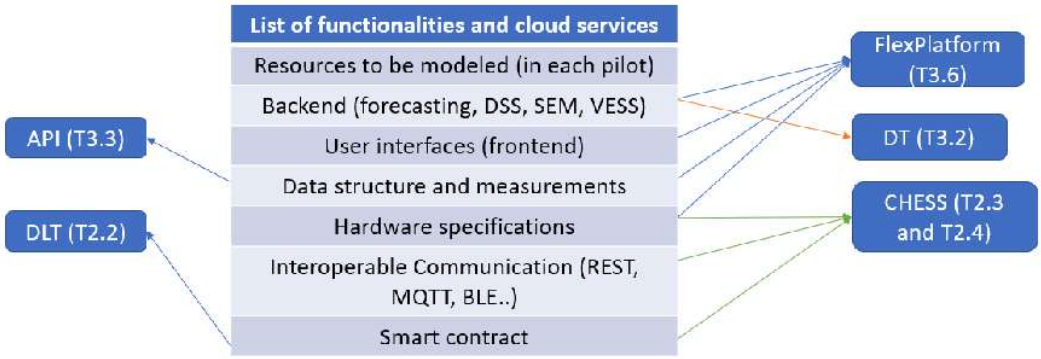


Figure 3: Interactions between tasks and functionalities

- **The backend application:** it will be developed mainly with python and intends to ensure the interaction and communication between the different features and specifications as follows:

  - **Forecasting tool (FT):** This module will allow to forecast power consumption and generation by using external APIs and advanced algorithms (e.g. SVM, NN). Training models will be also provided to ensure the tuning of the forecasting models. The API for weather (e.g; meteomatics, openweatherMap) has to be unified in all modules. This tool has to deliver also Short Term Forecasting features in interacting with skycamera.
  - **Knowledge base (KB):** This database responsible for the storing information from smart meters, gateways, CHESS-plug, other monitoring systems and elaborated knowledge from data analytics. FlexCHESS has to offer the possibility to implement decentralized repository (e.g. via Blockchain) and ensure the data protection and privacy. Also, only metadata, anonymized data and knowledge can be shared between CHESS-Node.
  - **Digital Twin (DT):** Beyond the accurate modelling of objects, HESS and assets It has to provide real-time features and monitoring. Moreover, the DT will monitor CHESS-Nodes to assess and predict their storage capabilities in interaction with FT.
  - **Flexibility Service Provision (FSP):** It includes a decision mechanism to determine the optimal Flexibility services to be activated and provide the high level setpoints to be used by the EMS. FSP can send requests to DT to assess storage capabilities.
  - **Decision Support System (DSS):** The DSS will allow the optimal sizing of additional storage system for CHESS-Node based on technical, economic, and environmental criterion. The obtained results must be assessed through simulations conducted in realistic scenarios. Various open frameworks, algorithms and solvers can be used in the DSS (e.g. MILP, genetic algorithm, CPLEX, Optimize of SciPy (e.g. trust-constr method)).
  - **Energy Management System (EMS):** EMS has to aggregate different assets at low-level, communicate through the MQTT broker and APIs provided by Arçelik and control in real time. The included FSQ will allow to quantify FS and ensure corrections in case of not-allowed deviations. Also, FS reconciliation has to be offered at the end of each service.
  - **DLT:** The blockchain layer has to be based on Proof of Stack (PoS) (e.g. Cardano or Ethereum 2.0). The DLT will mainly ensure useful functionalities (e.g. certification of origin, and protect crucial data) and establish smart contracts between CHESS owners, CHESS Node/FlexPlatform operators in a transparent way for a fair share of benefits.

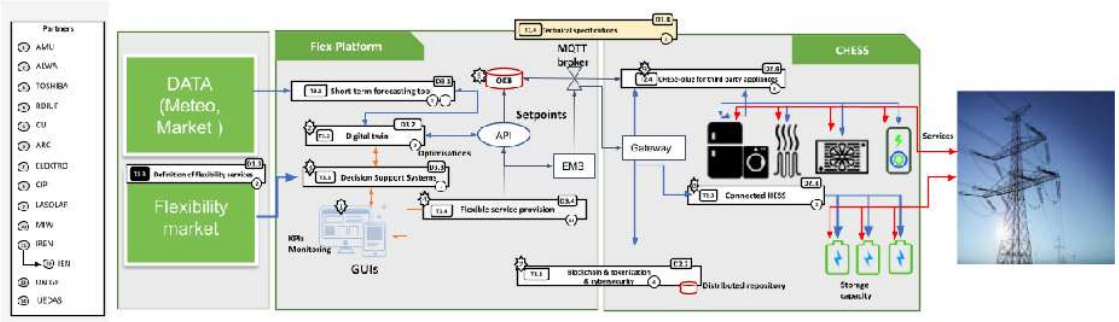


Figure 4: Updated taxonomy

Perspectives and actors for different software modules composing the FlexPlatform are listed and described in the following table 1, graphic and paragraphs. Leaders will be the key responsible for the development of each module to be integrated mainly by ALWA through the middleware.

Table 1 - List of software modules and corresponding actors

Modules	Task	Leader	Participants
FT	T3.1	Aix Marseille université	RDIUP
DT	T3.2	TOSHIBA	AMU, CU, RDIUP and ALWA
FSP	T3.4	algowATT	Toshiba, AMU
DSS	T3.3	Aix Marseille université	RDIUP
EMS	T3.5	CARDIFF UNIVERSITY	Toshiba, AMU
KB	T3.6	algowATT	All technical partners
GUI	T3.6	algowATT	All technical partners
DLT	T2.2	RDI'UP	CU

Based on the various discussions and the requirements linked to the frontend and backend above, the table 2 showcases the different functionalities to be considered and integrated in the final version. Mainly, three GUIs will be provided. The general interfaces will allow the redirection through buttons to the UIs of DSS and DT while considering the same styling for a smooth user experience.

The Table 2. illustrates the different functionalities provided by FlexCHESS for authentication, forecasting, mapping, configuration, filtering, optimisation, control, monitoring, FS provision, alerts, and smart contracting.

Table 2: FlexCHESS functionalities

Functionalities	GUI	FSP	DT	EMS FSQ/ control	DSS	FT	KB
User account creation and profile of user management	X (Signup and reset interfaces)						
Authentication security JWT	X (Login UI)						
Forecasting	X (list menu and display results sky camera)					**X**	

Listing of CHESS Node	X (List menu)						
Map CHESS plug	X (Mapbox)						
Filtering and search	X						
Configure CHESS Node (HESS)	Button		X (styling GUI)				
Optimize sizing (offline, planning of storage size and components hybridization)	Button				X (styling GUI, simplified models)		
Assets control/commands				X (setpoints)			
RT data monitoring				X			
Dashboard and visualization (**DV**)	X						
CO2 monitoring	X			X		X	
Grid constraints evaluation	Button		X			X	
Storage capability evaluation (operational way)	**DV**		X				
FS matchmaking and optimization		X					
FS activation		X					
FS monitoring	**DV**			X			
FS deviation detection	Alerts			X			
FS correction				X			
FS quantification/ reconciliation				X			

Maintenance and failure detection			X				
Alert managements	X		X	X			
knowledge elaboration and Data storage				X			X
Middleware	X						
Smart contract	X						
Certificate of origin	X						
Rating (stars)	X						

The key users or actors in the FlexCHESS ecosystem are: visitors, superadmin, CHESS owners, operators of the CHESS Node and FlexPlatform (that can be SME, DSO, aggregators ESCO, ECs, FS providers etc …). As shown in the Table 3, FlexPlaftorm offers various services based on role of users.

Table 3: FlexPlatform services

Visitors	**Superadmin**	**CHESS owner**	**CHESS Node operator**	**FlexPlatform operator**
browse chess nodes	Create/update/ delete/lock users mainly operators	browse chess-nodes	browse chess plugs and other CHESS Owner	Create, update, delete, user accounts
browse and map public chess	retrieve, add, update, delete parameters features	retrieve, add updated delete ESS	add, update, delete CHESS Node	add, update, delete CHESS Node baselines
retrieve forecasting		add update delete location	add/delete a chess-plug from the node	retrieve, filter, delete, create knowledge
		retrieve, add update delete sensors	retrieve, filter, confirm, and check and cancel market offers	retrieve, filter, delete, create reporting
		get the potential and benefits of their assets	configure, update, delete a CHESS portfolio (DT)	configure, update, delete a CHESS portfolio (DT)
		send message or notification to CHESS Node or operators	check the storage capability	retrieve, filter, delete, optimize portfolio via DSS
		Associate user with assets	Get/delete forecasting	retrieve and confirm flexibility

				service requirements
			add/get sky camera forecasting	
			Request, delete, retrieve a high-level optimisation	
			browse economic profits	
			Publish offers for storage regarding the capacity	
			notify operators when alerts	
			get origin of storage	
			Retrieve, filter knowledge	
			retrieve and confirm flexibility service requirements / conditions maybe through smart contract	

CRUD operations [CRUD] refers to the four basic operations a software application should be able to perform through HTTP request method – Create (POST), Retrieve (GET), Update (PUT), and Delete (DELETE). In the developed apps, users must be able to create data, have access to the data in the UI by reading the data, update or edit the data, and delete the data. In full-fledged applications, CRUD apps consist of 3 parts: an API (or server to be provided by Arçelik), a database, and a user interface (UI). The API contains the code and methods, the database stores and helps the user retrieve the information, while the user interface helps users interact with the app. The services in the table 3 have to be translated during the development and integration phases into CRUD operations as illustrated in Table 4.

Table 4: CRUD operations in FlexCHESS

		FLEXCHESS USER CATEGORIES				
ITEM	ACTION	VISITOR / GUEST	SUPERADMIN	CHESS OWNER	CHESS NODE OPERATOR	FlexPlatform OPERATOR
USER	CREATE		X	X	X	X
USER	RETRIEVE		X	X	X	X
USER	UPDATE		X	X	X	X
USER	DELETE		X (lock)	X	X	X
ESS	CREATE			X		
ESS	RETRIEVE	X		X		
ESS	UPDATE			X		
ESS	DELETE			X		
CHESS	CREATE			X	X	
CHESS	RETRIEVE	X		X	X	
CHESS	UPDATE			X	X	

CHESS	DELETE			X	X	
CHESS-NODE	CREATE				X	
CHESS-NODE	RETRIEVE	X			X	X
CHESS-NODE	UPDATE				X	
CHESS-NODE	DELETE				X	
CHESS-PLUG	CREATE			X		
CHESS-PLUG	RETRIEVE	X		X		X
CHESS-PLUG	UPDATE			X		
CHESS-PLUG	DELETE			X		
CHESS-BASELINE	CREATE			X		
CHESS-BASELINE	RETRIEVE	X		X		
CHESS-BASELINE	UPDATE			X		
CHESS-BASELINE	DELETE			X		
CHESS-FLEXIBILITY	CREATE			X		
CHESS-FLEXIBILITY	RETRIEVE			X		
CHESS-FLEXIBILITY	UPDATE			X		
CHESS-FLEXIBILITY	DELETE			X		
CHESS-NODE-BASELINE	CREATE				X	X
CHESS-NODE-BASELINE	RETRIEVE				X	X
CHESS-NODE-BASELINE	UPDATE				X	X
CHESS-NODE-BASELINE	DELETE				X	X
CHESS-NODE-FLEXIBILITY	CREATE				X	X
CHESS-NODE-FLEXIBILITY	RETRIEVE				X	X
CHESS-NODE-FLEXIBILITY	UPDATE				X	X
CHESS-NODE-FLEXIBILITY	DELETE				X	X
REPORT	CREATE				X	X
REPORT	RETRIEVE				X	X
REPORT	UPDATE				X	X
REPORT	DELETE				X	X

FLEXIBILITY REQUEST	CREATE				**X**	X
FLEXIBILITY REQUEST	RETRIEVE				**X**	X
FLEXIBILITY REQUEST	UPDATE				**X**	X
FLEXIBILITY REQUEST	DELETE				**X**	X

The Table 4. above will be reviewed and refined in WP3 by ALWA in close collaboration with all technical partners to provide additional services for the different users.

