



Deliverable 3.1: DEDALUS technological enablers for multi-value and energy carrier-agnostic residential DR (1st technology release)

D3.1: DEDALUS technological enablers for multi-value and energy carrier-agnostic residential DR (1st technology release)
WP 3, all tasks



This project has received funding from European Union's Horizon Europe Research and Innovation programme under the Grant Agreement No 101103998

Technical references

Project Acronym	DEDALUS
Project Title	Data-driven Residential Energy Carrier – agnostic Demand Response Tools and Multi-value Services
Project Coordinator	Diego Arnone ENG diego.arnone@eng.it
Project Duration	May 2023 – June 2026 (36 months)
Deliverable No.	D3.1
Dissemination level*	PU
Work Package	WP 3 - Technology Enablers for residential energy consumer data-driven 'activation'
Topic	HORIZON-CL5-2022-D4-01-01
Funding Scheme	HORIZON Innovation Actions
Lead beneficiary	3 (CARTIF)
Contributing beneficiary/ies	CARTIF, ENG, NTUA, ARC, COMS, OURPOWER, GREENPOINT, BLUEPRINT, FAELLESBO, NEOGRID, HERON, DOM-X, ISRAA, DYNAME, CINMNE, PEUSA, ROMUR
Due date of deliverable	30 April 2024
Actual submission date	01 May 2024
Authors	Susana M ^a Gutiérrez (CARTIF), José L. Hernández (CARTIF), Marzia Mammina (ENG), Luca Vitale (ENG), Ionut Anghel (TUC), Klemen Bregar (COMS), Giannis Papias (NTUA), Tuğçe Pinar (ARC), Mehmet Ali Akyol (ARC)
Reviews	Riccardo Naccarelli (UNIVPM), Sara Casaccia (UNIVPM), Ionut Anghel (TUC)
Keywords	technological enabler, residential, demand response, flexibility, first release, architecture, protocols, data models, flexibility modelling, privacy-preserving, blockchain

PU: Public — fully open. SEN: Sensitive — limited under the conditions of the Grant Agreement. EU classified (EUCI) under Decision 2015/444: RESTREINT-UE/EU-RESTRICTED (R-UE/EU-R), CONFIDENTIEL-UE/EU-CONFIDENTIAL (C-UE/EU-C), SECRET-UE/EU-SECRET (S-UE/EU-S), RE = Restricted to a group specified by the consortium (including the Commission Services)



Who We Are

	Participant Name	Short Name	Country Code	Logo
1	ENGINEERING - INGEGNERIA INFORMATICA SPA	ENG	IT	
2	ETHNICON METSOVION POLYTECHNION	NTUA	EL	
3	FUNDACION CARTIF	CARTIF	ES	
4	UNIVERSITA POLITECNICA DELLE MARCHE	UNIVPM	IT	
5	ARCELIK A.S.	ARC	TR	
6	COMSENSUS	COMSENSUS	SI	
7	OURPOWER ENERGIEGENOSSENSCHAFT SCE MIT BESCHRANKTER HAFTUNG	OURPOWER	AT	
8	GREEN-POINT 62 GMBH	GREENPOINT	AT	-
9	BLUEPRINT ENERGY SOLUTIONS GMBH	BLUEPRINT	AT	
10	FAELLESBO	FAELLESBO	DK	
11	NEOGRID TECHNOLOGIES APS	NEOGRID	DK	
12	EUROPEAN GREEN CITIES APS	GREEN CITIES	DK	
13	HERON SINGLE MEMBER S.A. ENERGY SERVICES	HERON	ELX	
14	DOMX IDIOTIKI KEFALAIOUCHIKI ETAIREIA	DOM-X	ELX	



15	ISTITUTO PER SERVIZI DI RICOVERO E ASSISTENZA AGLI ANZIANI	ISRAA	IT	
16	DYNAME SRL	DYNAME	IT	
17	CENTRE INTERNACIONAL DE METODES NUMERICS EN ENGINYERIA	CIMNE	ES	
18	PRODUCTORA ELECTRICA URGELLENCA	PEUSA	ES	
19	ROMUR RENOVALBES SL	ROMUR	ES	
20	CENTRICA BUSINESS SOLUTIONS BELGIUM	CENTRICA	BE	
21	UNIVERSITATEA TEHNICA CLUJ-NAPOCA	TUC	RO	
22	SMART INNOVATION NORWAY AS	SIN	NO	
23	FONDAZIONE ICONS	ICONS	IT	



Document History

V	Date	Description	Author
0.1	16/01/2024	ToC	Susana M ^a Gutiérrez (CARTIF)
0.2	26/01/2024	Reviewed ToC	All the involved partners, and also peer reviewers.
0.3	22/03/2024	First contributions to the deliverable	Susana M ^a Gutiérrez (CARTIF), José L. Hernández (CARTIF), Marzia Mammina (ENG)
0.4	26/03/2024	“Consolidated version” with ENG and CARTIF contributions	Susana M ^a Gutiérrez (CARTIF), José L. Hernández (CARTIF), Marzia Mammina (ENG)
0.5	27/03/2024	New version with more contributions from partners	Luca Vitale (ENG), Ionut Anghel (TUC), Susana Gutiérrez (CARTIF)
0.6	01/04/2024	Deliverable with the contributions from all the partners	John Papias (NTUA), Tuğçe Pinar (ARC), Mehmet Ali Akyol (ARC), Klemen Bregar (COMS), Susana Gutiérrez (CARTIF)
0.7	02/04/2024	Deliverable ready for peer reviewing (including executive summary and conclusions sections)	Susana Gutiérrez (CARTIF)
0.75	09/04/2024	UNIVPM and TUC peer-review	Sara Casaccia (UNIVPM), Riccardo Naccarelli (UNIVPM), Ionut Anghel (TUC)
0.8	10/04/2024	New version of the document, addressing the comments from the reviewers, and ready to be “approved” by all the involved partners	Susana Gutiérrez (CARTIF), John Papias (NTUA)
0.9	16/04/2024	Final version of de the deliverable, ready to be sent to the coordination team	Susana Gutiérrez (CARTIF)
0.99	30/04/2024	Quality check	Diego Arnone (ENG)
1.0	30/04/2024	Final version (ready for submission to the EC Portal)	Diego Arnone (ENG)

Disclaimer

The sole responsibility for the content of this publication lies with the authors. It does not necessarily reflect the opinion of the European Union. Neither the CINEA nor the European Commission is responsible for any use that may be made of the information contained therein.



Table of contents

1. Introduction	15
1.1. Purpose and audience of the document.....	15
1.2. Relation to other activities.....	15
1.3. Structure of the document	15
2. WP3 and the DEDALUS framework architecture.....	17
3. DR-ready smart device & flexible resource data collection – 1st technology release	19
3.1. ARCELIK's IoT Platform description.....	19
3.2. ARCELIK IoT platform deployment in the DEDALUS project.....	22
3.3. Next steps.....	22
4. Energy Data Space compliant interoperability layer – 1st technology release.....	23
4.1. FIWARE-based Interoperability Solution.....	23
4.1.1. FIWARE Context Broker.....	25
4.1.2. NGSI Agents.....	28
4.1.3. Smart Data Models	30
4.2. International Data Space (IDS).....	31
4.2.1. Data Connector	33
4.3. Interoperability in DEDALUS	36
5. DEDALUS flexibility and DR data models and standards – 1st technology release	40
5.1. Existing ontologies.....	40
5.1.1. Smart Applications REference Ontology (SAREF).....	41
5.1.2. OpenADR.....	44
5.1.3. Brick	45
5.1.4. domOS Common Ontology (dCO)	46
5.1.5. BIGG Project Ontology.....	49
5.1.6. Comparison and final decision.....	50
5.2. DEDALUS specific data requirements	51
5.3. DEDALUS ontology.....	54
5.4. Next steps.....	56
6. Multi-energy flexibility modelling for data-driven residential DR – 1st technology release	58
6.1. Overview	58
6.2. General Architecture.....	58



6.2.1.	Building-Level Analysis:.....	59
6.2.2.	Device-Level Analysis:.....	59
6.3.	Building-Level Flexibility Modelling Description	60
6.3.1.	Building-Level Electricity Consumption Modelling Description	62
6.3.2.	Building-Level Electricity Production Modelling Description	63
6.3.3.	Calculation of Flexibility	64
6.3.4.	Visualization and Analysis	64
6.4.	Next Steps	70
7.	Blockchain-based interoperability for flexibility tokenization & DR settlement – 1st technology release	72
7.1.	General architecture.....	72
7.2.	DEDALUS ontology compliant data model	73
7.2.1.	Building description	75
7.2.2.	Core and measurements.....	75
7.2.3.	Devices	75
7.2.4.	Thing description.....	76
7.2.5.	Data model API.....	76
7.3.	Data Ingestion Component.....	82
7.4.	Energy Forecasting Component	83
7.5.	Building Flexibility Trading.....	85
7.5.1.	The blockchain network.....	86
7.5.2.	Flexibility Tokenization.....	87
7.5.3.	Payments Tokenization.....	89
7.5.4.	Building Flexibility Decentralized Management.....	90
7.5.5.	Flexibility Aggregation and Settlement	99
8.	Privacy, security and data protection framework for trusted data sharing and AI	102
8.1.	Universal Secure Gateway.....	102
8.1.1.	USG Software-Only Architecture.....	103
8.2.	DEDALUS' privacy, security and data protection framework	109
8.2.1.	Privacy Protection Enforcement.....	111
8.2.2.	PPE Component Architecture	113
8.3.	Next steps.....	116
Conclusions.....		117
References.....		118



List of Tables

Table 1: Ontologies summary (pros&cons)	50
Table 2: DEDALUS data requirements summary	53
Table 3: dCO extensions needed	54
Table 4 Electricity Consumption Prediction Performance	63
Table 5: Insert DR Platform users API	77
Table 6: Insert community API	78
Table 7: Configure spaces, apartments, and buildings API	78
Table 8: Configure sensors API	79
Table 9: Insert Monitored Values and get input for training and prediction API	80
Table 10: Insert Estimated Profile API	81
Table 11: Forecasting component API	84
Table 12: Matching algorithm API endpoint	100
Table 13: Privacy and data protection requirements.	110

List of Figures

Figure 1: DEDALUS framework architecture	18
Figure 2: Simplified architecture of Arçelik HomeWhiz Platform	20
Figure 3: The communication architecture between Arçelik HomeWhiz Platform and Pilot Sites	22
Figure 4: Reference architecture for Smart Energy management solutions “POWERED BY FIWARE”	24
Figure 5: FIWARE platform high-level architecture	25
Figure 6: NGSI data model	26
Figure 7: UML representation of NGSI-LD information model	27
Figure 8: Orion Context Broker ¹⁰	28
Figure 9: FIWARE Backend IoT Device Management	29
Figure 10: NGSI interactions an IoT Agent makes with the context broker	30
Figure 11: IDS Reference Architecture	32
Figure 12: TRUE Connector architecture	34
Figure 13: FIWARE TRUE Connector ²⁹	34
Figure 14: OneNet Connector Architecture - v1 ³⁰	36
Figure 15: PoC of the DEDALUS interoperability tool	38
Figure 16: FIWARE Orion Context Broker reading and writing	39
Figure 17: General overview of the top levels of SAREF4BLDG	42
Figure 18: Overview of the SAREF4SYST ontology (source)	43
Figure 19: SAREF4CITY overview (source)	44
Figure 20: Top-level view of the openADR ontology (source:)	45
Figure 21: domOS ontology: main classes and properties	48
Figure 22: The BIGG ontology	49
Figure 23: DEDALUS ontology (dCO ontology (source) and the additions due to DEDALUS’ specific needs)	56



Figure 24: Workflow of the proposed methodology.....	59
Figure 25: Workflow of the building-level methodology	60
Figure 26: Performance of the LSTM model for energy consumption	62
Figure 27: Comparison between Actual and Predicted Energy Consumption	63
Figure 28: Baseline and Min/Max Flexibility for Energy Consumption for a weekday.....	65
Figure 29: Baseline and Min/Max Flexibility for Energy Consumption for a weekend day.....	65
Figure 30: Overview Graph for Energy Consumption for a weekday	66
Figure 31: Overview Graph for Energy Consumption for a weekend day.....	66
Figure 32: Baseline and Min/Max Flexibility for Energy Production for a weekday	67
Figure 33: Baseline and Min/Max Flexibility for Energy Production for a weekend day.....	67
Figure 34: Consolidated Graph for both Energy Consumption and Production for a weekday..	68
Figure 35: Consolidated Graph for both Energy Consumption and Production for a weekend day	68
Figure 36: Comprehensive Graph combining both Energy Consumption and Production along with Predictions for a weekday	69
Figure 37: Comprehensive Graph combining both Energy Consumption and Production along with Predictions for a weekend day.....	70
Figure 38: Architectural overview of BIFTS.	73
Figure 39: BIFTS Data Model	74
Figure 40: Data model API architecture.	77
Figure 41: Data ingestion mechanisms.....	82
Figure 42: Forecasting tool – General flow.	83
Figure 43: Forecasting component – dedalus-forecasting-models integration architecture.	84
Figure 44: Interaction flow for building flexibility aggregation	86
Figure 45: ERC721 energy flexibility smart contract.....	88
Figure 46: Lockable ERC20 methods.....	90
Figure 47: LockableERC20 definition.....	90
Figure 48: Apartment smart contract model.....	94
Figure 49: Building smart contract model.....	96
Figure 50: Trading session smart contract model	98
Figure 51: Greedy algorithm pseudocode for decentralized matching of orders.....	100
Figure 52: Settlement smart contract	101
Figure 53: Universal Secure Gateway device.....	102
Figure 54: High-level architecture of complete USG software platform integration.	104
Figure 55: Visualization of data flow in MQTT system with MQTT clients and MQTT broker. ...	105
Figure 56: User UI presenting daily consumption and production statistics based on 15-minute smart meter data.	108
Figure 57: User UI presenting actual and forecasted prosumption profile for the current day based on the 15-minute smart meter data.....	109
Figure 58: PPE as a main authority in the process of data exchange compliant to data protection and privacy preserving rules and policies.	112
Figure 59: CoReDEX Sequence Diagram.....	115



List of Acronyms

API	Application Programming Interface
AWS	Amazon Web Services
BE	Back-End
BEMS	Building Energy Management System
BIFTS	Blockchain-based Interoperability for Flexibility Tokenization and DR Settlement
BOT	Building Topology Ontology
CoReDEX	Compliant Regulatory Data EXchange
DC	Data Controller
dCO	domOS Common Ontology
DER	Distributed Energy Resources
DERes	Data Exchange Response
DEReq	Data Exchange Request
DLT	Distributed Ledger Technology
DP	Data Processor
DR	Demand Response
DS	Data Subject
DTO	Data Transfer Object
EC	European Commission
ECC	Execution Core Container
EDA	Exploratory Data Analysis
EFI	Energy Flexibility Index
EMS	Energy Management System
ETSI	European Telecommunications Standards Institute



EV	Electric Vehicle
GDPR	General Data Protection Regulation
GE	Generic Enabler
HTTP	Hypertext Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IDS	International Data Space
IDSA	International Data Spaces Association
IFC	Industry Foundation Classes
IoT	Internet of Things
ISO	International Organization for Standardization
IUDX	India Urban Data Exchange
JSON	JavaScript Object Notation
LoRaWAN	Long Range Wide Area Network
LSTM	Long Short-Term Memory
LWM2M	Lightweight Machine to Machine
MAE	Mean Absolute Error
ML	Machine Learning
MQTT	Message Queue Telemetry Transport
MS	Milestone
MSE	Mean Squared Error
NGSI	Next Generation Service Interface
NGSIv2	NGSI version 2
NGSI-LD	NGSI-Linked Data
OASC	Open & Agile Smart Cities
OpenADR	Open Automated Demand Response
OM	Ontology of Units of Measure



OPC	Open Platform Communications
OWL	Web Ontology Language
PoC	Proof of Concept
PPE	Privacy Protection Enforcement
PReq	Permission Request
Pres	Permission Response
PV	PhotoVoltaic
P2P	Peer-to-Peer
QoS	Quality of Service
RAM	Reference Architecture Model
RDF	Resource Description Framework
ReLU	Rectified Linear Unit
REST	Representational State Transfer
RNN	Recurrent Neural Network
SAREF	Smart Applications REference Ontology
SAREF4BLD	Smart Applications REference Ontology for Buildings
SAREF4ENER	Smart Applications REference Ontology for Energy
SAREF4SYST	Smart Applications REference Ontology for System
SAREF4CITY	Smart Applications REference Ontology for City
SC	Smart Contract
SI	International System of Units
TD	Things Description
TE	Technology Enabler
TFEU	Treaty of Functioning of the European Union
TRUE	TRUsted Engineering
UA	Unified Architecture



UC	Usage Control
UC APP	Usage Control App
UGW	Universal GateWay
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USG	Universal Secure Gateway
VEN	Virtual End Nodes
VTN	Virtual Top Nodes
WP	Work Package
W3C	World Wide Web Consortium



Executive summary

This deliverable outlines the different activities undertaken within the scope of WP3 tasks during the first year of the project, aligning them with the corresponding Technology Enablers as defined during the proposal phase. As stated in the DoA, these technology enablers will facilitate the development of the value-added services within WP4.