4. Real-Time Digital Twin

Within the use-cases Real-time digital twins can be used in the optimisation process, which are continuously synchronised with the physical assets that they represent. This permits representation of the actual asset technical data and operational state as well as providing simulation model support.

4.1. Digital Twin requirements

The proposed approach to supporting Energy digital twins within the FlexCHESS project is to use the Asset Administration Shell (AAS) approach defined within the Industrial Digital Twin Association (IDTA), see Figure 5. In this approach the digital twin comprises of a set of sub-models that represent different aspects of the asset. Some of these have been completed and some are currently being defined within the IDTA and the IEC. The parameters and classes, which represent the asset technical and operational data, such as defined in the Common Data Dictionary (CDD) which is standardised within the IEC, see Figure 6 and Figure 7, and is related to the other sub-models as shown within Figure 8. In this manner the master data ontologies can be accessed from the IEC CDD database (https://cdd.iec.ch/CDD/iec61360/iec61360.nsf). The IEC CDD is an International Standard (IEC 61360-4 DB) and serves as a common repository of concepts for all industrial/technical domains based on the methodology and the information model of IEC 61360 series, and provides:

- unambiguous identification of classes and properties, and their relations.
- commonly accepted terminology and definitions based on accepted sources such as IEC International Standards, other International Standards, industry standards, or public authorities;
- hierarchies of concepts enabling users to appropriately characterize their products and services;
- relevant conditions and constraints, if necessary, on possible values of characteristics;
- technical representation of concepts including units and data types and their identification

Figure 5 : Asset Administration Shell digital twin for the asset lifecycle (from IDTA)

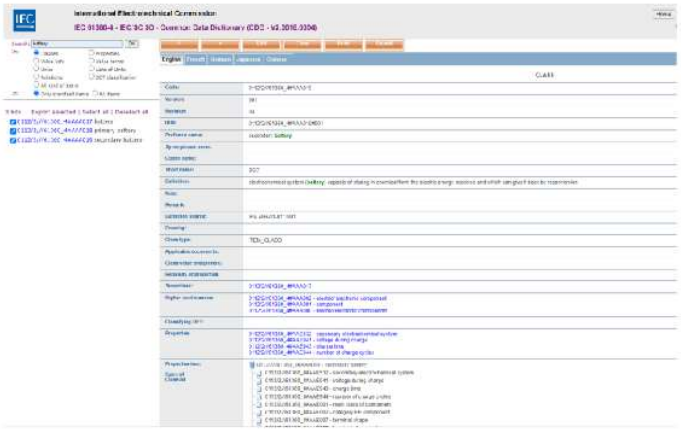


Figure 6 : Extract of secondary battery class from the IEC CDD

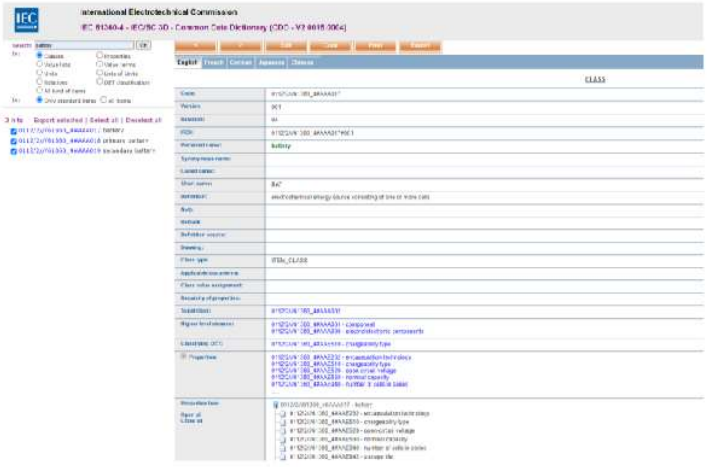


Figure 7 : Extract of battery class from the IEC CDD

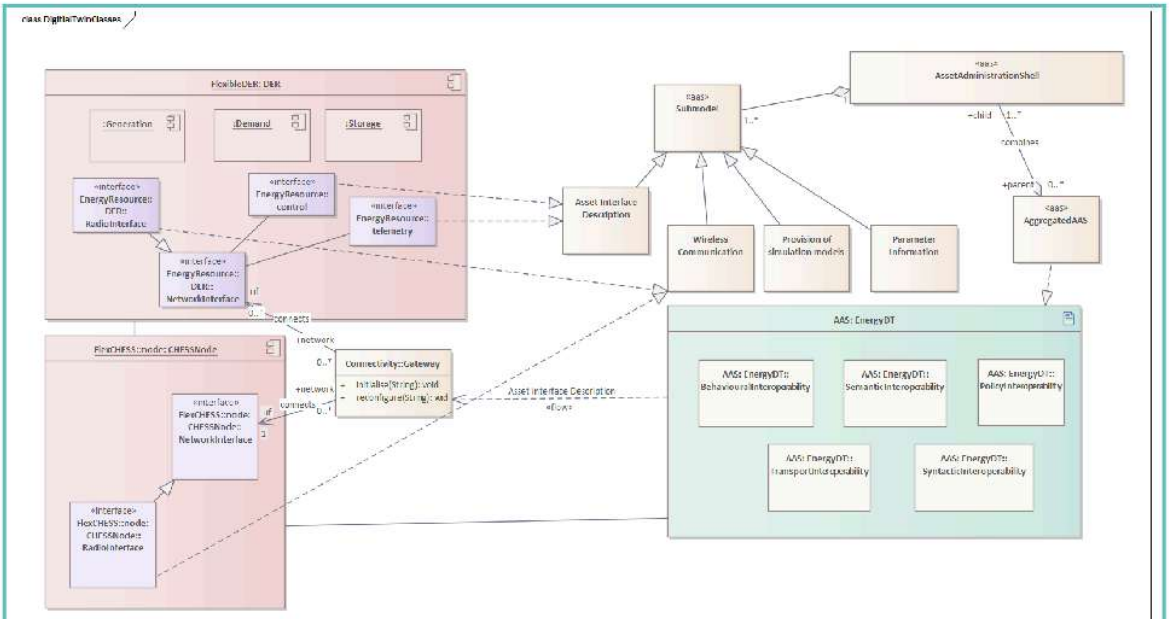


Figure 8 : FlexCHESS Digital Twin Classes

Table 5 : Extract from IEC CDD secondary battery and inherited properties (not exhaustive)

Code	Preferred name	Definition
0112/2///61360_4#AAE532	secondary electrochemical system	code of the electrochemical system of a secondary battery
0112/2///61360_4#AAE941	voltage during charge	maximum voltage of a secondary battery during charge with a constant current of 0.1 times the value of the nominal capacity
0112/2///61360_4#AAE943	charge time	value as specified by level (minmax) of the time after which a discharged secondary battery attains the fully charged state at specified ambient temperature and at specified direct current
0112/2///61360_4#AAE944	number of charge cycles	maximum number of charge/discharge cycles at the end of which the secondary battery retains 80% of its nominal capacity at specified duration
0112/2///61360_4#AAE001	main class of component	code of the main functional class to which a component belongs
0112/2///61360_4#AAE002	category EE component	code of the category to which an electric-electronic component belongs
0112/2///61360_4#AAE007	terminal shape	code of the shape of the terminals of an electric, electronic or electromechanical component
0112/2///61360_4#AAE008	terminal placement	code indicating the placement of the terminals of an electric-electronic or electromechanical component
0112/2///61360_4#AAE022	outside diameter	value as specified by level (miNomax) of the outside diameter of a component with a body of circular cross-section
0112/2///61360_4#AAE023	terminal diameter	value as specified by level (miNomax) of the diameter of the terminals of an electric-electronic or electromechanical component
0112/2///61360_4#AAE024	terminal pitch	nominal pitch of the terminals of an electric, electronic or electromechanical component
0112/2///61360_4#AAE072	terminal length	value as specified by level (miNomax) of the length of the terminals of an electric-electronic or electromechanical component extending below the seating plane
0112/2///61360_4#AAE257	power dissipation	permissible power which may be dissipated continuously, at specified conditions
0112/2///61360_4#AAE259	shape/size code BSI	BSI code of the shape/size of an electric-electronic or electromechanical component for placement on printed circuits
0112/2///61360_4#AAE262	encapsulation technology	code indicating the encapsulation technology which has been applied in an electric, electronic or electromechanical component
0112/2///61360_4#AAE347	CECC specification	CECC code of the specification in which an electric/ electronic or electromechanical component is released under the CECC quality assessment system
0112/2///61360_4#AAE510	chargeability type	code of the chargeability type of a battery
0112/2///61360_4#AAE529	open-circuit voltage	value as specified by level (miNomax) of the open circuit voltage of a battery
0112/2///61360_4#AAE530	nominal capacity	nominal capacity of a battery
0112/2///61360_4#AAE540	current rms	maximum rms current of an electric-electronic or electromechanical component at specified ambient temperature
0112/2///61360_4#AAE633	lacquered length	maximum length of the body of an electric/ electronic or electromechanical component with axial leads, including the lacquered part of the leads
0112/2///61360_4#AAE634	terminal material	code of the terminal material of an electric-electronic or electromechanical component
0112/2///61360_4#AAE683	temperature type	IEC code of the type of temperature indicating the specific part of a component, or its environment, to which it applies
0112/2///61360_4#AAE685	temperature	temperature of a component, or its environment, as a variable

0112/2///61360_4#AAE687	quality approval authority	abbreviated name of the authority for quality certification of products
0112/2///61360_4#AAE688	thermal resistance	maximum operating thermal resistance of an electric-electronic or electromechanical component, of a specified thermal resistance type
0112/2///61360_4#AAE753	inside diameter	value as specified by level (miNomax) of the inside diameter of a component with a body of circular cross-section
0112/2///61360_4#AAE841	storage temperature	permissible temperature range at which the device may be stored without any voltage being applied
0112/2///61360_4#AAE905	dissipation derating factor	minimum required power dissipation derating factor of an electric-electronic or electromechanical component, at ambient temperatures higher than the rated temperature
0112/2///61360_4#AAE940	number of cells in series	actual number of cells in series of a battery
0112/2///61360_4#AAE942	storage life	minimum duration of storage at specified ambient temperature at the end of which a cell or battery retains 80% of its original capacity
0112/2///61360_4#AAF276	stress temperature min	minimum temperature applied to a component during cycle stress
0112/2///61360_4#AAF277	stress temperature max	maximum temperature applied to a component during cycle stress
0112/2///61360_4#AAF278	stress ambient temperature	nominal ambient free air temperature applied to a component during high temperature life test
0112/2///61360_4#AAF279	stress relative humidity	nominal value of the ambient humidity relative to saturation humidity, applied to a component during humidity resistance stress

The AAS approach to supporting digital twins provides five aspects of interoperability, which are defined in ISO/IEC 21823-1:

Transport interoperability considers the data transfer between software applications based on an established communication infrastructure between the participating software applications. This facet is not addressed in this part of the series but will be considered in further parts of the series.

Syntactic interoperability considers the data format that the exchanged information can be understood by the participating software applications. This facet is not addressed in this part of the series but will be considered in further parts of the series.

Behavioural interoperability considers the expected outcomes to interface operations. This facet is addressed by the Asset Administration Shell in the way that the Asset Administration Shell provides a standardized interface to software applications.

Semantic interoperability considers the meaning of the data model within the context of a subject area so that it is understood by the participating software applications. The Asset Administration Shell addresses semantic interoperability by associating well-known concepts to the data, which is exchanged between the software applications.

Policy interoperability considers the compliance with the legal, organizational, and policy frameworks applicable to the participating software systems. The Asset Administration Shell addresses policy interoperability in the following way:

- The Asset Administration Shell provides uniform identity and access control management including usage restriction for information and services of assets.
- The Asset Administration Shell enables uniform structuring of information and services of assets. This allows that the structure of information and services of an asset is defined and maintained by the Asset Administration Shell and not by the individual software applications. This simplifies information management in manufacturing industries by both reducing the effort and increasing the quality of information.