The main objective in each section is to provide an explanation of the activities done within the scope of the corresponding task, as well as to outline the next steps planned for the upcoming months.

Section 2 situates the different tasks/developments within the current DEDALUS framework architecture (detailed explained within D2.3 – High-level requirements for the DEDALUS social, technological, and business framework). Section 3 contains information on the activities performed up to now within T3.1 (DR-ready smart device & flexible resource data collection). Moving to the next section (section 4), it outlines the technologies that are being proposed to be used concerning the Energy Data Space compliant interoperability layer, giving an idea of how they are going to be applied in the scope of the project, also describing the proof of concept that has been developed concerning this issue. Following that, section 5 covers the activities related to the development of the DEDALUS ontology. Turning to the next section (section 6), it explores the methodology that is being developed for estimating the flexibility of heterogeneous multi-energy loads within buildings or districts. With regard to section 7, it analyses the use of blockchain for flexibility tokenization and demand response settlement. As for the eighth section, it examines the DEDALUS framework devoted to privacy, security and data protection issues. Lastly, a conclusions section has been added at the end of the document to summarize the main findings. As aforementioned, all the sections include a sub-section devoted to outlining the next steps to be addressed in the near future.



1. Introduction

1.1. Purpose and audience of the document

D3.1 – DEDALUS technological enablers for multi-value and energy carrier-agnostic residential DR (1st technology release) is produced within the WP3 – Technology enablers for residential energy consumer data-driven ‘activation’. This document contains information about the first release of the DEDALUS technological enablers for multi-value and energy carrier-agnostic residential DR to support DR aggregation for residential buildings and corresponding to the first technical release MS4.

This document will provide a description of all the outcomes obtained within each task for this first technology release.

1.2. Relation to other activities

As the main objective of WP3 is to provide technology enablers to be used (when needed) by the rest of the developments to be done in the project, it is linked with many activities developed in other work packages.

In general, WP3 receive inputs from the following tasks:

- T2.1 – Gap analysis and use case capitalisation for DR in the residential sector.
- T2.5 – Residential consumers business/privacy/cyber-security requirements and functional specifications.

The main outcomes from WP2 to be used in WP3 are use cases and requirements, along with the DEDALUS’ architecture, addressed within T2.5.

At its turn, WP3 gives inputs to the following work packages:

- WP4: the technology enablers developed within WP3 would be available for all the services to be carried out within the scope of WP4 (e.g. interoperability from T3.2, ontology from T3.3, privacy and security issues from T3.6...).
- WP5: WP3 technology enablers will be also available to be used, when needed, in the different demo-sites.

1.3. Structure of the document

This document is structured as follows:

- Section 2 is devoted to the DEDALUS framework architecture, where the architecture obtained within T2.5 will be linked with the technology enablers to be developed in the scope of WP3.
- Section 3 outlines all the outcomes obtained in the scope of the T3.1, related to the DR-ready smart device and flexible resource data collection.



- Section 4 contains information regarding the Energy Data Space compliant interoperability layer that is being defined in the scope of the project (T3.2).
- Section 5 reports on the DEDALUS ontology that has been developed in order to let all the services, technology enablers and so on, communicate using the same “vocabulary” (T3.3).
- Section 6 focuses on the developments on estimating the potential flexibility that may be offered through heterogeneous multi-energy flexible loads (T3.4).
- Section 7 is devoted to the blockchain-based solution for privacy-preserving flexibility assessment, P2P flexibility tokenisation and demand response settlement using smart contracts (T3.5).
- Section 8 addresses the developments that will ensure compliance with all the applicable privacy, security and legal requirements (T3.6).



2. WP3 and the DEDALUS framework architecture

While the DEDALUS architecture has been outlined in D2.3 – High-level requirements for the DEDALUS social, technological, and business framework, a concise overview, along with its primary components relevant to WP3 is also provided in this document to enhance readability throughout the text.

As stated in D2.3, a six levels architecture has been designed. From top to bottom, the architecture begins with the data acquisition layer, which includes real world devices and data sources. Then, the business platforms level is in charge of gathering all the information coming from the first layer and make it available to the next level: the middleware, where the developments described within section 4 are considered. Business Platforms level includes the ARCELIIK IoT platform described within section 3 along with the business platforms available in each pilot. Regarding the Middleware layer, it is important to remark that its main aim is to define the communication among the different services, and also between them and the external data space.

Next level (Knowledge Layer) encompasses the data repositories and the DEDALUS ontology (refer to section 5), which are aimed at ensuring that all services communicate using a shared language. The use of ontologies guarantees a mutual understanding of information and explicitly states domain assumptions, thereby enabling a better comprehension of the available data.

As shown in Figure 1, a specific layer devoted to the DEDALUS value-added services has been defined. This layer accommodates the various developments within the scope of WP4 and WP2, as well as the services from WP3 (such as the ones described in sections 6 and 7). Finally, the Presentation Layer is devoted to the developments responsible for user interaction: nudging apps, dashboards and recommendation systems.

Finally, considering the importance of security, privacy, and data-protection issues, a dedicated layer (Security Layer) has been integrated into the architecture. This layer includes various developments, among which the one described within section 8 is worth highlighting.



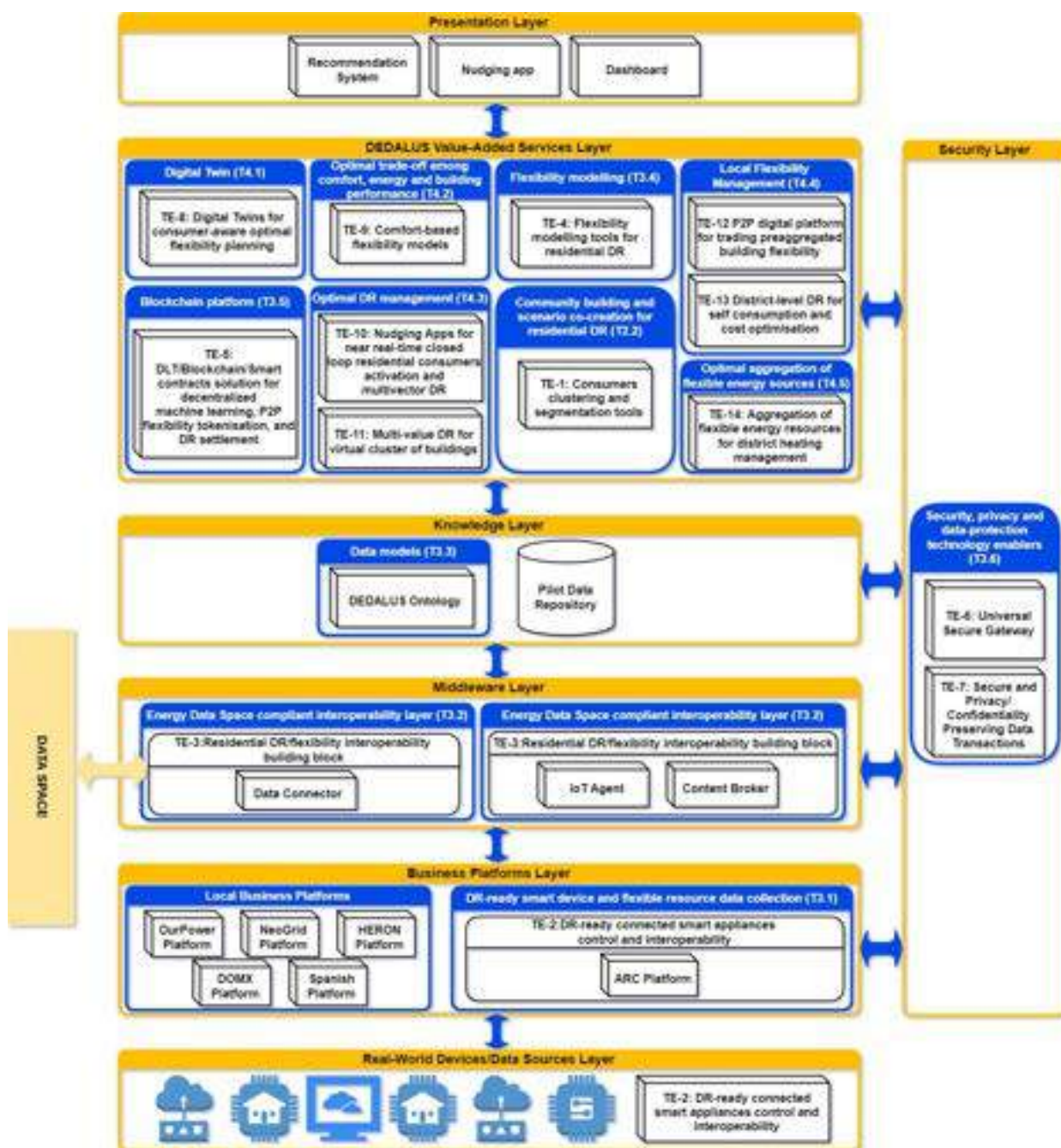


Figure 1: DEDALUS framework architecture

3. DR-ready smart device & flexible resource data collection – 1st technology release

This section contains detailed information about ARCELIK IoT platform (which commercial name is Arçelik HomeWhiz) and its deployment within the DEDALUS project. Besides, a summary of the next steps to be carried out is given.

3.1. ARCELIK's IoT Platform description

In the dynamic and rapidly evolving domain of smart home technologies, the integration of disparate domestic appliances into a unified, controllable ecosystem represents a significant advancement toward optimizing user convenience and energy efficiency. The development of platforms such as Arçelik HomeWhiz (named Arcelik IoT Platform in the DoA, and also in the rest of the document. Arçelik HomeWhiz is its “commercial name”) signifies a critical juncture in the way individuals interact with their domestic environments, encouraging a transition towards interconnected and intelligent habitats. This section aims to dissect the complex architecture and operational capabilities of the Arçelik HomeWhiz platform, elucidating its pivotal role in the innovation of demand response domain. By examining its system architecture, data management methodologies, and empirical applications within selected pilot regions, this discourse seeks to furnish a thorough understanding of how such platform can redefine standards in the domain of IoT-enabled domestic appliances, thus facilitating more sustainable and efficient home management practices.

Arçelik HomeWhiz, namely TE-2, is a smart home platform developed by the leading household appliances manufacturer, Arçelik. The platform provides users with a connected ecosystem where they can control and manage their smart home appliances remotely, using a smartphone or tablet.

HomeWhiz integrates a wide variety of smart home appliances, ranging from ovens, refrigerators, washing machines, dishwashers, to smaller appliances such as coffee machines and vacuum cleaners. For instance, users can control their washing machine from anywhere, scheduling a wash cycle to finish just as they get home. Similarly, they can preheat the oven on their way back from work or check the contents of their refrigerator while shopping. This connectivity contributes to an overall enhanced lifestyle, offering convenience, efficiency, and energy-saving benefits.

In addition, Arçelik HomeWhiz can collect and analyse data about usage patterns of the appliances, which can be utilized to improve energy efficiency, troubleshoot issues proactively and offer better personalized user experiences. The platform's integration with various smart devices makes it an important part of smart homes and gets significant attention in the space of IoT-enabled appliances.

Figure 2 depicts the overall architecture of the platform, which utilizes AWS Cloud for scalability and robustness of the system. The main cloud services used are AWS's IoT Core for connectivity,



Lambda for microservices, and RDS for storage. Smart home devices such as gateways, smart plugs, smart home appliances, and various sensors connect to the cloud via IoT Core over the MQTT protocol. Large household appliances usually have a built-in WIFI modules to connect to the cloud on the other hand smart plugs, and auxiliary sensors requires a gateway to connect. The communication between gateway and smart plugs & sensors done via Bluetooth Low Energy.

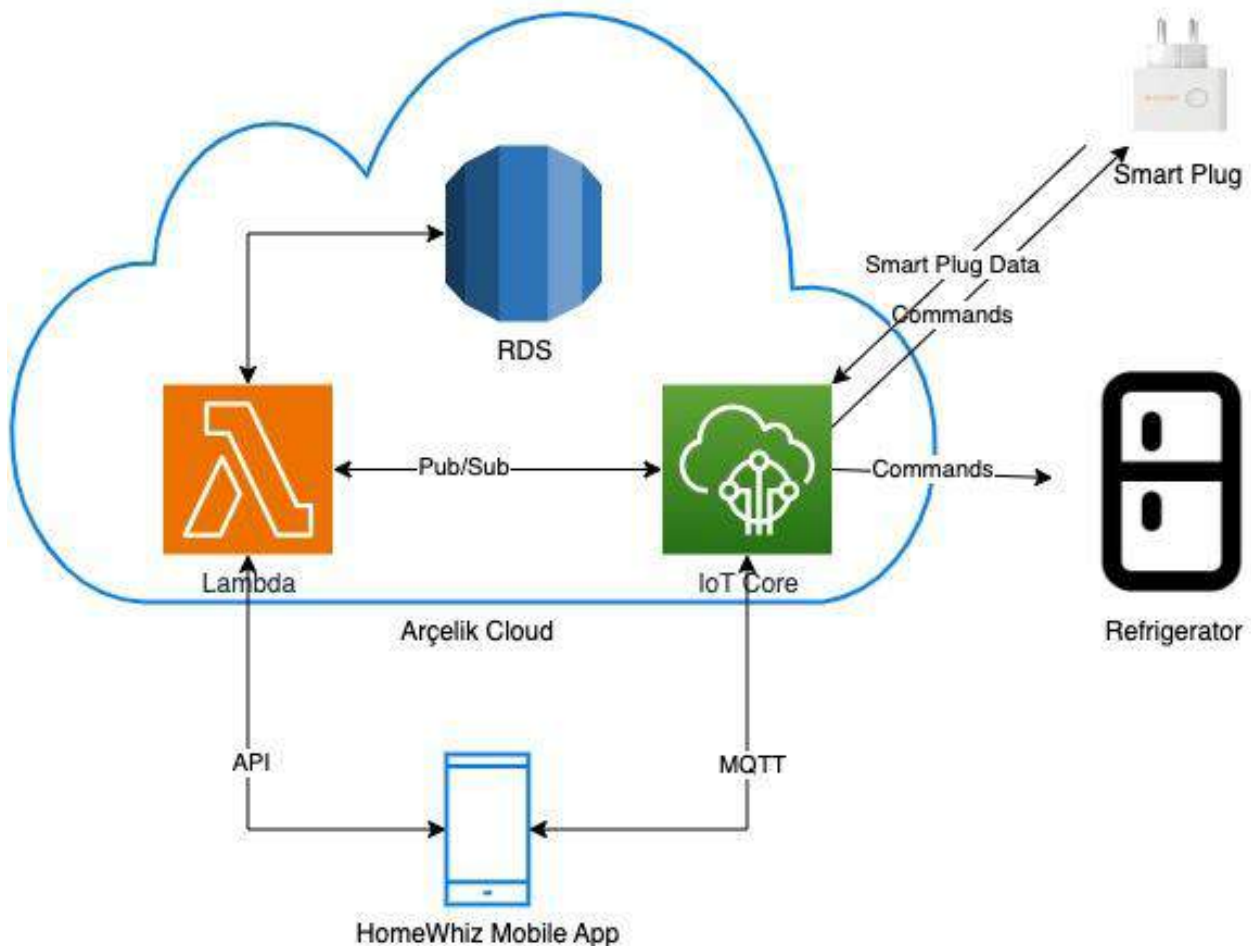


Figure 2: Simplified architecture of Arçelik HomeWhiz Platform

Designing a data model stands to the diverse requirements of flexibility services, web services, and MQTT integration, while accommodating data collection from Arcelik HomeWhiz gateways and smart plugs. The data model should encompass fields for power consumption, temperature, and humidity, reflecting the information collected from Arcelik HomeWhiz smart plugs and gateways deployed within various rooms or environments. It should prioritize scalability and interoperability to ensure seamless integration with different systems and platforms. Additionally, the model should facilitate real-time data processing and analysis to support dynamic adjustments and decision-making processes. Implementing standardized protocols and structures will enhance compatibility and ease of data exchange, optimizing the efficiency and effectiveness of the overall system.

In designing a data model in order to meet the multifaceted requirements of flexibility services, web services, and MQTT integration, while accounting for data collection originating from gateways and smart plugs, adherence to established standards, ontologies, and protocols is substantial. Embracing widely adopted communication protocols like MQTT ensures efficient data exchange between distributed components, while utilizing results for web services enables interoperability with diverse platforms. The model should incorporate a modular structure to accommodate future expansions and revisions, with a focus on maintaining compatibility with existing repositories and data management systems. By building upon existing standards and protocols, the data model can capitalize on prior industry expertise and infrastructure, expediting development efforts and fostering a cohesive ecosystem for managing power consumption, temperature, and humidity data collected from smart plugs and gateways across various environments.

Within the framework of DR-ready smart device & flexible resource data collection working task, ARC have been deliberated with pilot regions in Spain and Greece to determine the method for data extraction. The examination encompassed the investigation of resource data where current and voltage data acquired from smart plugs, which directly supply power consumption data, and alongside humidity and temperature data obtained from gateways. Communication with each pilot region is facilitated through MQTT brokers, which are installed and maintained within their respective regions. Transactions involving the issuance of commands to open and close smart plugs are also conducted via MQTT.

Figure 3 outlines the communication architecture for data collection, in which each pilot requiring its own MQTT broker to collect data from Arçelik Cloud, which consolidates all the collected data from devices and push it to the pilot sites for further analysis. In this architecture, voltage, current, and power measurements from smart plugs and temperature and humidity from gateways will be gathered. Gateway readings will be utilized to make DR-decisions based on the environment conditions inside the dwellings. Additionally, smart plugs will be used to control the non-smart devices by allowing remote power on and off based on the DR decisions.



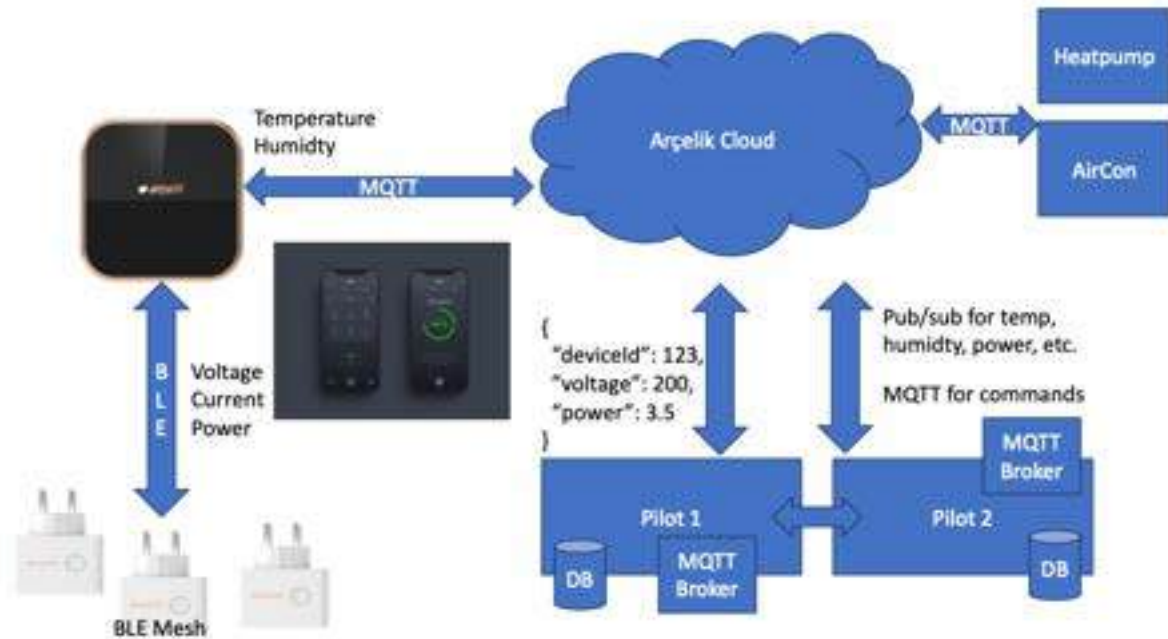


Figure 3: The communication architecture between Arçelik HomeWhiz Platform and Pilot Sites

3.2. ARCELIK IoT platform deployment in the DEDALUS project

In the initial phase, 10 smart plugs and 3 gateways were dispatched to a single domestic unit comprising 3 dwellings in Greece for POC (Proof of Concept). Subsequently, in the second phase, a total of 40 smart plugs and 12 additional gateways will be deployed across 4 more domestic units.

In Spain, during the first phase, 4 smart plugs and 2 gateways were provided for a single domestic unit comprising 2 dwellings for POC. In the second phase, an additional 20 smart plugs and 10 gateways will be dispatched.

3.3. Next steps

The next steps for DR-ready smart device integration with Arçelik HomeWhiz involve finalizing the data model for seamless data exchange, selecting and enabling specific large appliances to respond to DR signals, completing pilot site installations, and conducting system testing. Once everything is operational, the pilot programs in Spain and Greece can commence to assess the effectiveness of the DR strategy and collect data for further refinement. By following these steps, Task 3.1 can establish a robust system for DR-ready devices and flexible data collection, paving the way for wider deployment. Besides, the aforementioned Arçelik HomeWhiz data model will be adapted to the DEDALUS ontology (described within section 5).



4. Energy Data Space compliant interoperability layer – 1st technology release

This section provides guidance on managing technical and structural interoperability, offering an overview of the main technologies envisaged to be used within DEDALUS project. Additionally, an explanation on how these technologies are intended to be used in the scope of the project is included.

The Institute of Electrical and Electronics Engineers (IEEE) Standard Computer Dictionary defines interoperability as “the ability of two or more systems or components to exchange information and to use the information that has been exchanged”¹. In general terms, there is a list of interoperability layers defined, which includes: interoperability governance, integrated public service governance, legal interoperability, organisational interoperability, technical/syntactic interoperability, and semantic interoperability. In the scope of the DEDALUS project, semantic and syntactic interoperability are of outmost importance. Semantic interoperability provides that the precise format and meaning of exchanged data and information is preserved and understood throughout exchanges between parties. It includes developing vocabularies and schemas to describe data exchanges and ensures that data elements are understood in the same way by all communicating parties. In turn, the syntactic interoperability concerns applications, services and infrastructures, and the secure data sharing, describing the exact format of the information to be exchanged in terms of grammar and format.²

This section describes the interoperability building blocks that are part of the Technology Enabler 3 (TE-3), namely:

- FIWARE Context Broker;
- FIWARE Next Generation Service Interface (NGSI) IoT Agents;
- Data Connector.

4.1. FIWARE-based Interoperability Solution

FIWARE³ is an open-source initiative whose mission is to build an *open sustainable ecosystem around public, royalty-free and implementation driven software platform standards that will ease the creation of Smart Applications in multiple sectors*. FIWARE promotes open standards, open-source software, and open data to endorse collaboration, interoperability, and reusability.

¹ "IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries," in IEEE Std 610 , vol., no., pp.1-217, 18 Jan. 1991, doi: 10.1109/IEEESTD.1991.106963

² European Interoperability Framework – Implementation Strategy. Annex 2 to the Communication from the Commission to the European Parliament, the Council, the European Economic and Social Committee and the Committee of the Regions. COM (2017) 134 final, European Commission, Brussels, 2017

³ <https://www.fiware.org/foundation/>



FIWARE provides a common platform and a set of Application Programming Interfaces (APIs) that ensure interoperability between different systems and components. This standardization allows developers to integrate and scale their solutions easily, reducing complexity, and development time.

FIWARE is a curated platform of open-source components, called Generic Enabler (GE), that can be assembled together and with third-party components to build platforms that support the development of smart solutions faster, easier and cheaper. Figure 4 below shows, as example, the FIWARE reference architecture tailored to the Smart Energy domain.

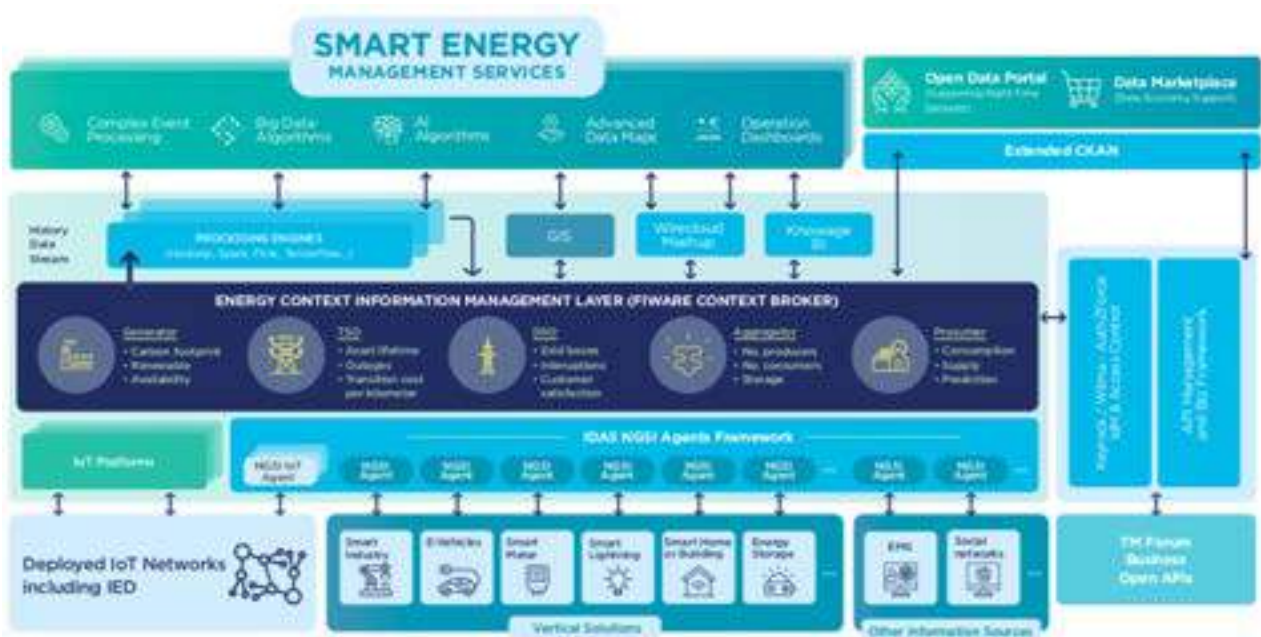


Figure 4: Reference architecture for Smart Energy management solutions “POWERED BY FIWARE”⁴

The FIWARE platform is structured in seven major parts, each representing a certain aspect of FIWARE services (Figure 5). A rich suite of complementary FIWARE components is available, dealing with:

- Interfacing with the Internet of Things (IoT), Robots, and third-party systems;
- Context Data/API management, publication, and monetization;
- Processing, analysis, and visualization of context information.

⁴ https://www.fiware.org/wp-content/directories/marketing-toolbox/material/FIWAREBrochure_SmartEnergy.pdf

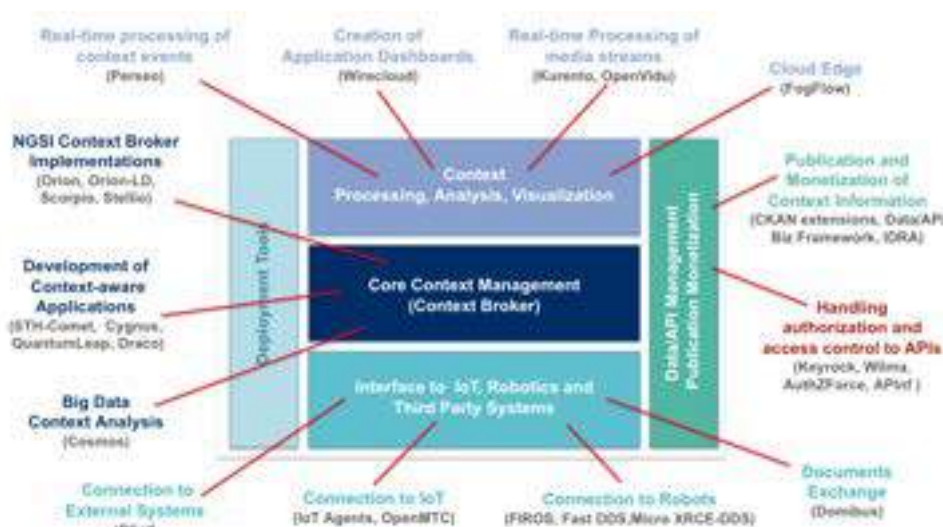


Figure 5: FIWARE platform high-level architecture⁵

4.1.1. FIWARE Context Broker

FIWARE Context Broker⁶ is the core component of a FIWARE platform. According to the OPEN DEI Design Principles for Data Spaces Position Paper [1], which reports agreements on the building blocks for a soft infrastructure and governance for data spaces, the Context Broker is one of the building blocks for interoperability being an example of Data Exchange Application Programming Interface (APIs) building block, which facilitates the sharing and exchange of data (i.e., data provision and data consumption/use) between data space participants.

The Context Broker allows to model, manage, and gather context information at large scale, enabling context-aware applications. It makes it possible to store and access context-information using simple context models, offers a Representational state transfer (REST) API and supports subscriptions and notifications. It can also be used by third-party applications to update or consume context information. FIWARE Next Generation Service Interface (NGSI) is the API exported by a FIWARE Context Broker, used for the integration of platform components within a FIWARE platform and by applications to update or consume context information. FIWARE NGSI API specifications have evolved over time, initially matching NGSI version 2 (NGSiv2) specifications and now being aligned with the European Telecommunications Standards Institute (ETSI) NGSI-Linked Data (LD) standard⁷.