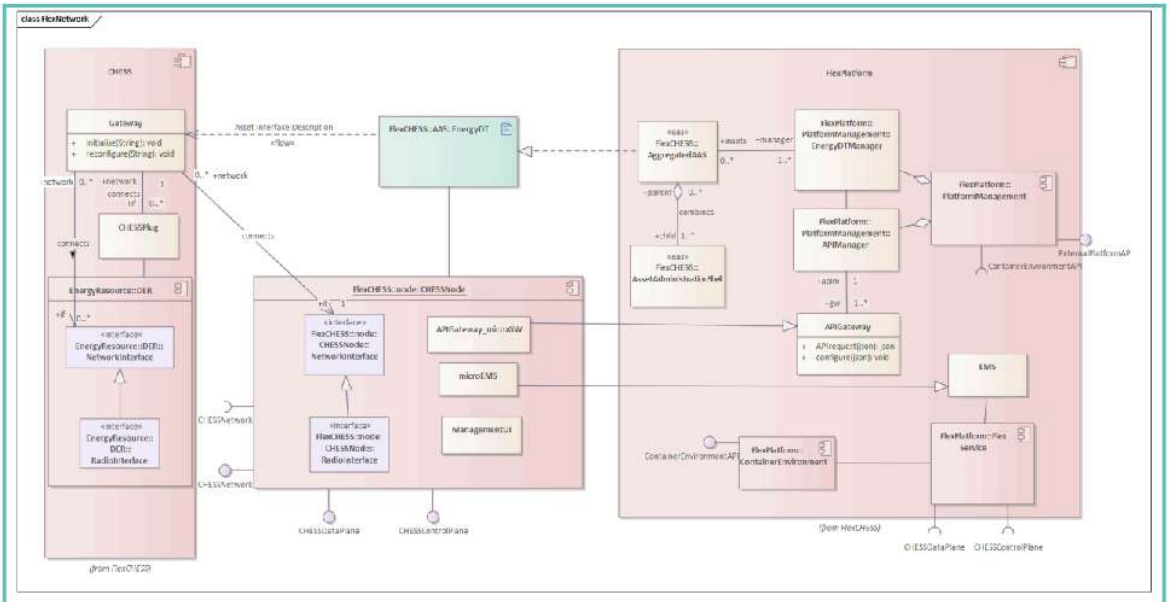


Figure 9 : FlexNetwork

The interfaces are defined within the AAS as shown in Figure 9. This permits a standardised structure and semantics required to interact with an asset or service, e.g., to request/subscribe to specific data points or to perform operations based on communication protocols such as Modbus, OPC UA, HTTP, and MQTT. The W3C Web of Things Thing Description is considered to have a base line for content and structure for submodule definition (see https://github.com/admin-shell-io/submodel-templates/tree/main/development/Asset%20Interface%20Description/1/0).

The component interfaces are shown in Figure 10. This indicates the main interfaces exposed or required by the FlexPlatform, CHESS nodes and DERs. The interfaces are further defined by the AAS interface related submodels (see Figure 9).

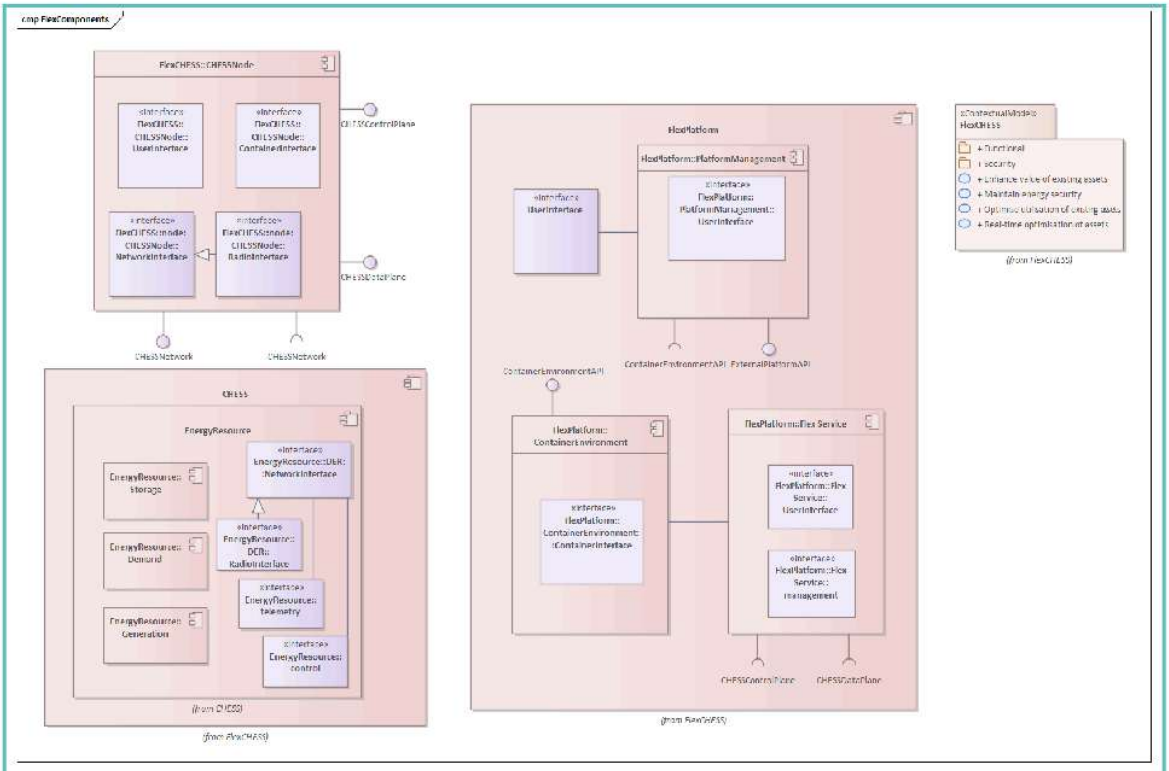


Figure 10 : FlexCHESS main components

4.2. Sequence diagrams

The following sequence diagrams illustrate the registration and configuration of flexibility services (Figure 11) and subsequent provisioning and use of the services (Figure 12).

The sequence diagrams are defined based on brainstorming sessions and meetings organized mainly with technical partners. These diagrams were defined by Toshiba and highlighted the service register process for between FS user and FS providers. Concretely, this figure presents the sequence diagram of the use case of flexibility services configuration and assets control through the physical assets to register in a specific service. In the Fig. 12, a sequence diagram highlights the steps for FS provision that include modelling, evaluation, service selection, service constraints configuration, commands, monitoring and performance evaluation and data reconciliation.