⁵ <https://fiware-academy.readthedocs.io/en/latest/>

⁶ <https://www.fiware.org/catalogue/>

⁷ ETSI GS CIM 009 V1.5.1 (2021-11), Context Information Management (CIM) NGSI-LD API, https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.05.01_60/gs_CIM009v010501p.pdf

The main elements in the NGSiv2⁸ model are entities, attributes, and metadata, as shown in Figure 6.

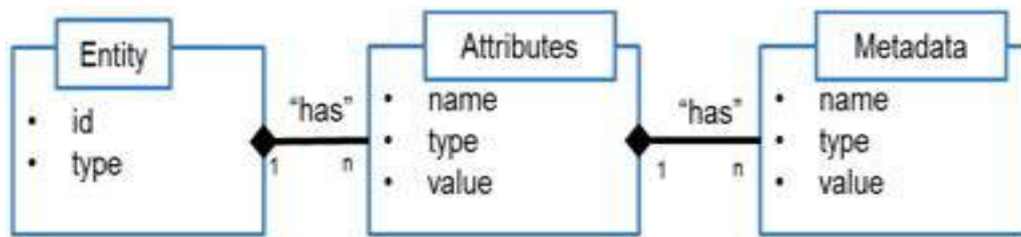


Figure 6: NGSi data model⁹

An entity represents a thing, any physical or logical object (e.g., a sensor, a person, a room, an issue in a ticketing system, etc.). It is characterized by an identifier and a type describing the “thing” represented by the entity. Each entity may have one or more context attributes, or properties, (e.g., a measurement value), featuring:

- a name (what kind of property),
- an attribute type (the NGSi value type of the attribute value),
- an attribute value (the actual data),
- metadata (describing properties of the attribute value, e.g., the accuracy of the measurement value).

Context metadata features:

- a metadata name (describing the role of the metadata),
- a metadata type (describing the NGSi value type of the metadata value), and
- a metadata value containing the actual metadata.

The other version of NGSi, NGSi-LD¹⁰, has a different underlying data model than NGSiv2. Data are described in triples in a subject-predicate-object pattern resulting in a graph representation of the context data. Furthermore, in NGSi-LD the ID of an entity should be a Uniform Resource Identifier (URI) ensuring a consistent representation of the identifier of an entity. Figure 7 shows the underlying model of NGSi-LD. It has no metadata element, and the Attributes element of NGSiv2 in NGSi-LD refers to either Property or Relationship, where the former represents literal values (strings, decimals, etc.), while the latter represent relationships between different entities.

⁸ <https://fiware-tutorials.readthedocs.io/en/latest/>

¹⁰ <https://ngsi-ld-tutorials.readthedocs.io/en/latest/>



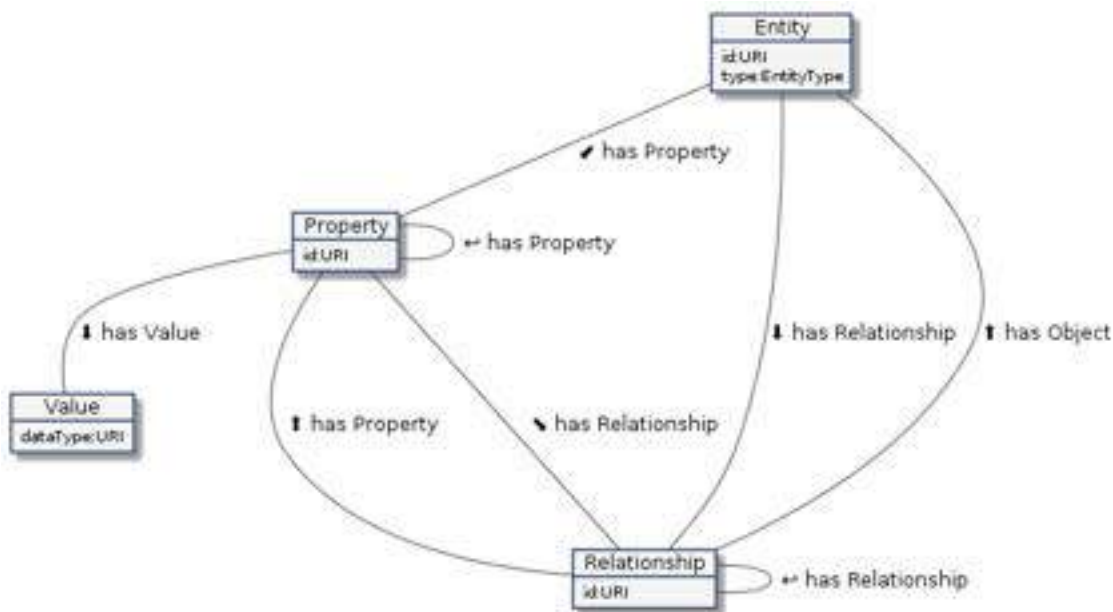


Figure 7: UML representation of NGSI-LD information model¹¹

Orion Context Broker¹² is an implementation of the Publish/Subscribe Context Broker GE and offers access to it via NGSIv2 REST API.

As shown in Figure 8, Orion Context Broker offers a set of operations used by context producers to publish and update context information and by context consumers to consume that information by means of synchronous or asynchronous tasks [2]. The expected uses of Orion are:

- *retention of current instances of harmonized data entities processed from IoT devices and external sources (context data);*
- *storage of a window of short term historical harmonized data entities that may be queried directly via a third-party application;*
- *storage of any analytics and intelligence results which become additional context data that can be queried or mashed up with other IoT data or external data sources;*
- *register context provider (source) applications, e.g., a temperature sensor within a room;*
- *update context information, e.g., send updates of temperature;*
- *being notified when changes on context information take place (e.g., the temperature has changed) or with a given frequency (e.g., get the temperature every minute);*
- *query context information.*

¹¹ <https://github.com/FIWARE/tutorials.Linked-Data>

¹² <https://fiware-orion.readthedocs.io/en/master/>



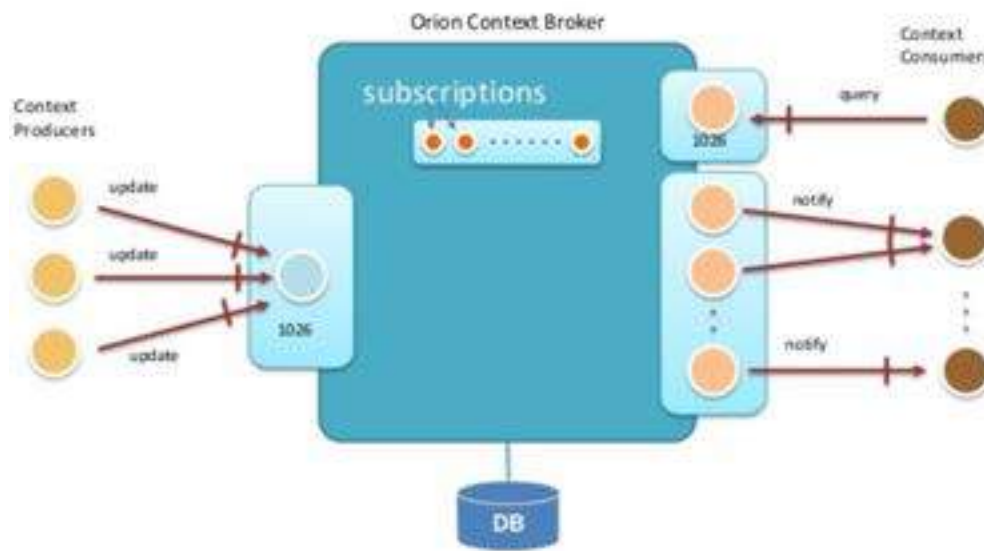


Figure 8: Orion Context Broker ¹⁰

Orion-LD¹³ is a Context Broker for context data management which supports both the NGSI-LD and the NGSIv2 APIs. It is a fork of the original Orion Context Broker extending support to add NGSI-LD and linked data concepts.

4.1.2. NGSI Agents

A wide range of GE is available in the FIWARE Catalogue⁶ to facilitate the interface with the IoT and third-party systems for gathering context information or trigger actuations in response to context updates. FIWARE provides GEs called NGSI Agents that allow to simplify the management and the integration of devices by collecting data from them through heterogeneous protocols and translating them into the NGSI standard platform language. NGSI Agents may also facilitate the integration of vertical solutions (e.g., smart meters, smart home, smart building, energy storages, etc.) and of other sources of data (e.g., energy management systems, social networks, etc.) with the FIWARE ecosystem. Besides the NGSI Agents already present in the FIWARE Catalogue, libraries are made available by FIWARE to create custom NGSI Agents.

NGSI IoT Agents stay at the southbound of the Context Broker and support several IoT protocols with a modular architecture. As an example, the Backend Device Management (IDAS) GE in Figure 9 illustrates a wide range of IoT Agents making it easier to interface with devices using the most widely used IoT protocols:

- *IoT Agent for JavaScript Object Notation (JSON) – a bridge between HTTP/ Message Queue Telemetry Transport (MQTT) messaging (with a JSON payload) and NGSI*

¹³ <https://github.com/Fiware/context.Orion-LD?tab=readme-ov-file>

- *IoT Agent for Lightweight Machine to Machine (LWM2M) – a bridge between the Lightweight M2M protocol and NGSI*
- *IoT Agent for Ultralight – a bridge between HTTP or MQTT messaging (with an UltraLight2.0 payload) and NGSI*
- *IoT Agent for Long Range Wide Area Network (LoRaWAN) – a bridge between the LoRaWAN protocol and NGSI*
- *IoT Agent for Open Platform Communications (OPC) Unified Architecture (UA) – a bridge between the OPC UA protocol and NGSI*
- *IoT Agent for Sigfox – a bridge between the Sigfox protocol and NGSI.*

It is possible to create custom IoT Agent by using the IoT Agent library¹⁴.

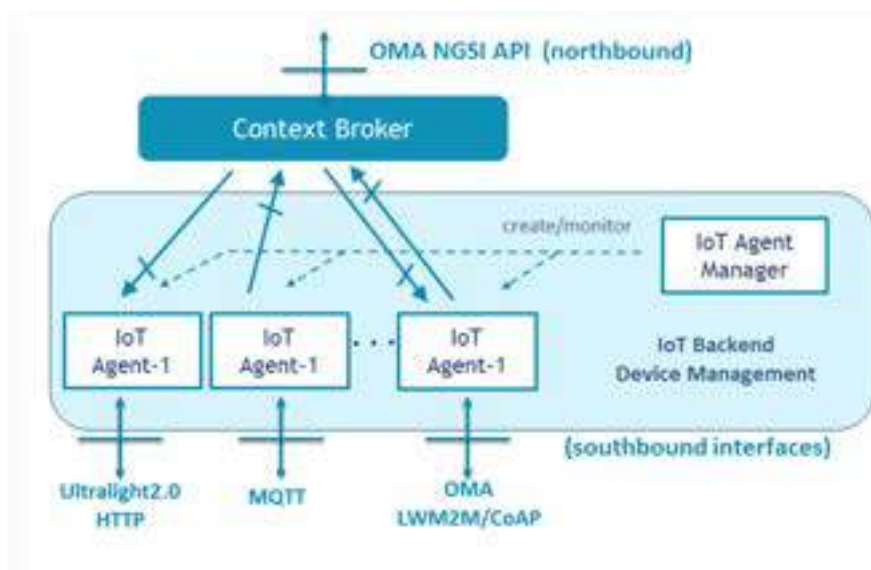


Figure 9: FIWARE Backend IoT Device Management¹⁵

NGSI IoT Agents also allows to send commands to devices. The sequence diagram in Figure 10 shows different NGSI interactions between an IoT Agent and the Context Broker.

¹⁴ <https://github.com/telefonicaid/iotagent-node-lib>

¹⁵ "FIWARE IOT Stack: IoT Agents (IoTAgent)", [Online]. Available: https://iot-platform-docs.readthedocs.io/en/task-updatepepproxdoc/device_gateway/.

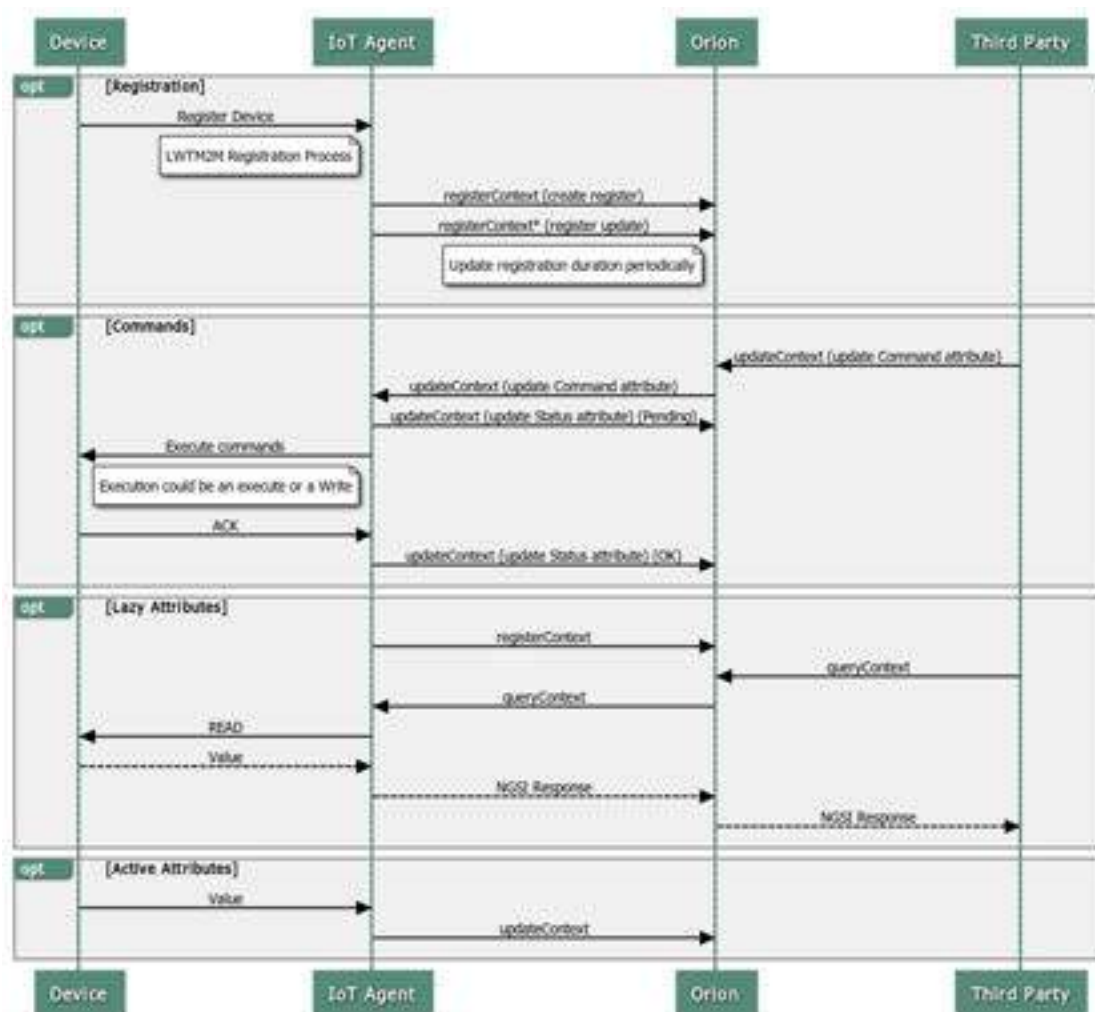


Figure 10: NGSI interactions an IoT Agent makes with the context broker¹⁶

4.1.3. Smart Data Models

The FIWARE Foundation, TM Forum, India Urban Data Exchange (IUDX), and Open & Agile Smart Cities (OASC) are leading a joint collaboration initiative to support the adoption of a reference architecture and compatible common data models that underpin a digital market of interoperable and replicable smart solutions in multiple sectors, starting with Smart Cities¹⁷.