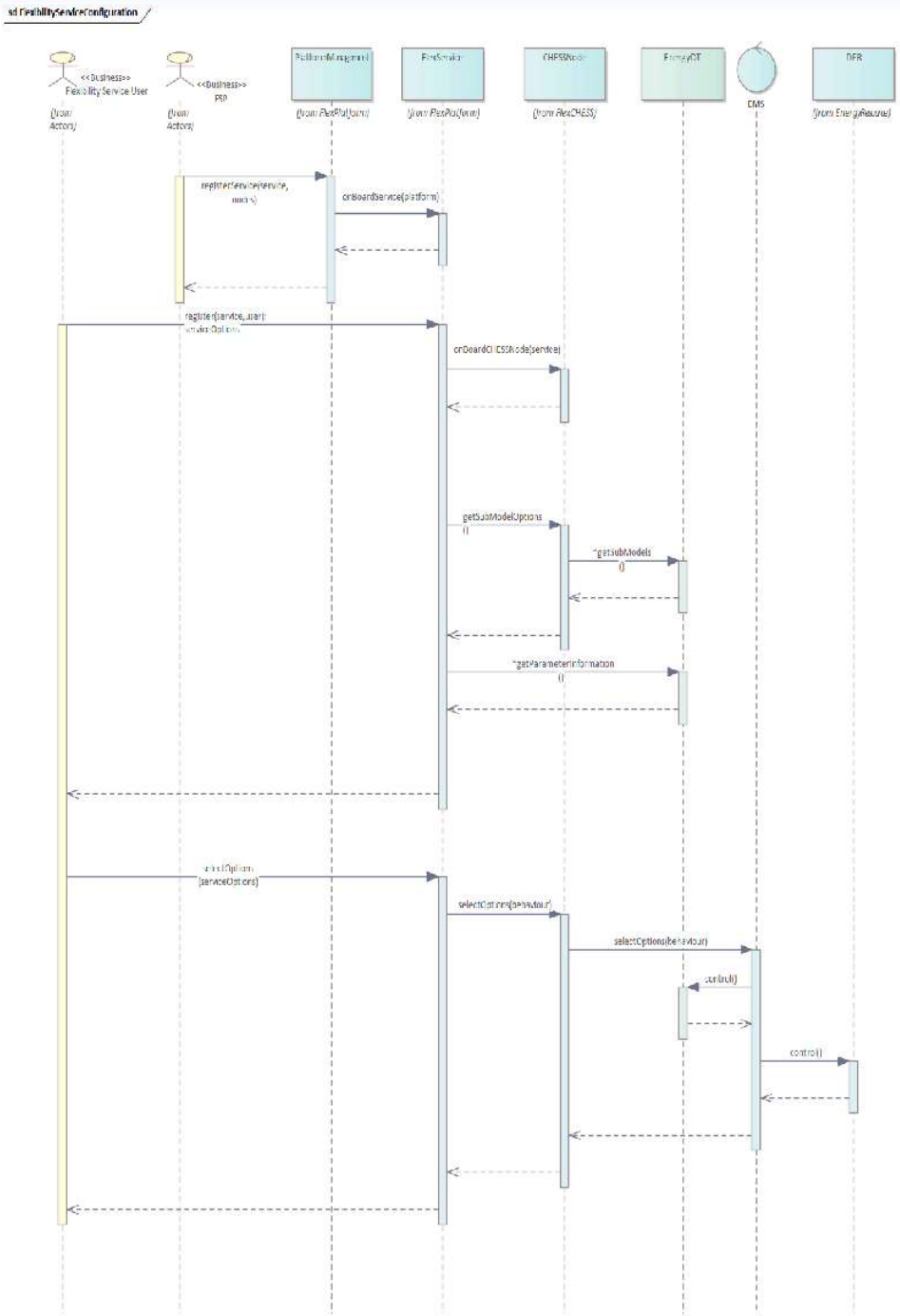


Figure 11 : Flexibility Service Configuration Sequence

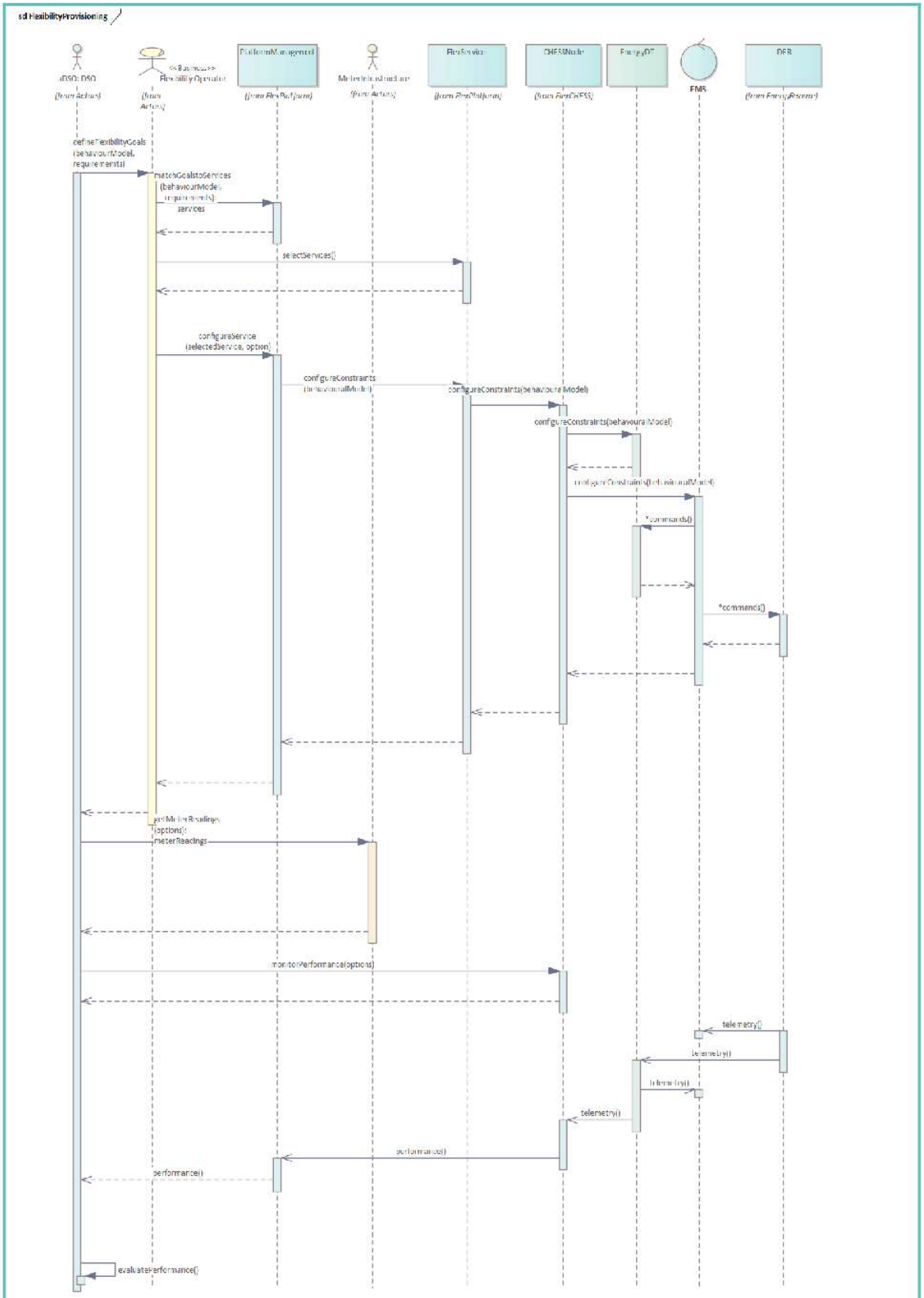


Figure 12 : Flexibility Provisioning Sequence

5. User interfaces

The Fig. provides a first specification of the CHESS node UI to illustrate the association and configuration of DERs relating to a CHESS node.

The user interfaces for interacting with the FlexCHESS platform and nodes will use existing commonly accepted approaches as necessary. For instance, the use of tools such as the AAS package explorer (Figure 14) or digital twin metamodel and graph explorers (Figure 15 and Figure 16). In addition, new UIs will be developed for management of the platform and CHESS nodes to permit association and configuration of assets (see Figure 13).

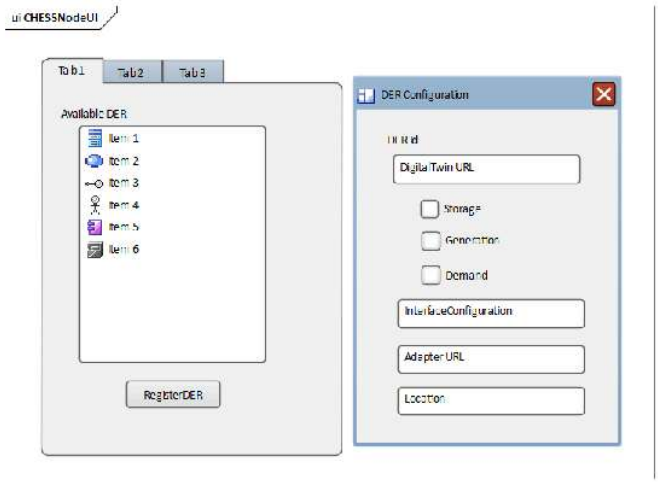


Figure 13 : CHESS node UI (CHESSNodeUI) example for associating the DERs to a CHESS node

The users will need a specific toolbox and DT explorer to browse components and elements required for behavioural or constraints models and configurations. As example we can consider the specifications of the IDTA explorer. Other examples of DT explorer can be found in AZURE.

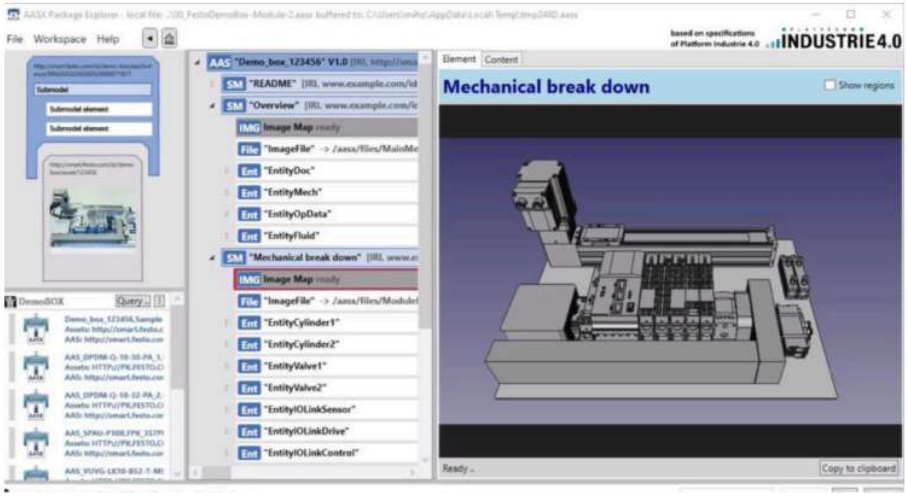


Figure 14 : AAS package explorer example from IDTA

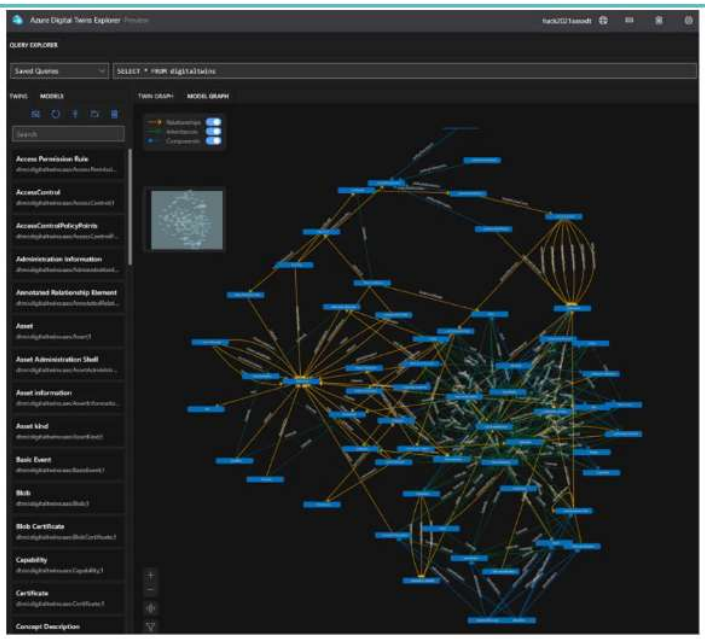


Figure 15 : AAS metamodel graph [Github]

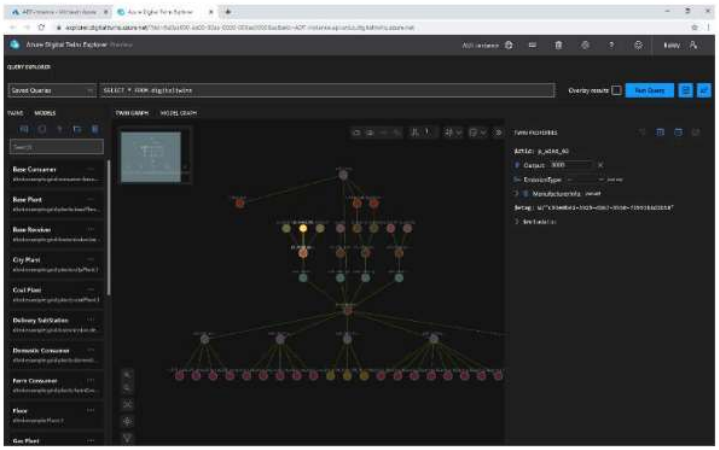


Figure 16 : Azure Digital twin graph explorer [Azure DT]

This subsection showcases the design of user interfaces, GUI, and provides an overview of the screens of dashboards to be developed for different users. The DVs below show how variables exchanged between the different modules will be visualized in the Flex-Platform. Moreover, key sketches of forms are defined by ALWA to be created and designed in T3.6. The styling of all user interfaces, dashboards, screens will be unified early-by design.

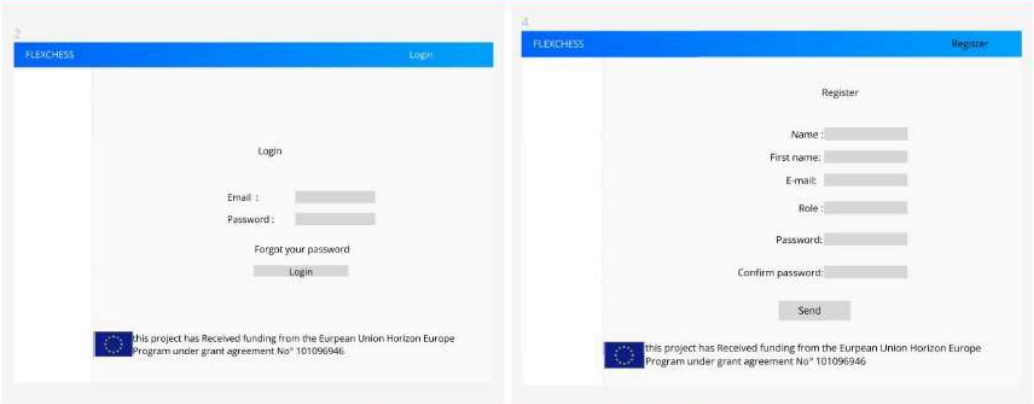


Figure 17: Authentication and register interfaces

The UIs have to include the FlexCHESS logo and website URL together with the EU logo and the Disclaimer. As shown in Fig. 17, FlexPlatfrom has to provide the possibility to signup based on a specific role, login, and reset password. Smart and easy technique of validation and confirmation should be implemented.

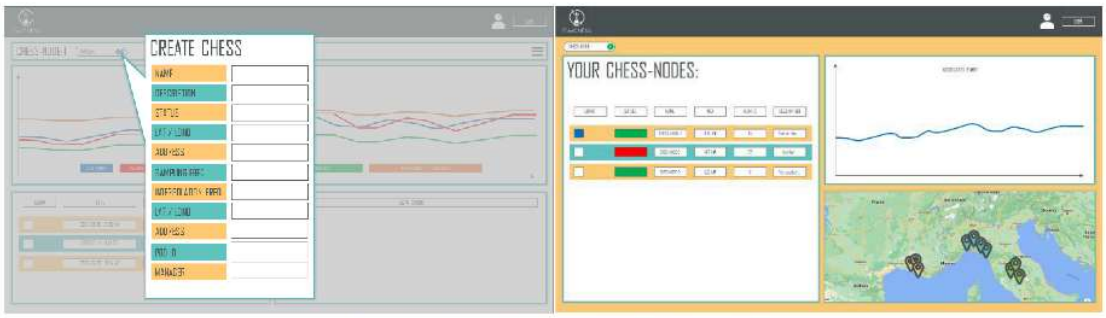


Figure 18: Forms for CHESS and CHESS Node

Specific forms will be developed to allow users to create their own CHESS. In the same direction, CHESS Node operator will be able to select and determine all information, tags and KPIs to be monitored related to their CHESS Node and Map the CHESSs composing the CHESS Node.

Figure 19: CHESS Owner's interfaces

Figure 20: CHESS Node operator's interfaces

Specific interfaces will be defined for CHESS owner and CHESS Node operators to follow and supervise their assets. Also, related features will be displayed based on their roles. Both can to measurement, reporting forms will be developed to allow users to create their own CHESS.