NGSI standard allows requesting data from many systems with a standard format, REST compatible, that can cope with most of the needs including geoquerying, next to real-time

¹⁶ <https://iotagent-node-lib.readthedocs.io/en/latest/architecture/index.html>.

¹⁷ <https://www.fiware.org/data-models/>



requests and heterogeneous sources¹⁸. By sharing data models, it is possible to have a full interoperability by knowing how to request for data and what structure has the retrieved data. Data models play a crucial role because they define the harmonised representation formats and semantics that will be used by applications both to consume and to publish data.

Smart Data Models¹⁹²⁰ is an Umbrella Repository of free and open-licensed models based on real world use-cases. A Smart Data Model includes four elements:

- the schema, or technical representation of the model defining the technical data types and structure,
- the specification of a written document for human readers,
- a Uniform Resource Identifier (URI) with a working Uniform Resource Locator (URL) with basic data about the attribute or the entity
- the examples of the payloads for NGSiv2 and NGSI-LD versions.

All data models are public and of royalty-free nature. The licensing mode grants to the users:

- Free use
- Free modification
- Free sharing of the modifications

Data models are grouped into subjects. Currently, every subject is a git repository. The subjects can belong to one or several domains and the domains represent industrial sectors.

A full list of the data models in machine-readable JSON format can be found in the file official list data models JSON²¹. Further details on the available subjects, properties, and their descriptions can be found via the search tool²².

4.2. International Data Space (IDS)

A Data Space is a decentralized infrastructure for trustworthy data sharing and exchange in data ecosystems based on commonly agreed principles [1]. Data Spaces have been proposed by the European Union as a way to support data sharing, while respecting the rules that safeguard European values²³. A Data Space ensures trust and security. Moreover, data sovereignty is ensured to guarantee the control of the data by their owner: a data owner can define usage restriction to their data, before sharing it, and data consumers must accept the usage restrictions.

¹⁸

https://docs.google.com/presentation/d/1k021ZzRyk2PtikubOFNa3KyXeXL5nTmWWxyhsOY0r1k/edit#slide=id.g10849e1f66c_0_1062

¹⁹ <https://smartdatamodels.org/>

²⁰ <https://github.com/smart-data-models>

²¹

https://github.com/smart-data-models/data-models/blob/master/specs/AllSubjects/official_list_data_models.json

²² <https://smartdatamodels.org/index.php/ddbb-of-properties-descriptions/>

²³ <https://digital-strategy.ec.europa.eu/en/policies/strategy-data>



The International Data Spaces Association (IDSA)²⁴ is a powerful alliance to drive the advancement of data sovereignty, secure data exchange, and standardized data interoperability in the digital ecosystem.

The International Data Space (IDS) is a virtual data space leveraging existing standards and technologies, as well as governance models well-accepted in the data economy, to facilitate secure and standardized data exchange and data linkage in a trusted business ecosystem. It thereby provides a basis for creating smart-scenarios and facilitating innovative cross-company business processes, while at the same time guaranteeing data sovereignty for data owners.²⁵

IDSA has created the International Data Spaces (IDS) Reference Architecture Model (RAM) (Figure 11), version 4.0²⁶ at the time of writing. Data and endpoints are described semantically in the IDS Information Model²⁷, and certification ensures that every system follows the architecture and uses the information model in the defined way.

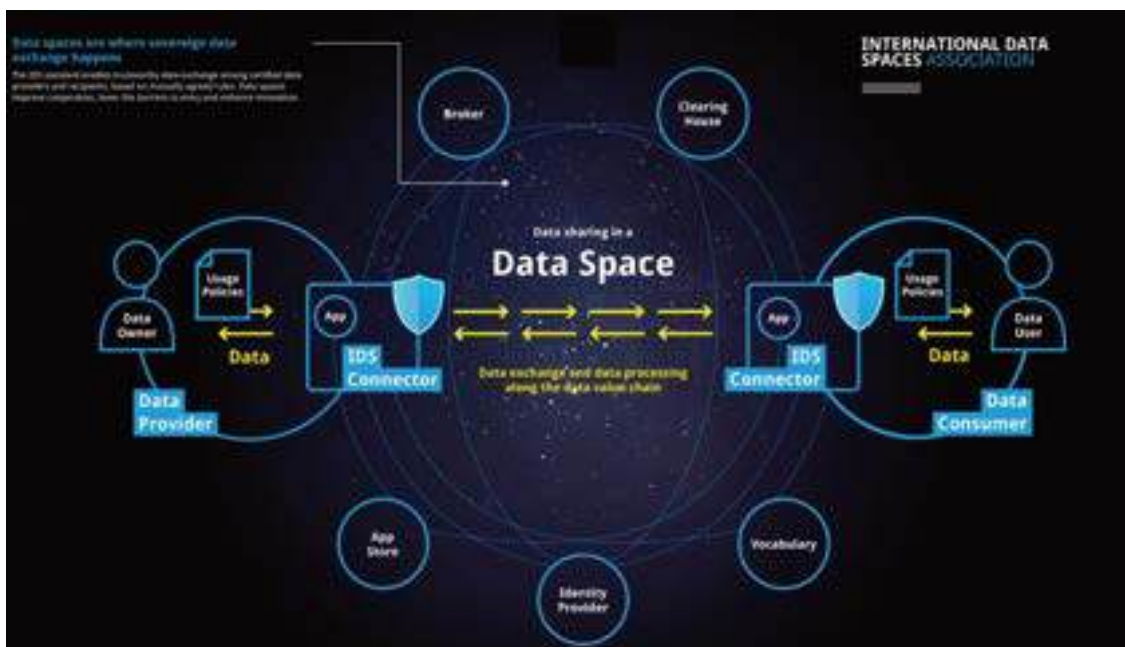


Figure 11: IDS Reference Architecture²⁸

The IDS Connector is the core of the Data space. Is the gateway to connect existing systems and their data to an IDS ecosystem. The IDS Connector allows to exchange data and enrich it with metadata.

²⁴ <https://internationaldataspaces.org/>

²⁵ https://docs.internationaldataspaces.org/ids-knowledgebase/v/ids-ram-4/introduction/1_1_goals_of_the_international_data_spaces

²⁶ <https://docs.internationaldataspaces.org/knowledge-base/ids-ram-4.0>

²⁷ https://docs.internationaldataspaces.org/ids-knowledgebase/v/ids-ram-4/layers-of-the-reference-architecture-model/3-layers-of-the-reference-architecture-model/3_3_informationlayer

²⁸ <https://docs.internationaldataspaces.org/ids-knowledgebase/v/how-to-build-data-spaces/>

The Identity Provider should offer a service to create, maintain, manage, monitor, and validate identity information of and for participants in the IDS.

IDS Connectors and Identity Provider are mandatory components of the dataspace.

A Metadata Broker acts as a mediator between data providers offering data, and data users requesting data. It also acts as a data source registry.

The Clearing House provides clearing and settlement services for all financial and data exchange transactions. The Clearing House logs all activities performed in the course of a data exchange. The logging information can also be used to resolve conflicts.

The Industrial Data Space promotes the development of a business ecosystem in which participants may develop software (especially data services) and make this software available via the App Store. AppStore consists of a registry for available IDS Apps in this AppStore. IDS Apps are independent, functional, and re-usable software asset that is deployable, executable, and manageable on an IDS Connector.

In order to better define one's own data, domain-specific vocabularies can be created and made available for all IDS participants.

4.2.1. Data Connector

IDSA collaborate with FIWARE Foundation to drive the advancement of data sovereignty, secure data exchange, and standardized data interoperability in the digital ecosystem. Standardized interoperability is essential to build up a Data Space because different data ecosystems will exchange different kinds of data in different formats and protocols. With standardized interoperability, every system can be interoperable in the IDS.

The TRUsted Engineering (TRUE) Connector (Figure 12) enables the trusted data exchange in order to be active part of an IDS Ecosystem. The TRUE Connector is composed of three components:

- *Execution Core Container (ECC), an open-source project designed by Engineering Ingegneria Informatica (ENG) in charge of the data exchange through the IDS ecosystem representing data using the IDS Information Model and interacting with an external Identity Provider. It is also able to communicate with an IDS Broker for registering and querying information.*
- *Back-End (BE) Data Application, an open-source project designed by ENG. It represents a trivial data application for generating and consuming data on top of the ECC component.*
- *Usage Control (UC) Data Application, a customized version of the Platoon project base application for integrating Usage Control functionality.*



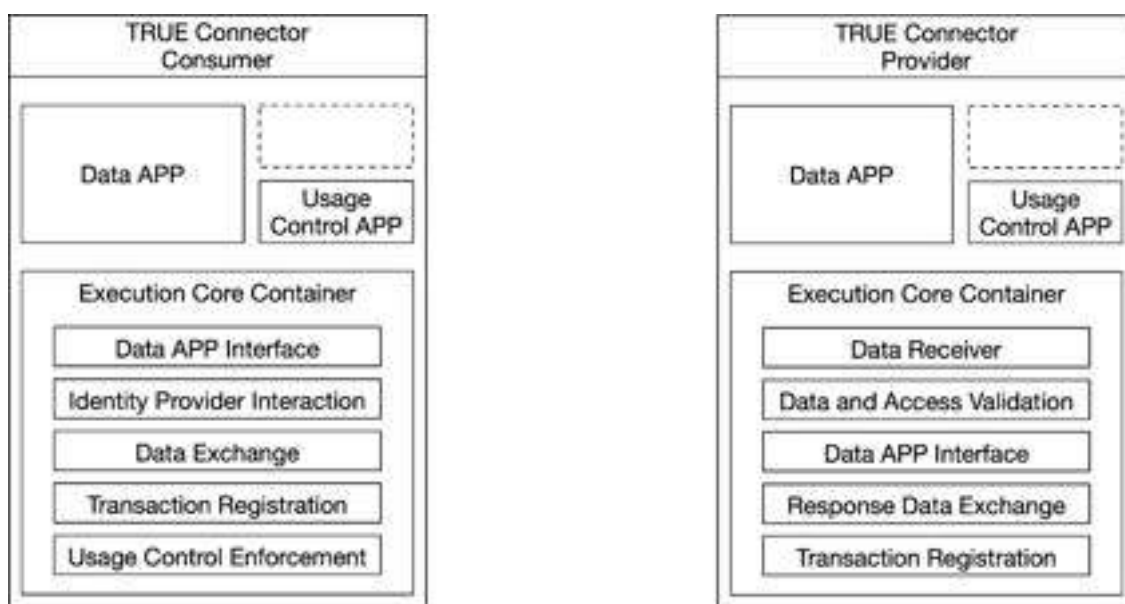


Figure 12: TRUE Connector architecture

The FIWARE TRUE Connector²⁹ is a customization of TRUE Connector that includes a NGSI-LD Data App to enable the data exchange using the FIWARE NGSI-LD Context Broker. As can be seen in Figure 13, data providers and data consumers have their own FIWARE platform with the Context Broker and the NGSI-LD Data App of the Prosumer converts in NGSI-LD standard context information and send it to the Consumer and vice versa.

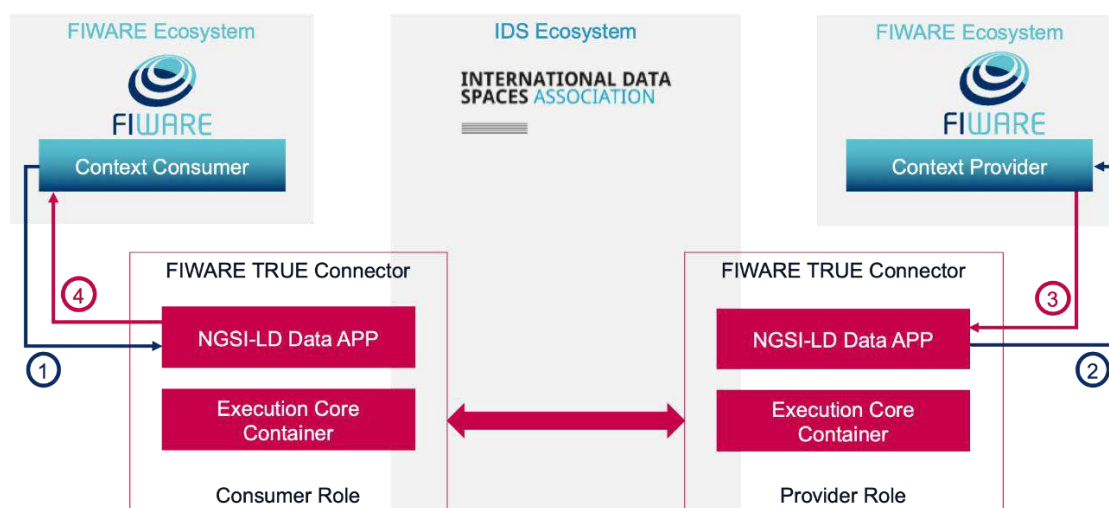


Figure 13: FIWARE TRUE Connector²⁹

²⁹ <https://fiware-true-connector.readthedocs.io/en/latest/>

The OneNet connector³⁰, based on TRUE Connector and developed in the context of OneNet project³¹, aims to enable a European Energy Data Space, combining the IDS principles with the advantages of the FIWARE ecosystem ensuring a seamless and secure data exchange in a completely end-to-end decentralized approach³². The OneNet Connector is ready to be deployed and integrated in any existing platform and offers user-friendly interfaces (both as REST APIs and GUI) enabling users and platforms to share data. In addition, the OneNet Connector offers a pre-defined, dynamically evolving list of Cross Platform Services, business objects and corresponding Data Profiles being available for enabling semantic and data interoperability.

The OneNet connector architecture includes (Figure 14):

- a graphic user interface (GUI), which allows to configure the OneNet Connector and use its main features. It allows registration as data provider / consumer and creation of new data offerings service.
- OneNet Data App, whose main goal is to be the entry point of the OneNet Connector and provides access to all the main features of the OneNet Connector, including those for data exchange, through standard interfaces based on REST API. It is based on the NGSI standard for the implementation of APIs and for data exchange and offers three main services:
 - Entity creation
 - Registration / Subscription
 - Data retrieving
- Context Broker, is the FIWARE component that manages the overall data exchange in the standard NGSI-LD format
- IDS based components, a series of components and tools, that enable the IDS processes for the data exchange:
 - ECC has the task of implementing IDS-based data exchange processes (authentication, metadata exchange, access control, logging, etc. ...)
 - Clearing House (CH), logging system of all data exchanges (not available in the first version but being integrated afterwards)
 - Usage Control App (UC APP), module for verifying access and use of data (not available in the first version, but being integrated afterwards).