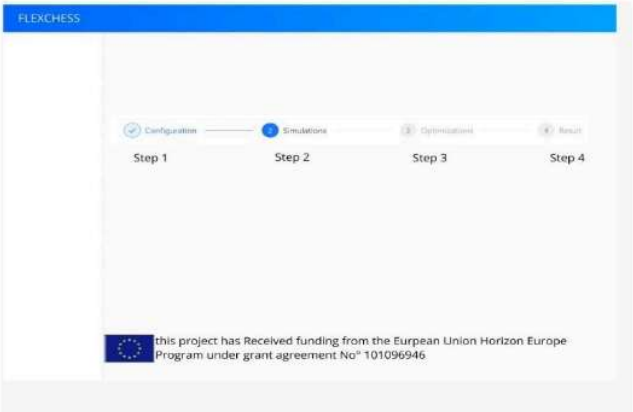


Figure 21: Interface for DSS

A specific UI will be designed for the DSS as no real-time features are needed. A "steps" layout will be implemented to configure, simulate, optimize the configuration, and report results in a specific document.

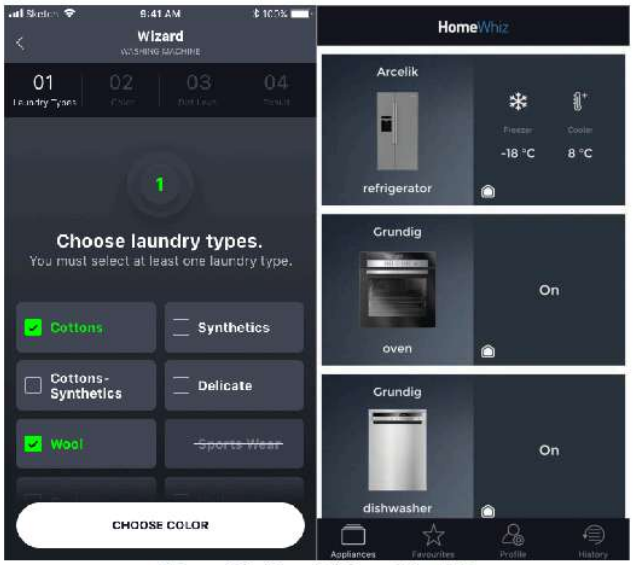


Figure 22: HomeWhiz mobile APP

Regarding the Mobile APP, HomeWhiz provided by Arçelik mainly for CHESS owner in order to follow and monitor the consumption and status of their appliances and ensure seamless communication and notification with them.

6. Data structure and list of variables

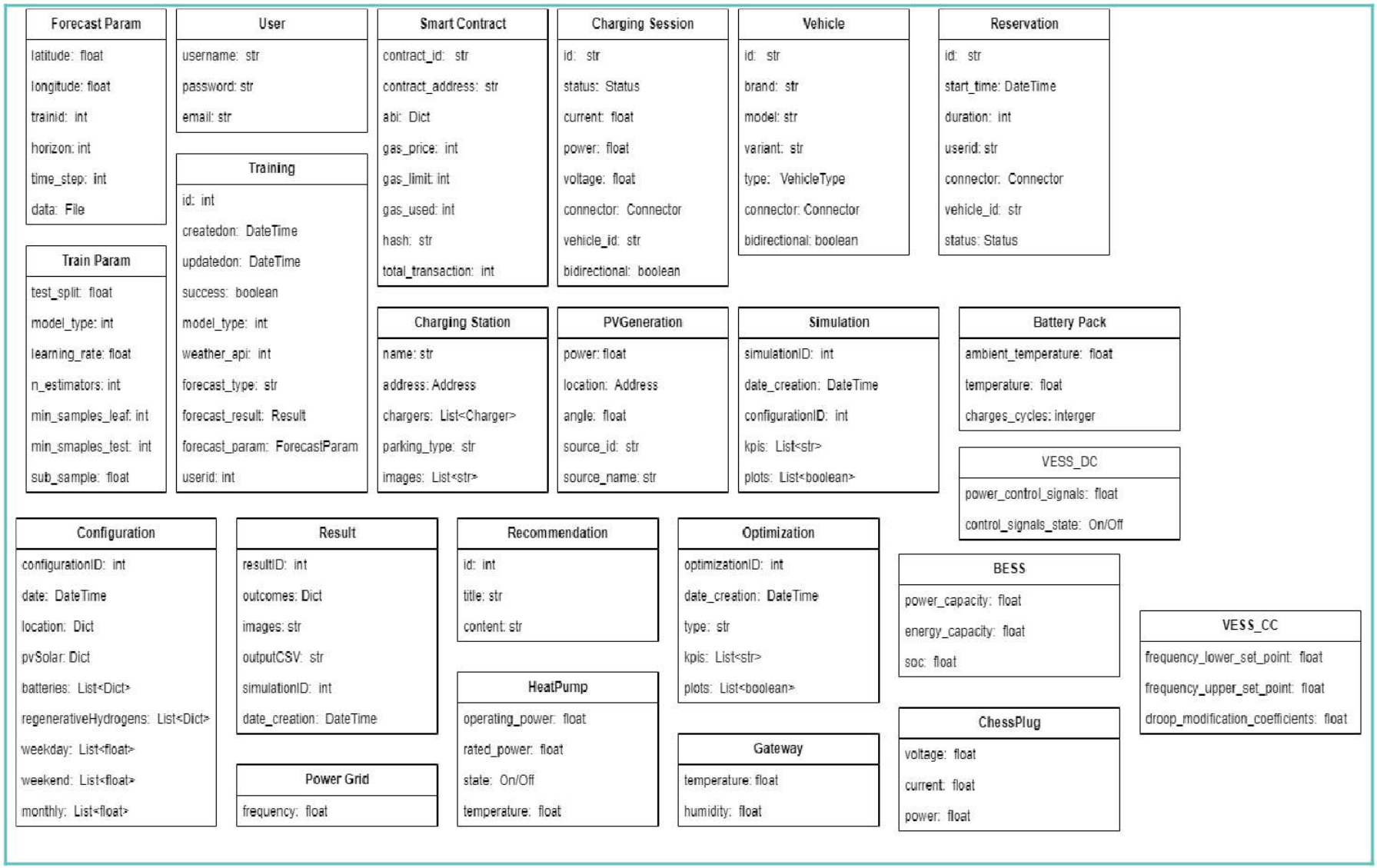
Section 6 clearly introduces the data management and protocols that are able to support the scalability, availability, modularity and usability features of the FlexCHESS solutions. Also, functional and non-functional specifications will be detailed.

6.1. Modules variables:

In order to streamline the integration process later in the project when different modules will be ready, this section of the document describes modules input, output, data models and variables that are exposed publicly to other modules. An excel sheet of variables list (see the headers below) were circulated to unify parameters between partners. The page below depicts an overview of data models and variables.

Visual element where attached	Data Field	Type	Description	Unit of Measure	Source / Destination	Protocol	Value Type	Value Range	Polling Frequency
Battery Pack	battery_soc	Calculation	Overall SOC in the ba	%	BMS	MQTT	Integer	[0..100]	0.1 s

From Gateway		From DTP		From FP			From FSP	From tools
assets	to DTP	to FP	to Gateway	to FSP	to DT	to Tools	to FP	to FP


Forecast Param
latitude: float
longitude: float
trainid: int
horizon: int
time_step: int
data: File
User
username: str
password: str
email: str
Training
id: int
createdon: DateTime
updatedon: DateTime
success: boolean
model_type: int
weather_api: int
forecast_type: str
forecast_result: Result
forecast_param: ForecastParam
userid: int
Train Param
test_split: float
model_type: int
learning_rate: float
n_estimators: int
min_samples_leaf: int
min_smaples_test: int
sub_sample: float
Smart Contract
contract_id: str
contract_address: str
abi: Dict
gas_price: int
gas_limit: int
gas_used: int
hash: str
total_transaction: int
Charging Session
id: str
status: Status
current: float
power: float
voltage: float
connector: Connector
vehicle_id: str
bidirectional: boolean
Vehicle
id: str
brand: str
model: str
variant: str
type: VehicleType
connector: Connector
bidirectional: boolean
Reservation
id: str
start_time: DateTime
duration: int
userid: str
connector: Connector
vehicle_id: str
status: Status
Charging Station
name: str
address: Address
chargers: List<Charger>
parking_type: str
images: List<str>
PVGeneration
power: float
location: Address
angle: float
source_id: str
source_name: str
Simulation
simulationID: int
date_creation: DateTime
configurationID: int
kpis: List<str>
plots: List<boolean>
Battery Pack
ambient_temperature: float
temperature: float
charges_cycles: interger
VESS_DC
power_control_signals: float
control_signals_state: On/Off
Configuration
configurationID: int
date: DateTime
location: Dict
pvSolar: Dict
batteries: List<Dict>
regenerativeHydrogens: List<Dict>
weekday: List<float>
weekend: List<float>
monthly: List<float>
Result
resultID: int
outcomes: Dict
images: str
outputCSV: str
simulationID: int
date_creation: DateTime
Power Grid
frequency: float
Recommendation
id: int
title: str
content: str
HeatPump
operating_power: float
rated_power: float
state: On/Off
temperature: float
Optimization
optimizationID: int
date_creation: DateTime
type: str
kpis: List<str>
plots: List<boolean>
Gateway
temperature: float
humidity: float
BESS
power_capacity: float
energy_capacity: float
soc: float
ChessPlug
voltage: float
current: float
power: float
VESS_CC
frequency_lower_set_point: float
frequency_upper_set_point: float
droop_modification_coefficients: float

Each box of the figure represents either an input, output, or resource data model. These data are generally made available under an API for easy communication and data exchange.

- **Forecast Param**: the forecasting param is the input of the forecasting module. It can contain different attributes such as location (latitude, longitude), trainId, horizon, time_step and data file.
- **Train Param:** As for forecast param, the train param is the input of the training module. It has many variables such as test_split, model_type, learning_rate, n_estimators, min_samples_leaf and min_sample.
- **Training:** An object or data model that is the output of training a forecasting algorithm (mainly solar pv and wind turbine). It contains information such as success status, model type, forecasting type, result, input param.
- **User:** A given user of the system. The object contains user credentials such as username, email and hashed password.
- **Battery Pack:** hold all the necessary information of the battery pack. It contains information such as ambient temperature, charge cycles, voltage, power dissipation etc.
- **Chess Plug:** Flex chess plug is a physical device that is plugged to an end user device to make it controllable. The object contains information such as voltage, power, current.
- **BESS:** for battery energy storage system, is a data model of the module responsible for batteries energy management. It contains information such as power capacity, energy capacity and state of charge (soc).
- **HeatPump:** hold all necessary information of a given heat pump of the system. It has temperature, rated power, operating power and on or off state.
- **VESS_CC:** for Virtual Energy Storage System Central Controller, contains information such as upper and lower threshold and droop modification coefficients.
- **VESS_DC:** for Virtual Energy Storage System Central Controller, contains information such as on or off control signals, power control signals.
- **Charging Station:** A data model that contains Ev charging station data such as address, chargers, images, parking type etc.
- **Vehicle:** Contains information of a given electrical vehicle such as name, model, brand, battery, connector, whether it allows bidirectional charging or not.
- **Reservation:** A reservation allows the end user to book a charging session in advance. This collection contains all information required to manage the reservation such as duration, connector, status, date etc.
- **Charging Session:** is a process of charging an Ev. As this process generates a lot of data from beginning to the end, this collection holds all the information such as power, voltage, connector, vehicle.
- **PVGeneration:** Hold information about the generation of a given solar PV generation in a given time interval such as power, source, angle location etc.
- **Smart Contract:** Hold to programmatically sign and execute a contract that can be between parts. This collection holds information on all smart contract transactions.
- **Configuration:** Contains information on assets, loads, location of the energy system. It used to make optimization and simulation.
- **Recommendation:** A best practice that has to be performed to enhance the performance of the energy system.
- **Optimization:** A data model that is the output of the optimization process that aims to find the optimal size of energy system components. It contains information such as optimization, KPIs and plot images.
- **Simulation:** Hold information derived from the simulation process that aims to simulate a given energy system for a given period.
- **Result:** A data model resulting from the simulation.
- **Gateway:** hold information such the temperature, humidity
- **Power Grid:** hold information such as frequency.

In D1.4, RDIUP, Toshiba and ALWA have provided the possible technological options for the FlexCHESS parts mainly for the computational resources and software to be developed in WP3 while considering open technologies.

As shown in the Table 6, the main common technological parts are the JSON file, Python language and MQTT. That guarantees interoperability at information and communication layers and facilitates the integration. As basically, APIs will be used to interact between modules so AZURE, HEROKU, AWS etc… will not affect or slow down the Middleware and FlexPlatform development. To avoid database harmonization and synchronisation issues, technical partners decided to consider MySQL.

Table 6: Technological options

Needs	Constraints and requirements	Smart APPs	ALWA's platform	Digital twin
Language of programming	Open source, standardized libraries, POO	Python	Python	Python (or C#, Java, Javascript)
Backend	interoperable modules (microservices)	Django	Flask	Docker Containers, Kubernetes
REST API	Ease of integration and interoperability	Django Rest Framework	Flask	AAS / OpenAPI, WSO2 API manager
User interfaces	User-friendly and ergonomics	ReactJS /	Angular	Open3D, Monogame, Digital twin explorer
Secure data storage	Security and interoperability, easy use	MongoDB/ MySQL	MySQL	PostgreSQL / InfluxDB
DATA FILE	Interoperability	JSON	JSON	JSON (or UADP)
Middleware	Rapidity and secure data distribution	MQTT	MQTT (RabbitMQ)	MQTT (or AMQP, DDS, OPC UA)
Hosting	Scalability, ease of use	HEROKU for DSS	AZURE (Not Dockerized)	AWS and AZURE for DT

6.2.Security and privacy requirements:

These requirements aim to increase awareness about data security and GDPR compliance and ensure user acceptance. Basically, personal data and privacy are parts of main concerns and has to be ensured by our FlexCHESS platform and extensions. Therefore, the main standards and regulations to be addressed are:

- Security standard ISO/IEC 27001/2022, is the international gold standard for information security management (ISM)
- ISO 27002/2022, is an information security guidelines and techniques to implement and improve ISMs.
- Privacy standard ISO/IEC 27701:2019 for certified systems including clauses linked Data subject rights.
- GDPR-compliant and the new updates such as “ePrivacy directive [IONOS]”
- prEN 17529:2020 "Data protection and privacy by design and by default", June 2020 (CEN-CENELEC)
- And a specific ISO/IEC 27019:2017 provides guidance based on ISO/IEC 27002:2013 applied to process control systems used by the energy utility industry for controlling and monitoring the production or generation, transmission, storage and distribution of electric power, gas, oil and heat, and for the control of associated supporting processes [ISO27019].

Users’ (e.g. CHESS owners) personal data will be carefully protected by these tools with ISO 27001 certified systems and stored in an encrypted database as GDPR-compliant. Also, when reports are shared with end-users, FlexCHESS put all the info collected through an aggregation and anonymization process to make sure everyone’s privacy stays protected.

In this direction, workshops should be organized between technical and pilot partners to define a whole FlexCHESS security approach early by-design through an engaging discussions and co-creative way.

At this stage, the security and GDPR compliance and security have to consider major 5 principles:

- **Data minimization**: It involves limiting the collection of relevant data to only what is necessary to fulfil a specific purpose. When implementing data minimization, any processing should use the minimum amount of data required.
- **User control**: FlexPlatform users need to be in control of their personal data. They have the right to revoke their consent at any time and remove any data from the system. That can be ensured strong AA functionalities and control of users' data.
- **Data distribution** : The distribution of a dataset refers to the shape of the graph when all possible values are plotted on a frequency graph, showing how often they occur. They are features ensuring by the middleware to distribute and broadcast between entities.
- **Data aggregation**: The technique of compiling large amounts of information from a given database and organizing it into a more comprehensive, exploitable, and explorable format. It can be applied at any scale, from pivot tables to data lakes, to summarize information and draw conclusions based on data-rich findings (middleware).
- **Data cryptography**: Cryptography provides a secure method of communication, preventing unauthorized parties (commonly referred to as adversaries or hackers) from accessing secret messages exchanged between authorized parties such as the encryption or hashing of data like password.

The requirements related to these principles have to be considered and implemented when the software and modules are developed in WP2 and WP3.

In the other hand, The European Union Agency for Cybersecurity (ENISA) is a European Union (EU) agency based in Greece that is responsible for promoting cybersecurity in the EU. ENISA provides guidance on pseudonymization techniques, which include hashing, data encryption, tokenization, anonymization, and more. Pseudonymization is a data privacy technique that involves replacing personally identifiable information (PII) with pseudonyms or false identifiers to protect individuals' privacy. It can be achieved by using techniques like hashing, tokenization, encryption, or anonymization.

Finally, a GDPR checklist has to be created and surveys have to be defined and circulated. Then, feedback will be collected from all partners and analysed to provide recommendation and best practices.

6.3. Smart Contract specifications:

The smart contract used in FlexCHESS to offer useful services and allow transparency and traceability of CHESS. These smart contracts and tokenisation mechanism to be implemented in WP2. The smart contract will be then occurred when:

- a user applies for a storage origin certificate.
- a CHESS owner accepts to join a CHESS Node which allow a fair share and transparent profits.
- a smart contract ca be used for Flexibility quantification and reconciliation to be remunerated at the right price.
- crucial data is trading based on the consent of consumers.
- flexibility service is activated to save all contract parameters (e.g., price, penalties etc ...)

The data to exchange and transactions to support has to be defined according to the useful scenarios. Then, the key components and system architecture of the Blockchain-based solutions, technologies and functionalities are defined in the following parts.

- **Truffle:** Truffle is a world-class development environment, testing framework and asset pipeline for blockchains using the Ethereum Virtual Machine (EVM).
- **Remix IDE:** Remix, more commonly known as Remix IDE, is an open-source Ethereum IDE you can use to write, compile and debug Solidity code. As such, Remix can be a hugely important tool in Web3 and dApps development.
- **Metamask:** MetaMask is a cryptocurrency wallet that enables users to store Ether and other ERC-20 tokens. The wallet can also be used to interact with decentralized applications, or dapps.
- **Ganache:** Ganache is a personal blockchain for rapid Ethereum and Corda distributed application development. Ganache can be used across the entire development cycle; enabling to develop, deploy, and test dApps in a safe and deterministic environment. Ganache comes in two flavors: a UI and CLI. Ganache UI is a desktop application supporting both Ethereum and Corda technology.
- **Web3 JS:** Web3.js is a JavaScript library that allows developers to interact with Ethereum blockchain nodes. It provides a set of APIs for connecting to a node, sending and receiving transactions, managing accounts and smart contracts, and more. Web3.js is built on top of the Ethereum JSON-RPC API, which exposes the functionality of the Ethereum node to developers. It provides an easy-to-use interface for accessing the Ethereum blockchain and developing decentralized applications (dApps).

Compiling contracts: After writing the smart contract code, it will be compiled through Remix IDE to get the abi and address variables to store them in a js file.

```
import web3 from "web/web3.js";
const address = '0xCDUAccADL/1JL99Bl69Ada2fJ6BJ180aa2bd6c50';
const abi = [
    {
        "inputs": [],
        "stateMutability": "nonpayable",
        "type": "constructor"
    },
    {
        "anonymous": false,
        "inputs": [
            {
                "indexed": false,
                "internalType": "uint256",
                "name": "orderId",
                "type": "uint256"
            },
            {
                "indexed": false,
                "internalType": "string",
                "name": "actionType",
                "type": "string"
            },
            {
                "indexed": false,
                "internalType": "enum Order.Deactivated",
```

Figure 23: Js file for abi and contract address

The abi: The Application Binary Interface (ABI) is the JavaScript method that will be used when testing and deploying the contract. Ganache can be used as a network. It offers a list of addresses with 100 ETH balance for each one which can be used as accounts in MetaMask.

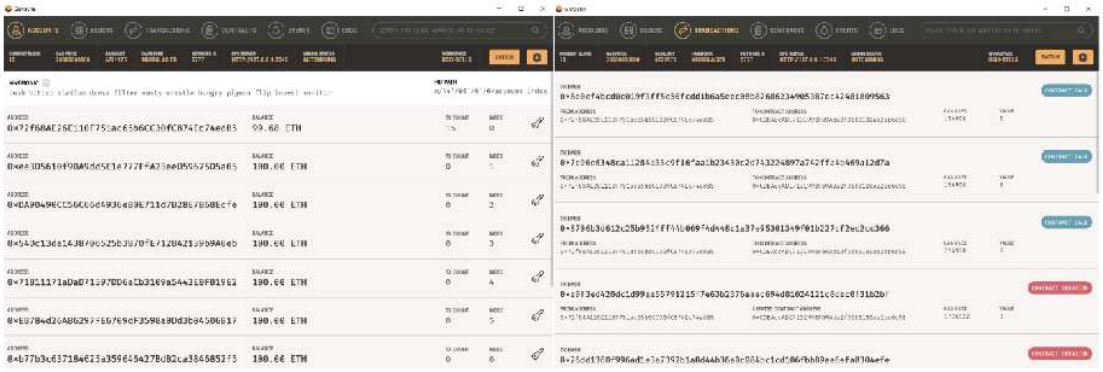


Figure 24: Ganache addresses and a list of different transactions shown by Ganache

Gas: In Ethereum, gas is a unit of measurement that represents the amount of computational effort required to execute a transaction or smart contract. Every transaction on the Ethereum network requires a certain amount of gas, and the sender of the transaction must pay for the gas in ether, the native cryptocurrency of Ethereum. Gas is used to prevent denial-of-service attacks on the Ethereum network by ensuring that each transaction is computationally expensive enough to discourage malicious actors from spamming the network. The cost of gas is determined by the current gas price, which is determined by the supply and demand of the network. Gas prices are denominated in Gwei, which is a subunit of ether, and the total gas used in a transaction is calculated as the product of the gas price and the gas limit.

The gas limit is the maximum amount of gas that a sender is willing to spend on a transaction. If a transaction exceeds the gas limit, it will fail, and the sender will lose the gas they have already spent. The gas limit is set by the sender and can be adjusted depending on the complexity of the transaction.

The amount of gas used in a transaction depends on several factors, including the complexity of the transaction, the amount of data that needs to be processed, and the amount of computation required to execute the transaction. Transactions that involve interacting with smart contracts typically require more gas than simple transactions that only transfer ether.

In summary, gas is a crucial component of the Ethereum network that ensures that each transaction is computationally expensive enough to discourage malicious actors from spamming the network. The amount of gas used in a transaction depends on several factors and can be adjusted by the sender by setting the gas limit.