³⁰ OneNet Consortium, “Extended Interoperability and Management with FIWARE D6.3” , 2023, [Online]. Available: https://www.onenet-project.eu/wp-content/uploads/2023/09/OneNet_D6.3_v1.0.pdf

³¹ <https://www.onenet-project.eu/>

³² International Data Space Association, “Data Connector Report” , 2023, [Online]. Available: https://internationaldataspaces.org/wp-content/uploads/dlm_uploads/IDSA-Data-Connector-Report-5_March-2023.pdf



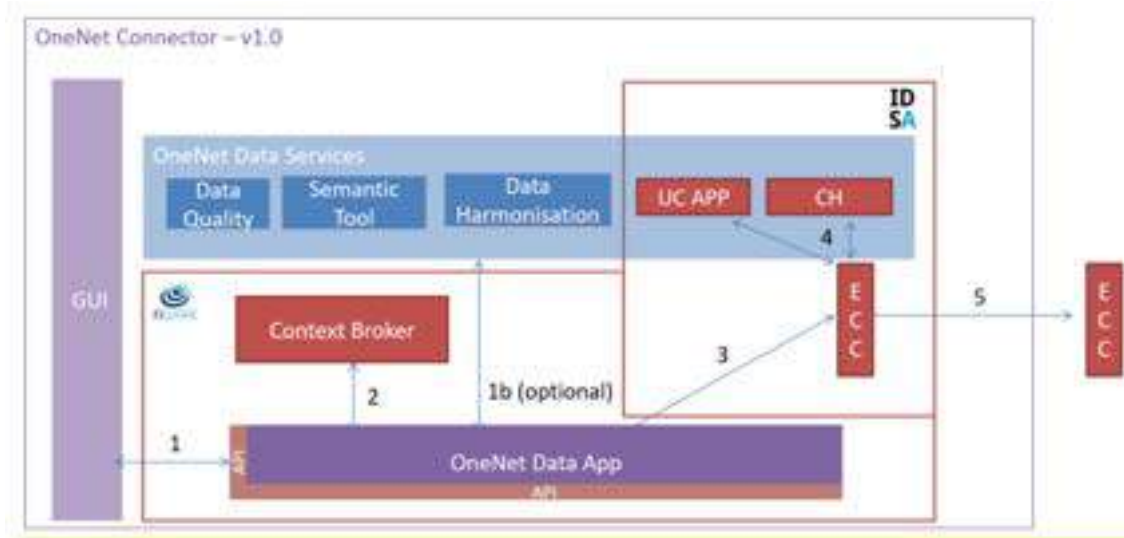


Figure 14: OneNet Connector Architecture - v1³⁰

4.3. Interoperability in DEDALUS

In the context of the DEDALUS project, the FIWARE Context Broker is part of the interoperability middleware that is in charge of ensuring the proper communication between the local business platforms used by project pilots and the value-added services provided within WP4. In that sense, a questionnaire has been circulated to pilots to gather the needed information, such as format of the data that can be provided, frequency of data sending, protocols that can be used to send data, value-added services of interest. The information gathered using this questionnaire has been very useful to define the interaction (from an interoperability point of view) between the value-added services and the local business platforms. Where applicable, the Context Broker should be able to send back data/information from the value-added services to the local business platforms.

The Context Broker will also provide semantic interoperability. Well-defined data format (JSON-LD) and assignation of unique IDs (URLs or URNs) for both data entities and the relationships between entities will ensure that semantic meaning can be programmatically retrieved from the data itself. JSON-LD, which is an extension of JSON, is a standard way of avoiding ambiguity when expressing linked data in JSON so that the data is structured in a format that can be analysed by machines. This method ensures that all data attributes can be easily compared when coming from a multitude of separate data sources. The FIWARE NGSIv2 information model has been evolved to better support linked data (entity-relationships), property graphs, and semantics by exploiting the capabilities offered by JSON-LD. This work has been conducted under the ETSI ISG CIM initiative and has been branded as NGSI-LD. DEDALUS pilots and partners in charge of the value-added services (WP4) will share the data models they use to be included in the Context



Broker. As anticipated in section 4.1.3, FIWARE already provide a wide range of data models as Smart Data Models but new data models can also be added by using a set of guidelines ³³.

Finally, in order to ensure interoperability with Energy Data Spaces, the OneNet connector will be deployed for exchanging/sharing data offered by Data Provider. This connector allows an easy integration and cooperation among the Data Providers and Data Consumers maintaining the data ownership and preserving access to the data sources. The Context Broker is the core component of the OneNet connector in FIWARE-based implementations of the IDS Architecture. The Context Broker offers the FIWARE NGSI APIs and associate information model (entity, attribute, metadata) as the main interface for sharing data in the Data Space. Data Providers use the APIs to publish or to expose the offered data to be retrieved by the subscribed Data Consumers.

As part of the deliverable D3.1 “DEDALUS technological enablers for multi-value and energy carrier-agnostic residential DR (1st technology release)”, a Proof of Concept (PoC) has been designed and implemented. The main purpose of the PoC is to demonstrate the feasibility of the current solution, and its functionalities. The PoC has been designed integrating the FIWARE platform with a few virtual devices to get results about platform functionalities in a straightforward way. The PoC includes a simulated demo site with only two temperature sensors sending data via Message Queue Telemetry Transport (MQTT) protocol. It has been assumed that the two temperature sensors are connected to electric devices to monitor their working status and report anomalies. The FIWARE platform is connected to the sensors, receiving the appropriate MQTT topics from the sensors, by means of the Node-RED IoT Agent configured for the PoC. The Agent works as a wrapper between the sensors and the FIWARE Context Broker. Two versions of the Context Broker have been deployed, namely Orion-v2 and Orion-LD. The FIWARE Context Broker receive continuously NGSI entities, provided by the Node-RED IoT Agent on the basis of the received measurements.

The core of the PoC is the Context Broker. It provides an NGSI-LD-based HTTP REST API for both registration and query and a subscription/notification approach. The use of NGSI-LD offers several possibilities for interaction with third-party platforms in both directions: information is queried or notified to modules responsible for propagating it for other interests, such as historical information, large-scale data management, etc.

The solution provided has been developed by means of Node-RED, for a user-friendly no-code programming approach. As visualised in Figure 15, a set of custom nodes have been developed to enrich the palette library and integrate the monitoring part with FIWARE, and also to exploit NGSI-LD for the interoperability.

³³ <https://smartdatamodels.org/index.php/guidelines/>



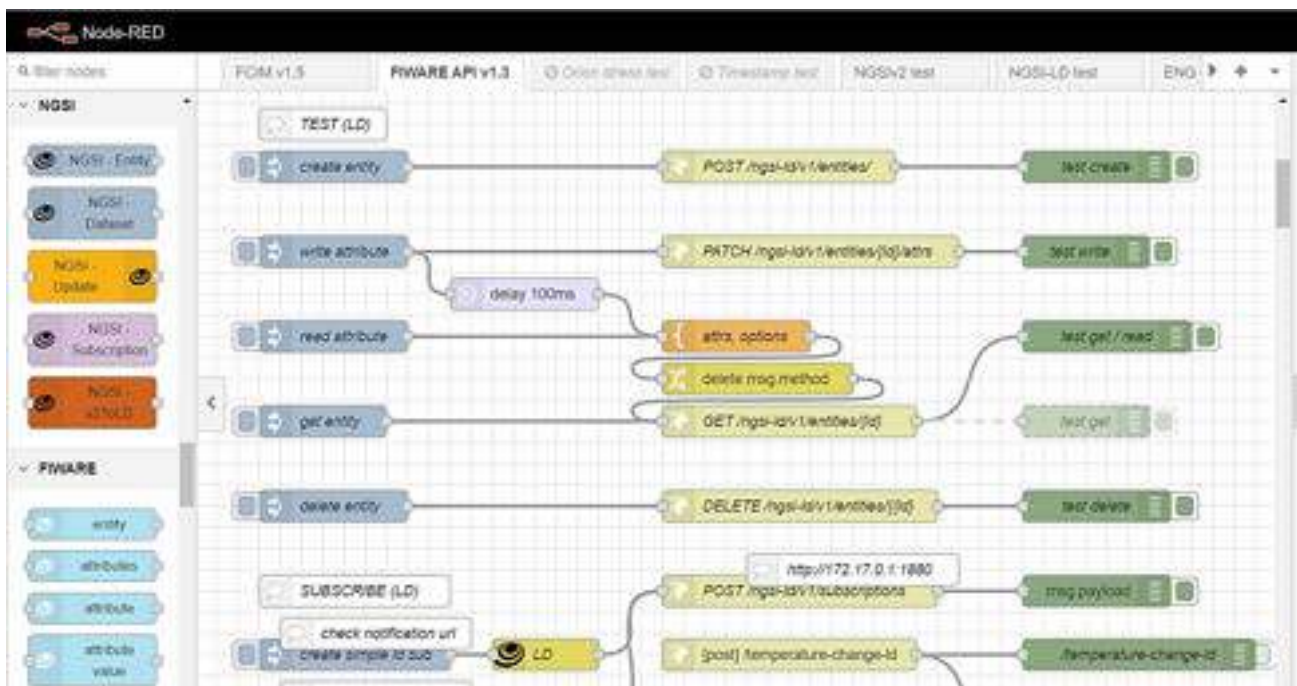


Figure 15: PoC of the DEDALUS interoperability tool

The simulated scenario simply consists of three demo smart meters, continuously providing random data about temperature, humidity and pressure. The measurements from the simulated sensors are sent and received/registered through MQTT. All the sensor data are collected and wrapped into NGSI-LD entities, using data models taken from the SAREF ontology and the corresponding JSON-LD @context file.

```
{
  "@context": [
    "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld",
    "https://schema.lab.fiware.org/ld/context"
  ],
  "id": "urn:ngsi-ld:TemperatureSensor:001",
  "type": "TemperatureSensor",
  "category": {
    "type": "Property",
    "value": "sensor"
  },
  "temperature": {
    "type": "Property",
    "value": 0,
    "unitCode": "CEL"
  }
}
```

Figure 16 isolates the portion of code in Node-RED, that is used to read and write the Orion Context Broker, by wrapping the temperature read from sensors into an NGSI-LD element.

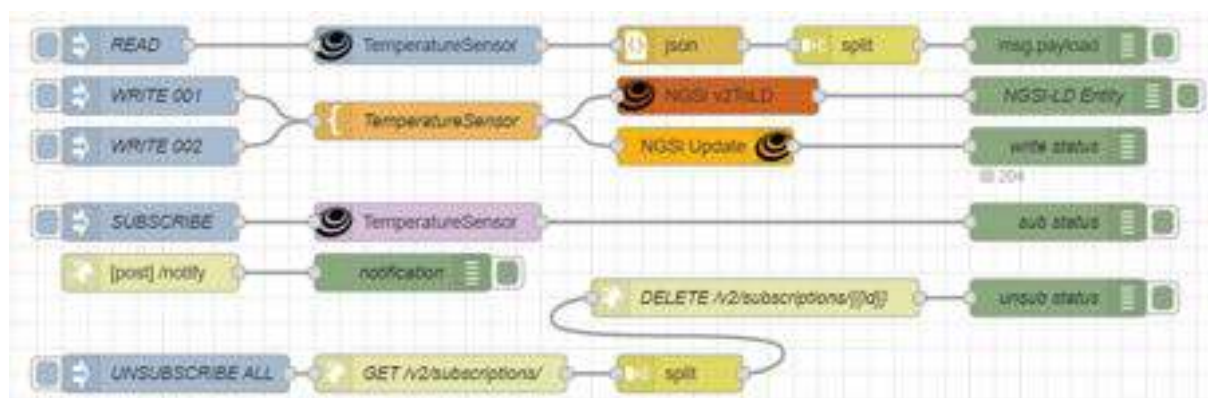


Figure 16: FIWARE Orion Context Broker reading and writing

The PoC demonstrate how data can be easily wrapped by the visual integration tool, thanks to the adoption of NGSI-LD standard. It can be exploited for demonstration and refinement for the pilots requiring a custom adaptation. The source code is stored in a Github repository and available on demand.

5. DEDALUS flexibility and DR data models and standards – 1st technology release

This section contains information on the ontology that has been defined in the scope of the DEDALUS project in order to facilitate seamless integration and interoperability among disparate systems. The work described within this section and in the previous one is encompassed in TE-3 (Residential DR/flexibility interoperability building blocks).

First of all, the standards and ontologies that have been reviewed to determine the starting point for the DEDALUS ontology are described. Then, the specific data requirements of the project are addressed. Finally, the DEDALUS ontology is presented (based on the work addressed in the previous sections).

Ontologies are able to provide a standardized way to represent knowledge and concepts within a specific domain. They help developers in modelling complex systems, defining relationships between entities, and establishing a common understanding of domain-specific terminology. Ontologies are used to model knowledge domains, and enable a common understanding of information, to facilitate interoperability and data exchange among disparate software components. By providing a shared vocabulary, ontologies allow for more effective data integration, querying, and analysis, making them essential for complex systems where consistent data interpretation is critical.

The creation of DEDALUS ontology, together with all the information addressed in the previous section regarding interoperability, constitutes TE-3 – Residential DR/flexibility interoperability building blocks (already described in other deliverables, such as D2.3 – High level requirements for the DEDALUS social, technological and business framework).

5.1. Existing ontologies

This section delves into a state-of-the-art analysis focusing on various ontologies that could be used in the scope of the DEDALUS project. This exploration of existing ontologies aims to ascertain their suitability for addressing the challenges and requirements of the project. The main objective here is to identify the most fitting ontology aligned with the project's needs, thus directing the selection process towards effective knowledge representation and system development.

The list of investigated ontologies has been done based on the know-how of different partners in charge of this activity, taking into account their previous experience in this matter.



5.1.1. Smart Applications REference Ontology (SAREF34)

SAREF ontology is focused on smart appliances, systems, meters, and spaces, facilitating interoperability among IoT solutions. In order to cover other domains, multiple extensions of SAREF being also available: SAREF4BLDG³⁵ (building domain), SAREF4ENER³⁶ (energy domain), SAREF4SYST³⁷ (system domain), SAREF4CITY³⁸ (smart cities domain).

This ontology provides building blocks for customization based on specific application needs. It is standardized and maintained under the European Telecommunications Standards Institute (ETSI).

SAREF is centered on the device concept, defined as a tangible object designed for a set of specific tasks in households, public buildings and/or offices. In order to accomplish these tasks, the device performs one or more functions, structured by categories (e.g. actuating, sensing and metering). Devices may have properties (e.g. temperature, energy...), and which can be related to specific measurements.

SAREF4ENER is an extension of SAREF aiming to provide a standardized way to describe energy-related devices, systems, and services. It is built on top of SAREF, and defines new classes, properties, and rules related to the energy domain. As an example, it provides mechanisms to describe:

- devices as solar panels, electric vehicles, smart meters...
- systems like microgrids
- services such as energy management or demand response

By using SAREF4ENER it is possible to develop applications that can easily communicate with different energy devices, systems, and services regardless the communication protocol they and/or their manufacturer.

SAREF4BLDG, is an extension of SAREF ontology that includes building, space, and zone concepts, and uses the containment relationships in order to model links between spaces. It was created based on the Industry Foundation Classes (IFC) standard for building information, although not the whole standard has been transformed since it exceeds the scope of the ontology, which is limited to devices and appliances within the building domain. Figure 17 provides a general overview of the top levels of SAREF4BLDG ontology.