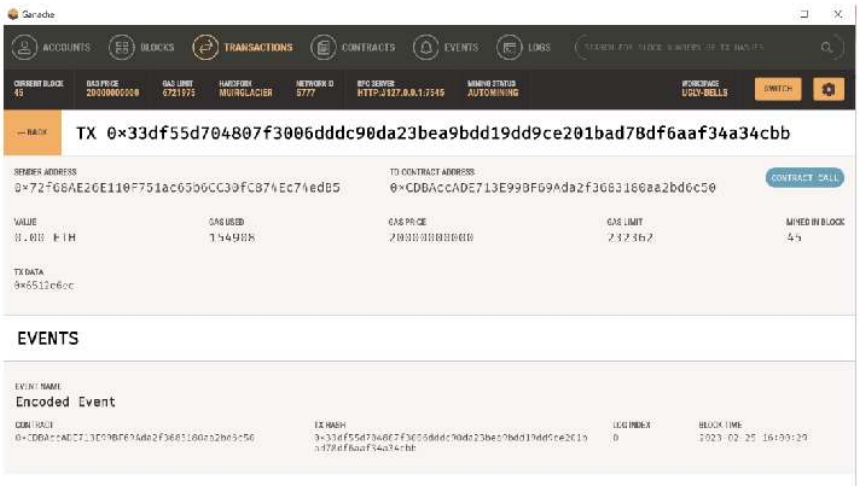


Figure 25: Example of gas costs for a transaction

7. Hardware systems :

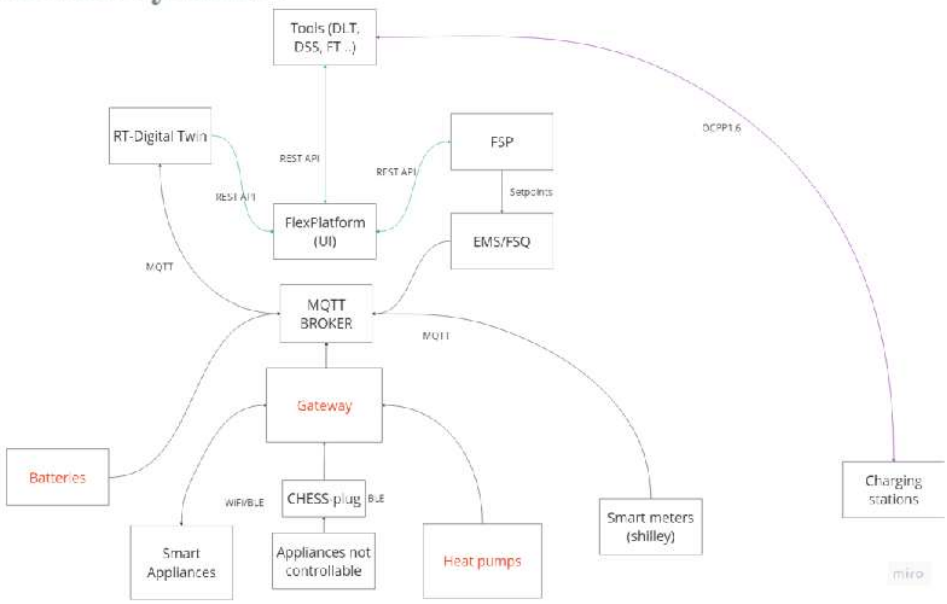


Figure 26: The communication and interactions between cloud and physical layers (FlexPlatform and CHESS)

The Fig. 26 depicts the protocols that will be used between different modules. At Cloud level, REST API will be used to exchange data between them. An MQTT broker will be provided by Arçelik to ensure the communication between FlexPlatform and CHESS. At CHESS level, the BLE mesh and Wi-Fi will ensure the communication.

On the other hand, charging stations will communicate directly with the APP extensions, without the mediation of the MQTT broker, also through Internet connections by using the OCPP protocol, which is a specific application protocol for communication between EV charging stations and a central

management system. OCPP provides interoperability, independency of system suppliers with manufacturers, flexibility, and end-to-end security.

A specific smart meter Shelly will be used as it is a MQTT compliant device.

7.1. Interoperability Specifications

Various of protocols have to be considered in the interoperable communication layer. Therefore, based on the study done in [TKI]. Although a 100% fair comparison is not possible of the protocols because of the above mentions aspects of high level vs. low level and the different types of interoperability, the following overview tries to capture the study results in a summary table. Below is given an overview of openness, interoperability and maturity when using these standards in-home connectivity (for unlocking flexibility):

	Version	Openness	Interoperability	Maturity	Registration	Energy flow reservation	Metering	Adjust capacity
ECHONET Lite	1.13	High	High	Medium/ High	●	○	○	○
EEBus SPINE	1.1.1	Medium/ High	High	Medium	●	●	●	●
EFI	2.0	Medium/ High	Medium/ High	Low	●	●	●	●
KNX	2.1	Medium	High*	High‡	●	○	●	○
OCF	2.1.0	Medium/ High	High*	Medium/ High	●	○	●	○
OCPP‡	2.0.1	Medium/ High	Medium*	Medium	●	○	●	○
SEP – IEEE 2030.5	2.0	High	Medium/ High	Medium/ High	●	●	●	●
OpenADR 2.0	1.1	High	Medium/ High*	High	●	○	●	●
Modbus ≡		High	Low	High	○	○	○	○

Table 7: Protocols comparaison (Source: TKI URBAN ENERGY)

With the introduction of the "device model" in OCPP 2.0, the protocol is more extensible. When a manufacturer uses an implementation with an extension to the protocol, it is only interoperable if other manufacturers also implement the same extension.

According to this study, two protocols have to be studied during the development phase:

- The SEP 2.0 protocol or IEEE 2030.5 standard formalizes the requirements for many aspects of the smart energy ecosystem including device communication, connectivity and information

sharing requirements. It follows a RESTful architecture utilizing widely adopted protocols such as TCP/IP and HTTP. Interoperability (CSEP) was formed by the WiFi, ZigBee, HomePlug and Bluetooth Alliances to specify certifications requirements.

- OpenADR protocol is quite generic (due to the nature of DR programs), which means that it can be used in a wide range of areas ranging from utility to EMS communication to EMS to device communication. The OpenADR alliance organizes interoperability test events, provides a testing tool and certification. Testing and certification includes several mandatory cases that are tested and certified to ensure that any client can communicate when installed and enrolled. This means that the technical interoperability is high.

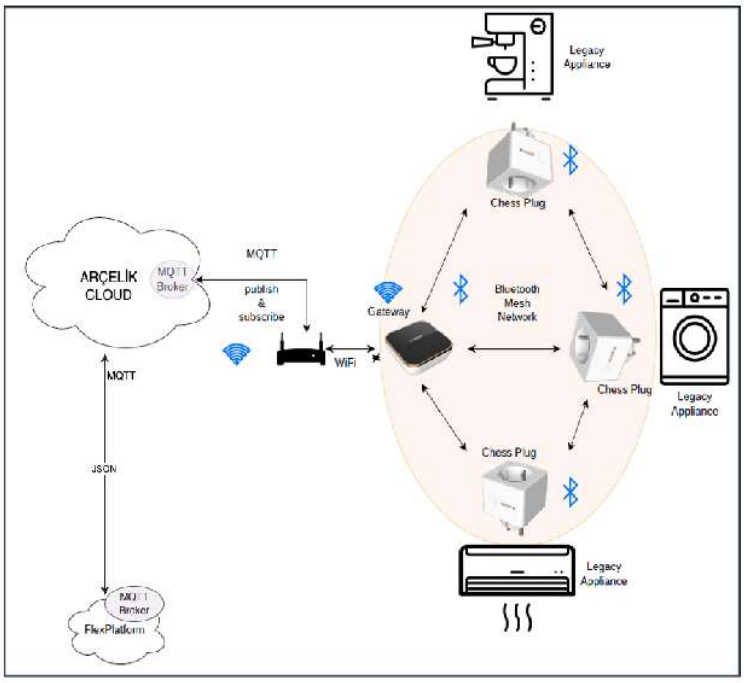


Figure 27. Integration flow.

The integration between FlexPlatform and Arçelik relies on the MQTT protocol for communication. Data is exchanged via specific topics, with Arçelik Cloud publishing data and subscribing to commands. Chess-Plug gateways facilitate this integration, transmitting temperature, humidity, voltage, current, and power data.

Communication occurs via MQTT and Bluetooth Low Energy, with Bluetooth Mesh Network for extended range. This integration enables device control and data sharing, supporting efficient management in pilot buildings. The integration flow is depicted in Fig. 1 and more information is provided below.

FlexPlatform and Arçelik must be communicated over the MQTT (Message Queuing Telemetry Transport) protocol. An MQTT broker must be deployed within the FlexPlatform and required topics for collecting data and sending commands are configured.

Such as: For data collection DEVICE_ID/data and For commands DEVICE_ID/commands

Arçelik Cloud publishes data to a data topic and FlexPlatform subscribes to that topic. On the other side, Arçelik Cloud subscribes to commands topics, and FlexPlatform is published to that topic. Based on that Arçelik Cloud controls devices and shares data to FlexPlatform.

7.2. Chess-Plug/Gateway Specifications

Arçelik Cloud has an MQTT broker and topics for collection of data and device controls must be configured. Such as:

- For data collection GATEWAY_ID/PLUG_ID/data
- For commands GATEWAY_ID/PLUG_ID/commands

The gateway (see Appendix A) has built-in temperature and humidity sensors. The measuring range of the humidity sensor built into the gateway is between 5 and 95. Its unit is percent. The measurement resolution is 0.1. Data is transmitted if there is any change in the measured value.

The measuring range of the temperature sensor built into the gateway is between 0 and 45. Its unit is degrees Celsius. The measurement resolution is 0.1. Data is transmitted if there is any change in the measured value.

It has WIFI and Bluetooth modules and communicates with Arçelik Cloud via MQTT. Gateway must publish temperature and humidity measurements to the data topic and must subscribe to the command topic for controlling Chess-Plugs.

Chess-Plug can measure instant voltage, current, and power. The current value supported by the Chess-Plug is between 0 and 15. The measurements are transferred to Arçelik Cloud via the gateway. It communicates with the gateway via Bluetooth Low Energy (BLE) and it supports Bluetooth Mesh Network to extend the range of connections from gateways with multi-hop communication.

Chess-Plugs will be used to turn the electricity on and off for each of the devices and appliances within the pilot buildings. The signal for that decision will be fired from the FlexPlatform with MQTT messages depending on the demand response analysis for a specific area. The technical hardware specifications of the gateway and Chess-Plug are provided in Appendix A.

7.3. SKYcamera

The main requirement of the skycamera is the low-cost, the resolution of images and the latency time. It is useful to know the sky conditions at local level in order to provide a short-term forecasting. Thus, it is good to know the accuracy of weather forecasts for mainly PV generation or building consumption. Here are the key technical requirements:

- 180 degree (all sky) coverage.
- Weatherproof - it is very important to prevent water intrusion.
- Heat tolerant (will be roof mounted).
- Capture still images for time-lapse video.
- All images tagged with time, temperature and humidity (inside the enclosure).
- Connected to building network by Ethernet.
- The camera must be completely automated, producing a time-lapse video every 24 hours.

Some of these requirements will be met by the hardware, others by software. Open scripts and Python will be used for the software (e.g. the https://github.com/ghethco/AllSkyCamera).

In order to reduce the costs of the skycamear, open technologies will used such as We are using a Raspberry Pi 4B with the HQ camera (at least 2 MP). Some Raspberry Pi compliant HQ camera are available like IMX477 with a 12.3 MP resolution that can be connected to a 16mm telephoto lens. Wi-

Fi or ethernet can be used for communication or for powering the cards. All electronics can be powered via 5V USB-C. Easily, temperature and humidity sensors should be added (e.g. DHT11 sensor).

Regarding the enclosure, it has to be weatherproof, and easily opened and closed which facilitate the manipulation of camera, raspberry Pi and sensors.

Two main issues to be considered when creating the skycamera are the overheating inside the box and the cleaning of the transparent dome to keep clear images.

7.4. Equipment and laboratory facilities

The smart platform of interconnected distributed energy resources -SPIDER-is a lab facility under AMU. The SPIDER is designed in a modular way and consists of various equipment such as programmable DC power supply (Cinergia B2C-15 and EA-PS 9080-100 1U), programmable AC power supply (Cinergia GE&EL-15), Power converters (Imperix make), passive elements and real-time controller (OP4510, HIL404 and dSpace). These equipment are utilized to create a prototype of real-time systems or validate the various scenario of power electronics and power systems. The SPIDER also provides the high-performance personal computers (PCs) to the researchers to work on high computing problems and complex tasks.

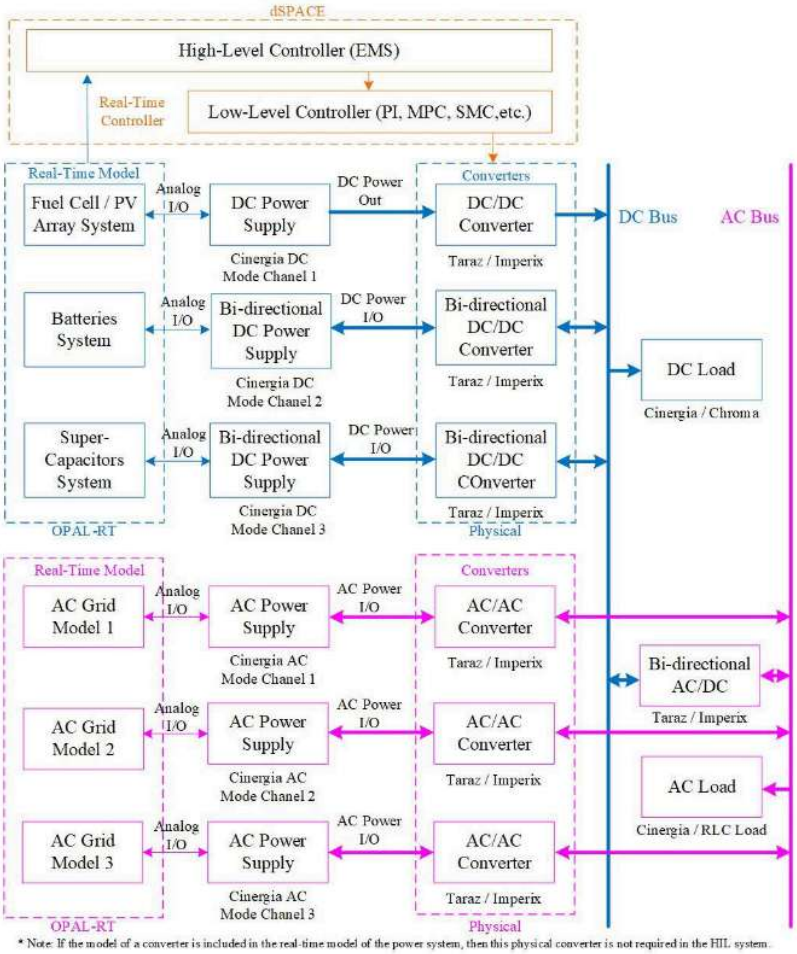


Figure 28: Diagram and I/O exchange

The charging station setup at the SPIDER is hybrid setup for the charging electric vehicles (EVs) using renewable energy sources, grid and diesel generator. The main component of this setup is the cinergia B2C-15 as PV emulator, Lithium-ion batteries as an energy storage, Cinergia GE&EL-15 as a grid simulator, power converter (inverter) and their control (MPPT and inverter control). Additionally, a personal computer is also provided for the operation and monitoring. The main function of this setup is to charge the EVs through PV. The reliability of the system is improved by adding the energy storage in case the PV is not present. The energy Furthermore, in this setup, grid and diesel generator are also used to provide the un-interrupted supply for charging the energy storage. Afterward, the setup is used to charge the EVs and house loads. The main purpose of this setup at the platform is to analyse the data for the charging of Li-ion batteries and EV batteries specially using parameters like state-of-charge (SoC), Charging-rate (C-rate) and state-of-health (SoH). Additionally, to study the use of energy storage as flexible load for the frequency and voltage regulation to provide the ancillary services.

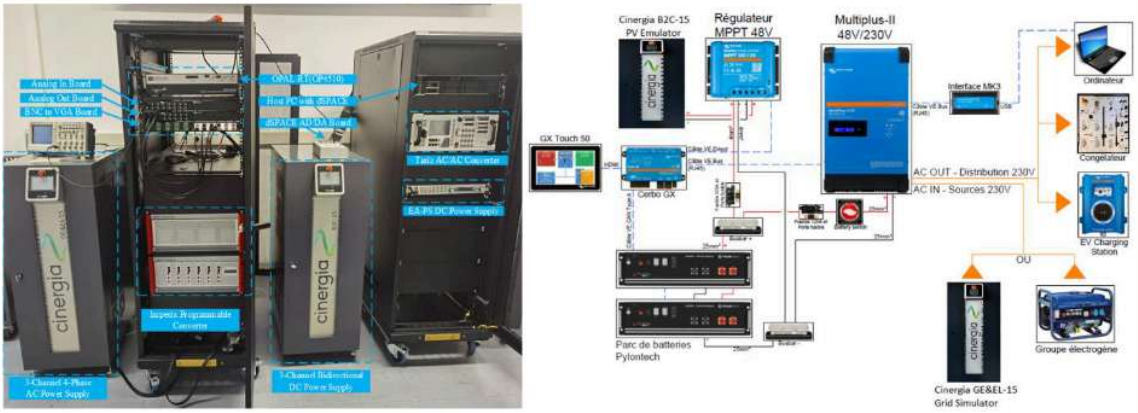


Figure 29: The architecture of the test bend

The SPIDER has software testing facilities in other words it provides an environment to verify the correctness and robustness of a design or model. This platform has various equipment such as OP4510, dSpace, HIL402 and B-Box which helps to justify and validate the software based design of a model or architecture. For example, a MATLAB based design can be tested in software-in-the-loop (SIL) using aforementioned devices. They are also capable to interface using Modbus and CAN protocol. This application of the devices can be used to test software correctness and robustness for eMobility and powertrain applications.

8. Technical specifications of VESS frequency control scheme

A Virtual Energy Storage System (VESS) is a combination of different controllable distributed flexibility resources, which can be operated in a coordinated way to mimic the operation of a large-capacity conventional energy storage system. Different distributed flexibility resources could be considered within the VESS demonstration area, such as Battery Energy Storage systems (BESS), clusters of Electric vehicles (EV), renewable energy resources and flexible demand units (e.g. buildings with electric heat pumps).

A frequency control scheme of the VESS will be developed to provide ancillary services to the Power system operator in Great Britain (GB). Recently, the GB system operator started procuring new services to balance the grid frequency in real time, such as Dynamic Regulation (DR), in addition to the traditional frequency response services. Through these services, a flexibly aggregator, e.g. a VESS operator, can provide frequency support to the power system and remunerate accordingly.

The proposed frequency control scheme of the VESS consists of two levels of control, as shown in Figure 1. At the VESS level, a central control will be coordinating the operation of the VESS components, to ensure that their aggregated response is equal to the required value. At VESS components level, distributed controllers will regulate the VESS components operation in response to frequency deviations. Serval priorities will be considered by the central control, such as the life span of the BESS, the travel patterns of EVs drivers and the comfort levels of flexible demand users.

The hierarchical frequency control scheme will facilitate the fast and accurate response of the VESS components to frequency deviations without the need for the expensive real-time communication system between a central entity and the components. Instead, the communication signals between the central control and distributed controllers will take place every few minutes. The communications between the two levels of control will also allow the VESS operator to quantify the available flexibility from the VESS components.

The proposed frequency control scheme can activate only a part of the VESS available capacity to provide ancillary services to the transmission system operator and enable the remaining capacity to be used to provide other services to a distribution system operator for instance. The control scheme also allows the VESS operator to deliver different flexibility capacities at different times of the day.

Different distributed controllers will send signals (i.e. Measurements in Figure 30) that contain the measurements related to their controlling devices, e.g. State of Charge of BESS, every few minutes. The central controller will use this information to create and send several setting points (i.e. Setting points in Figure 30) to the distributed controllers, which used it to determine the instantaneous response to frequency deviations from the VESS components connected to it. As different distributed frequency controllers will be developed for different devices, different setting points will be sent to update it at different time steps.

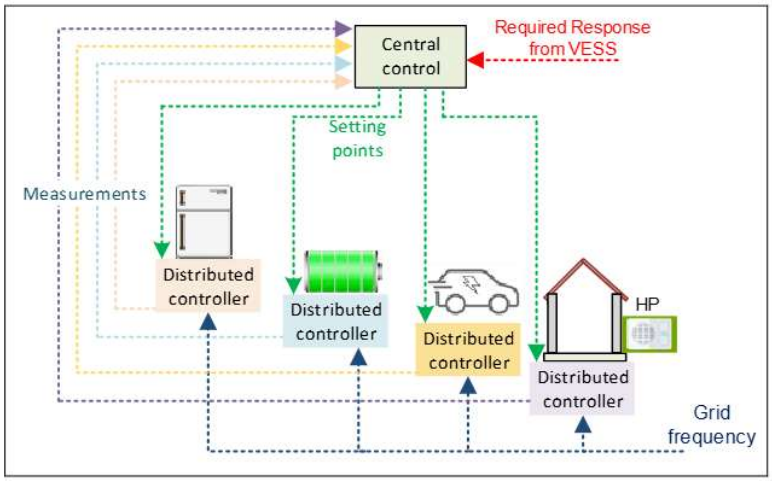


Figure 30: Frequency Control scheme of the VESS.

9. Conclusions:

This document D1.4 contains the preliminary cloud-based software specifications for the FlexCHESS solutions, which is fulfilled as part of WP1, Task 1.4, of the Work Plan. The report describes the design and architecture to serve as a reference and guide for the implementation of other tasks in WP2 and WP3. It starts with a general architecture of all software systems in the cloud, and the goes into the detail of the FlexPlatform and its main tools: DT, FSP, EMS, DSS, FT and KB

Also, the hardware system architecture and requirements were defined and specific for the CHESS-Plug, Gateway, skycamera and the interoperable communication layer.

All objectives in this deliverable were achieved, the outcomes of D1.4 will feed almost of tasks to develop the different modules and devices. Certainly, further tasks may bring changes in some of the specifications that are described in this document.

References

- CRUD: « CRUD And Resourceful Routing - Job Board » [archive], sur *docs.railsbridge.org*
- IONOS: https://www.ionos.fr/digitalguide/sites-internet/droit-dinternet/le-reglement-eprivacy-projet-de-lue/
- AAS: "A short introduction to properties, submodels & Asset Administration Shells (AAS)": https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Hardshell-softcore_ppt.html
- Github: GitHub - JMayrbaeurl/opendigitaltwins-assetadminstrationshell: Provide a DTDL v2 implementation of the Industry 4.0 Asset Administration shell ontology
- Azure DT: https://learn.microsoft.com/en-gb/azure/digital-twins/overview
- ISO27019: https://webstore.iec.ch/preview/info_isoiec27019%7Bed1.0%7Den.pdf
- TKI: "IN-HOME ENERGY FLEXIBILITY PROTOCOLST" TKI URBAN ENERGY 2020

Appendix A

GATEWAY	
Connection	Wireless Home Connection (2.4 GHz)
	BLE Mesh Connection (2.4 GHz ISM)

Dimension	70.1 x 70.1 x 21.6mm
Power Adapter	ATECH OEM INC/ADS012T-B050200
Input Voltage/Frequency/ Current	100-240V~ / 50-60Hz / 0.5A
Output Voltage/Current/Power	5.0V / 2.0A / 10.0W
Average Active Productivity	79,114%
Low power (10%) efficiency	86,677%
No Load Power Consumption	0.039W
Operating temperature / humidity	0°C – 45°C/5%~95% IP30
Material	Plastic & Aluminum
	Insulation Class -II
Additional information	There are internal Temperature & Humidity sensors inside the device.
	Depending on the environment, it may take approximately 6 hours for the Temperature and Humidity values to be measured correctly.
* 802.11b/g/n 2.4Ghz supported	

CHESS PLUG	
Connection	BLE Mesh Connection (2.4 GHz ISM)
User interface	1 LED Notification Light (Color)
	1 Reset Button (Hole])
Dimension	57 x 57 x 78.6 mm
Power source	<1W
Power consumption	100-240V~ / 50-60Hz / 0.5A
Input Voltage	100 - 240 VAC
Operating temperature/humidity	0°C – 45°C/5%~95% IP20
Material	Plastic & Aluminum

	Insulation Class -II
Additional information	It also provides instant Power consumption, Current and Voltage information. Maximum current value 15 A

FlexCHESS

Contact

contact@flexchess.eu

Aix Marseille Université

LIS UMR 7020 CNRS

Campus de Saint Jérôme – Bat. Polytech

52 Av. Escadrille Normandie Niemen

13397 Marseille Cedex 20

https://www.flexchess.eu/

@FlexchessHorizonEU

www.linkedin.com/company/flexchess/

https://twitter.com/ChessFlex

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No°101096946