³⁴ <https://saref.etsi.org/>

³⁵ <https://saref.etsi.org/saref4bldg>

³⁶ <https://saref.etsi.org/saref4ener/v1.1.2/>

³⁷ <https://saref.etsi.org/saref4syst/v1.1.2/>

³⁸ <https://saref.etsi.org/saref4city/v1.1.2/>



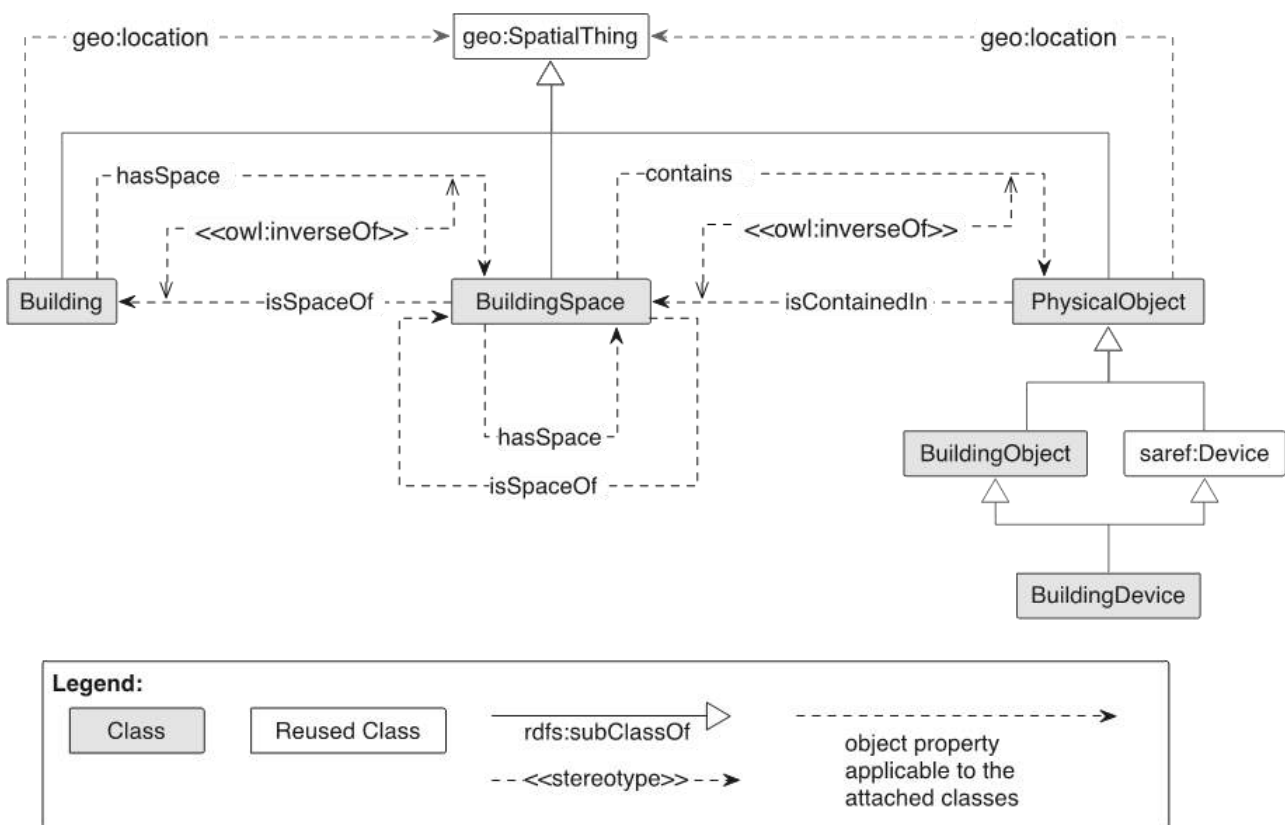


Figure 17: General overview of the top levels of SAREF4BLDG

SAREF4SYST³⁹, also built on top of SAREF and uses system connections to represent adjacency between zones. This ontology defines systems, connections between systems, and connection points to which systems may be connected. Using this set of core concepts, the topology of features of interest can be defined. SAREF4SYST can be instantiated for use in different domains. Properties of systems are usually state variables (e.g. temperature), while properties of connections are typically flows (e.g. heat flow). This ontology consists both of a core ontology, and guidelines to create ontologies following the SAREF4SYST ontology pattern. A graphical overview of this ontology is provided in Figure 18.

³⁹ <https://saref.etsi.org/saref4syst/>

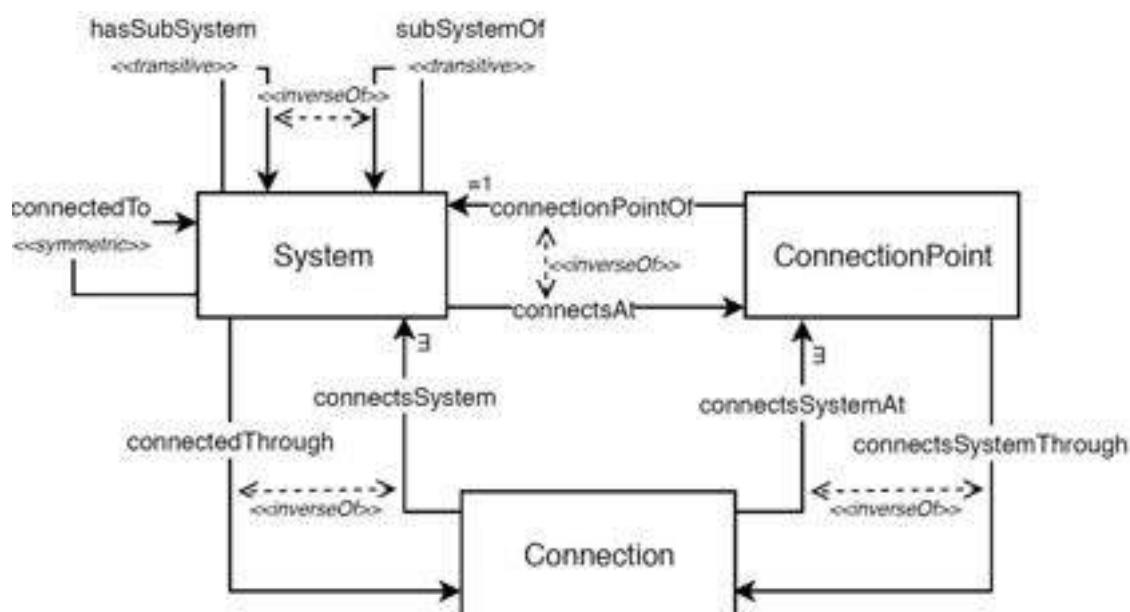


Figure 18: Overview of the SAREF4SYST ontology (source ⁴⁰)

SAREF4CITY⁴¹ is an extension of SAREF focused on the Smart Cities domain, and reuses six ontologies. Its main aim is to extend SAREF in order to create a common core of general concepts for smart city data oriented to the IoT field. An overview of SAREF4CITY is shown in Figure 19.

⁴⁰ <https://saref.etsi.org/saref4syst/v1.1.2/diagrams/Overview.png>

⁴¹ <https://saref.etsi.org/saref4city>





The OpenADR (Open Automated Demand Response) ontology was developed in order to model the OpenADR protocol. OpenADR is an open and standardized interface that assists Demand Response (DR) providers in the communication of DR signals within customer facilities in the scope of automated demand response program events. In this scope, DR providers (which communicate DR signals to customer facilities in order to generate automated demand response program events) are called Virtual Top Nodes (VTN), while customers are known as Virtual End Nodes (VEN). VEN can receive and respond to events, and they can also generate reports and control demand-side resources.

⁴³ <https://albaizg.github.io/OpenADRontology/OnToology/ontology/openADRontology.owl/documentation/index-en.html>

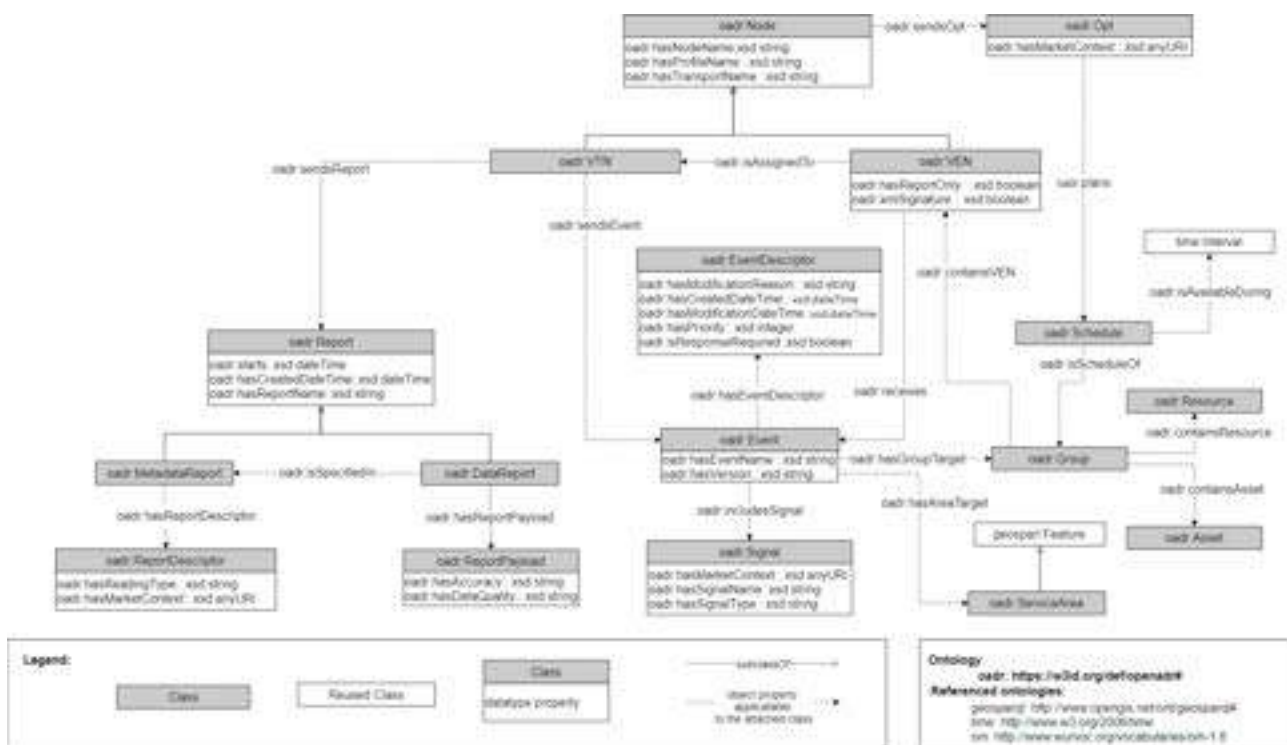


Figure 20: Top-level view of the openADR ontology (source: 44)

It is important to notice that the OpenADR ontology was developed within a European Project, and, as this project has finalized, the maintenance of the ontology has been discontinued.

5.1.3. Brick45

Brick is an open-source initiative that aims to standardize semantic descriptions of the physical, logical and virtual assets in buildings, and the relationships between them. Brick defines:

- An extensible dictionary of terms and concepts in and around buildings
- A set of relationships for linking and composing concepts together
- A flexible data model that allows for the integration of Brick with existing tools and databases.

Brick includes entities to represent equipment, locations, points, and the relationships between all of these concepts. In this sense, equipment concept describes physical devices that provide services to the building, while points represent data sources or control inputs (e.g. sensors, setpoints, commands, alarms...). Properties of equipment (such as design capacities and voltage ratings) are also considered within Brick, as well as location (which can be both physical or logical spaces grouped based on similar features (e.g. thermal zones)).

44

<https://albaizq.github.io/OpenADRontology/OnToolology/ontology/openADRontology.owl/documentation/resources/img/openADR.png>

45 <https://brickschema.org/ontology/>



Relationships can also be modelled using Brick, and they describe how classes (equipment, points, locations...) are related to each other. These relationships can be used to describe physical composition (e.g. how equipment is composed of others), logical composition (e.g. how zones are composed of rooms) or topology (e.g. how equipment is related to rooms).

It is important to notice that Point instances in a Brick model may include “foreign key” properties to link them to the data source where the data can be found (e.g. historical storage, location in a BAS network...). Concerning this issue, Brick metadata can be obtained by using queries against a Brick model, and this metadata may contain useful information to fetch or subscribe to the building telemetry.

5.1.4. domOS Common Ontology (dCO46)

This ontology (developed in the scope of the domOS project⁴⁷) focuses on providing a common structure to describe devices and services in smart home environments and energy management systems.

dCO ontology integrates the Thing Description ontology, SAREF core ontology, SAREF4ENER, and Building Topology Ontology (BOT). By using dCO, a common directory of information can be created for IoT service providers in order to let them search for building metadata. It is important to notice that other ontologies have been also included in dCO in order to make it more complete, for example the Ontology of Units of Measure (OM), which provides a standard way to represent and describe units of measurement and their relationships. OM is designed to be used in conjunction with other ontologies (e.g. SAREF4ENER, TD, BOT...), and is based on the OWL (Web Ontology Language), and RDF (Resource Description Framework) standards and is also aligned with the International System of Units (SI) and ISO standards for units of measure.

The W3C Thing Description Ontology⁴⁸ (TD) was developed by the World Wide Web Consortium (W3C) in 2020 with the objective of describing the characteristics and capabilities of IoT devices and services. It defines a set of classes, properties, and relationships that can be used to describe not only the physical and functional properties of a device, but also its communication and security capabilities. It is in line with other ontologies, such as SAREF4ENER. TD enables the development of applications that can easily integrate and communicate with different IoT devices and services without paying attention to the protocol they are connected or its specific manufacturer.

The Building Topology Ontology⁴⁹ (BOT) part of dCO, was designed to describe the physical structure and layout of buildings, their components, and the location of component properties. BOT main aim is to provide a standardized way to represent building information across different

⁴⁶ <https://www.w3id.org/dco>

⁴⁷ <https://www.domos-project.eu/>

⁴⁸ <https://www.w3.org/2019/wot/td>

⁴⁹ <https://w3c-lbd-cg.github.io/bot/>



applications and domains. It is based on OWL (Web Ontology Language) and RDF (Resource Description Framework) standards, which makes it easy to be integrated with other ontologies, such as SAREF4ENER and TD. This ontology provides seamless integration into BIM authoring tools like Revit and IFC (BOT facilitates export and import). It provides a rich functional building topology, focusing on asset management and description of the building itself, but lacks concepts to characterize systems, equipment, devices and their telemetric data. In order to represent these concepts, BOT requires alignment with other ontologies such as Brick or SAREF, and this is what has been done within domOS (as stated before, domOS ontology integrates the TD ontology, SAREF core ontology, SAREF4ENER and BOT). It is important to notice that information on how to align BOT and BRICK is available here⁵⁰.

Figure 21 shows the main classes and properties of the dCO ontology.

⁵⁰ <https://raw.githubusercontent.com/w3c-lbd-cg/bot/master/BRICKAlignment.ttl>



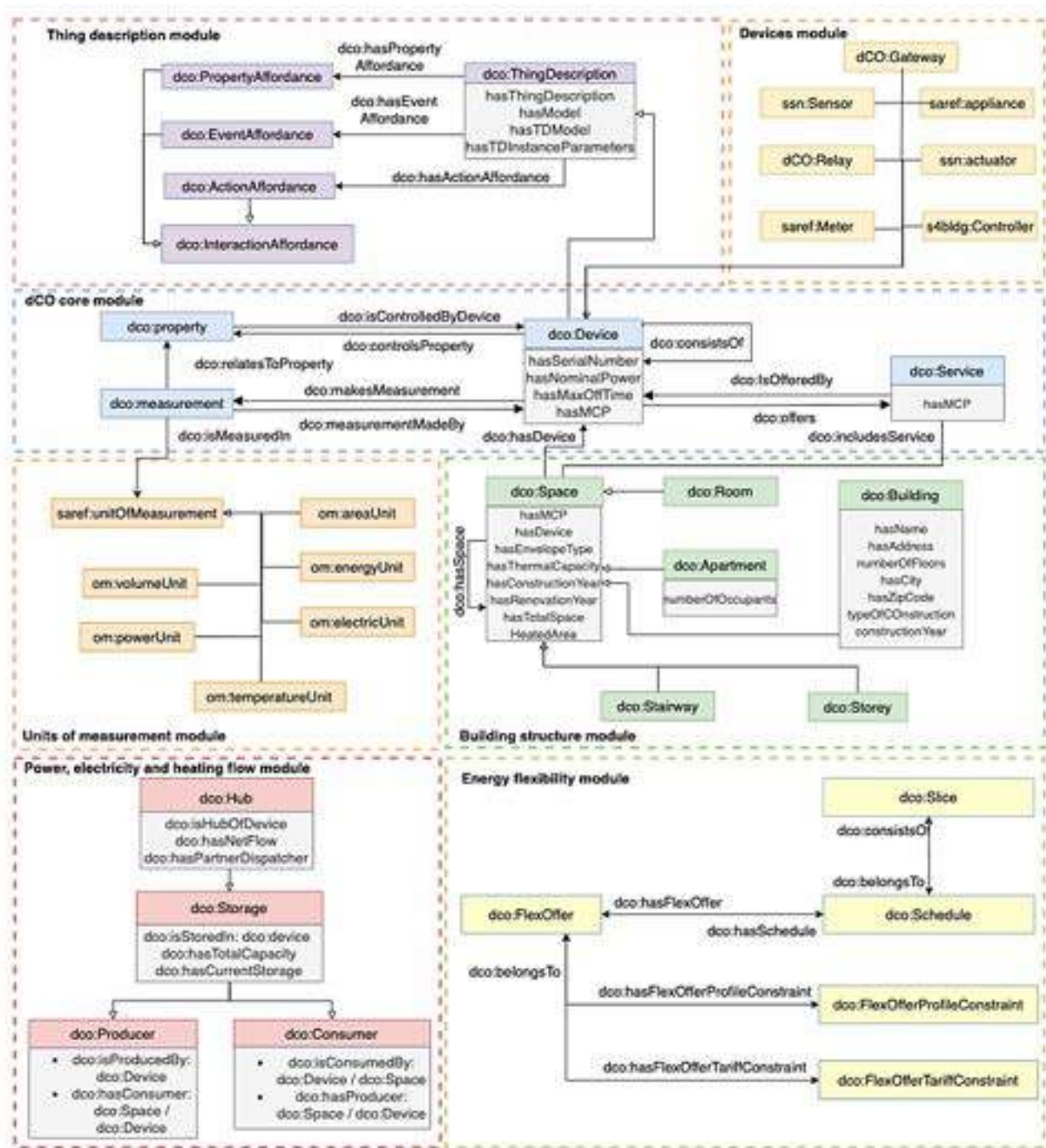


Figure 21: domOS ontology: main classes and properties

dCO has been designed following a modular approach (source: deliverable D3.5 from domOS project) in order to enhance its maintainability. dCO modules are listed below:

- dCO core module
- Building structure module
- Thing description module



- Device module
- Power, electricity, and heating flow module
- Energy flexibility module
- Units of measurements module

5.1.5. BIGG Project Ontology⁵¹

This ontology (developed in the scope of the BIGG Project⁵²) is designed to represent various aspects of smart grids systems, including energy generation, distribution, consumption, and optimization.

The BIGG project (Building Information aGgregation, harmonization, and analytics platform) was launched to demonstrate the utilization of big data technologies and data analytic techniques on buildings' life cycle across six different pilot testbeds. The BIGG ontology fully conceptualizes knowledge for BIGG Use Cases in the building domain, aligned with existing ontologies and data standards.

BIGG ontology has a modular structure consisting of core and extensions. The core includes three essential class groups, while the extensions provide additional classes and relationships (see Figure 22).

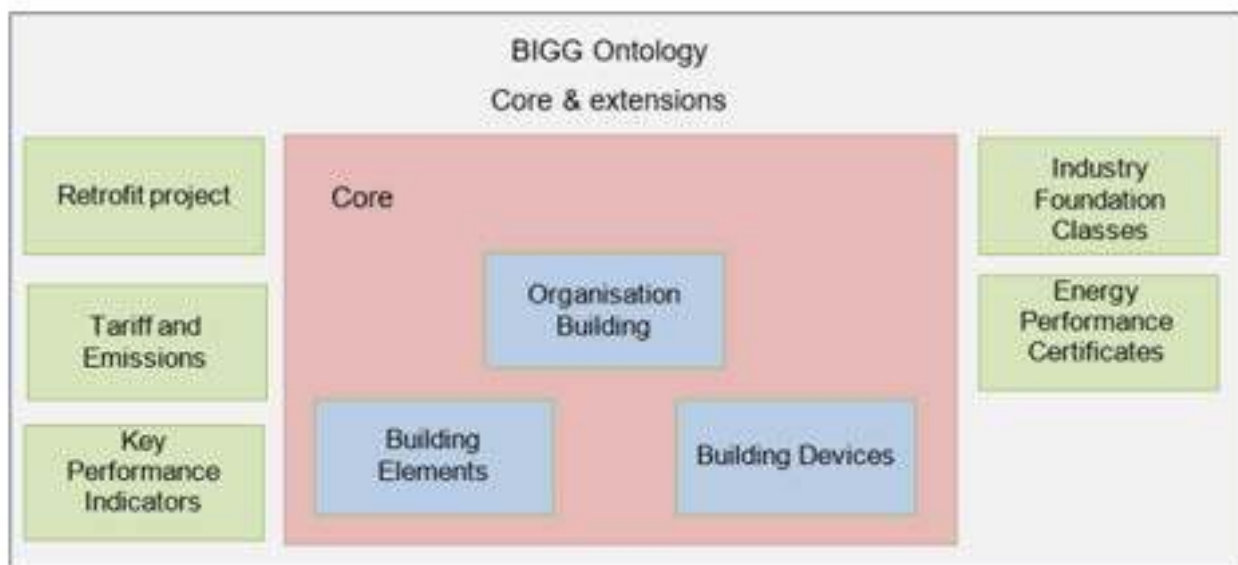


Figure 22: The BIGG ontology⁵³

⁵¹ <https://github.com/biggproject/Ontology>

⁵² <https://www.bigg-project.eu/>

⁵³ <https://build-up.ec.europa.eu/en/resources-and-tools/publications/bigg-need-harmonizing-input-data-and-ai-toolbox-revolution-context>



Similar to the dCO ontology, the BIGG ontology is based on multiple “well-known” and available ontologies, as SAREF, IFC, BOT...

As shown in Figure 22, BIGG Ontology’s core considers three blocks: building elements, organisation building and building devices, but it also provides some extensions on top of its core:

- Extension for Retrofit projects
- Extension devoted to Tariff and Emissions
- Key Performance Indicators extension
- Extension focused on Industry Foundation Classes
- Energy Performance Certificates extension

5.1.6. Comparison and final decision

It is important to note that one of the most important crucial practices in ontology development is based on reuse, so five ontologies were finally explored to determine the most suitable one to serve as the foundation for the DEDALUS ontology. A summary of these ontologies, including their pros and cons, has been included in Table 1 to provide clarity.

Table 1: Ontologies summary (pros&cons)

Ontology	Features	
	Pros	Cons
SAREF	Focused on smart appliances, systems, meters and spaces.	
OpenADR	Many extensions are available.	Some concepts needed within DEDALUS are not available (e.g. flexibility)
Brick	Focused on modelling OpenADR protocol, so it contains multiple concepts related to flexibility and demand response.	Focusing on DR, more generic concepts are missing.
BIGG	Focused on buildings.	As it is focused on buildings, other concepts are missing.
dCO	Designed to represent multiple aspects of smart grid systems, including energy generation, distribution, consumption, and optimization. Other ontologies	



have been taken as a base
(e.g. SAREF, BOT, IFC...).

Each of the described ontologies has been designed to accomplish a set of purposes which are linked, somehow, with the DEDALUS project. However, out of all reviewed ontologies, dCO seems to be the most appropriate ontology to be used in DEDALUS. As aforementioned, dCO has been recently developed (early 2023) in the scope of the domOS project, which is also related to flexibility and demand response, similar with DEDALUS project. More than that, dCO incorporates two ontologies of high interest for DEDALUS project, namely the energy domain ontology SAREF (SAREF4ENER) and building domain ontology BOT. Since BOT (built on Semantic Web technology) misses modeling support for systems such as HVAC or Lighting (which are defined in BRICK) dCO takes advantage of SAREF4ENER to cover this missing part.

If the energy flexibility module of dCO is not enough regarding DEDALUS' needs, OpenADR (or a subset of it) could be also incorporated to the final DEDALUS' ontology, but this aspect will be studied in depth if OpenADR is finally needed.

With regard to the ontology developed within the BIGG project, several interesting entities and properties have been identified, such as AnalyticsResults for baseline and benchmarking purposes, so, if finally needed, this ontology would be also considered. However, it was not selected because its scope is broader than the one envisaged by the DEDALUS project. For example, BIGG also includes information on Renovation Projects and EPCs, which are not considered within DEDALUS. Additionally, the BIGG project was concluded by the end of 2023, whereas the domOS project is still ongoing. Therefore, if any further information is needed regarding the dCO Ontology, contacting a project partner would be easier for domOS than for the BIGG project.

Last but not least, it is important to note that, although dCO was recommended at the conclusion of this ontology review, its selection was tentative. This is because, as detailed in the subsequent section, a process was planned to assess the alignment between the DEDALUS data requirements and the ontology. This evaluation was intended to ensure that dCO was indeed suitable for application within the project. Once this evaluation was completed, the final decision on the ontology to be used was made.

5.2. DEDALUS specific data requirements

As the DEDALUS project has a broad scope of data requirements (coming mainly from the technology enablers (WP3), the services to be developed (WP4) and the pilots (WP5)), a detailed analysis of all of this information requirements have been necessary in order to know what entities, attributes and relationships would be needed within the final DEDALUS ontology. Considering this matter, the following process was defined as the methodology to develop the final ontology:

- First of all, a SoTA review was done in order to choose the most appropriate ontology to be used as a starting point taking into account DEDALUS high level needs.



- Then, all the partners (mainly technology enablers (WP3), services to be developed (WP4) and the pilots (WP5)) were asked to specify their specific data requirements from their specific point of view.
- Once the aforementioned specific data requirements were available, a matching between them and the selected ontology was conducted.
- Based on the result of the matching, a proposal has been done on how to extend the starting ontology with entities/attributes/relationships from other known (and public) ontologies.

The specific data requirements received from WP3, WP4 and from the pilot sites are summarized in Table 2.



Table 2: DEDALUS data requirements summary

DATA-SET	FUNCTIONALITY	LEVEL / ASSET	DATA COLLECTION	DATA ORIGIN	DATA-SET CONTENT DESCRIPTION
Electricity consumption	TE-14	Dwelling	BMS	Smart meter	Electricity consumption from lighting, plugs and other assets
Comfort measurements	TE-14	Spaces	BMS	Ambiental sensor	Temperature, humidity, CO2
Heat pumps operation	TE-14	Heat Pump	BMS	Heat pump (Arcelex services)	Power consumption, set-point, fan speed, state...
Battery operation	TE-14	Battery	Battery Management System	Battery	Charging/discharging power, SOC, state (running, alarm,...)
PV production	TE-14	PV panels	PV inverter	PV inverter	Power generation, Max. Available power
Weather	All	Region	Web service	Weather station	Climate conditions (temp, humidity...) in a location
Dwellings thermal condition	Comfort algorithms	Building, spaces, zones	Arcelex web service	Hub Arcelex	Temperature and humidity of the dwellings
Dwelling electric consumption	Demand response	Building	PGUSA DB	Smart meter	Electric consumption
Dwelling electric consumption	Demand response	Building	Web service	Smart meter	Electric consumption
Electricity storage	Demand response	Building (storage)	BMS	Product device (storage)	Storage capacity for charge/discharge
Electricity inputs	Demand response	Building (storage)	BMS	Product device (storage)	Setpoints of battery operation
Solar production	Demand response	Building	Web service	Inverter	Solar exceedant of the city hall solar panels
Electricity consumption smart plugs	Demand response	Building	Arcelex web service	Arcelex smart plugs	Electricity consumption smart plugs and if they are on or off
Heating Data (Gas Boilers)	Flexibility services	Household	database	Heating Controller	Gas boiler consumption and several other parameters provided by the system
Heating Data (Heat pump)	Flexibility services	Household	database	Heating Controller	Gas boiler consumption and several other parameters provided by the system
Electricity consumption	Flexibility services	Household	database	Energy Meter	Electricity consumption (kWh) at household level
Indoor Air Quality Data	Comfort Services	Household	database	IAQ sensor	Data for CO2, Particle MATTER, Temperature & Humidity
Electricity consumption	Flexibility services	Building/Apartment	excel, csv		Electricity consumption (kWh) from the central meter
Electricity consumption	Flexibility services	Device/System	excel, csv		Electricity consumption (kWh) from lighting and smart devices (heatpumps, HVAC, EV charging stations, Battery Storage, home appliances, etc.)
PV Panel production	Flexibility services	PV System	excel, csv		Electricity production (kWh) from the PV system
External Weather conditions	Flexibility services	Region	Web service, excel, csv		Climate conditions (temp, humidity, solar radiation - GHI, DNI, Diffuse etc.) in location
Electricity consumption	Clustering	Building/Apartment	excel, csv		Electricity consumption (kWh) from the central meter
Heat consumption	Clustering	Device/System	excel, csv		Heat consumption
Electricity consumption	Demand response	Building	Server (PONTOON X/F)	Smart meter	Electricity consumption of apartments (lighting, plugs etc.)
Electricity storage	Demand response	Building (storage)	BMS	Product device (storage)	Storage capacity for charge/discharge
E-mobility	Demand response	Building (charging station)	BMS	Product device (charging station)	Loading capacity of e-mobility
Electricity production	Forecast electricity production	Building (photovoltaics)	Server (PONTOON X/F)	Smart meter	Electricity production of the photovoltaic system
Weather	Forecast electricity production	Photovoltaics	Web service	Weather station	Weather conditions (radiant intensity, clouds) at the location
Electricity consumption	TE-5 (DLT/Blockchain/Smart contracts solution)	Building, apartments	BMS	Smart meter	Total building electricity consumption, apartment individual consumption (not mandatory), asset level (not mandatory)
Electricity Production	TE-5 (DLT/Blockchain/Smart contracts solution)	Building	BMS	Smart meter	Electricity Production from RES (not mandatory)
Power consumption	Flexibility services	Device/Appliance	Web service / MQTT	Smart plugs	Power consumption data collected from Arcelex's smart plugs that measure the consumption from installed devices/home appliances.
Temperature	Flexibility services	Room	Web service / MQTT	Gateway	Temperature data collected from Arcelex Gateways from room/environment
Humidity	Flexibility services	Room	Web service / MQTT	Gateway	Humidity data collected from Arcelex Gateways from room/environment
Electricity consumption	Demand response	Building	Server (PONTOON X/F), BMS	Smart meter	Electricity consumption of apartments (lighting, plugs etc.)
Electricity storage	Demand response	Building (storage)	BMS	Product / device (storage)	Storage capacity for charge/discharge
E-mobility	Demand response	Building (charging station)	BMS	Product / device (charging station)	Loading capacity of e-mobility
Electricity production	Forecast electricity production	Building (photovoltaics)	Server (PONTOON X/F), BMS	Smart meter (apartment)	Electricity production of the photovoltaic systems
Comfort data	Comfort-based services	Building/Apartment	database, excel, csv	IAQ sensors	Temperature, Humidity, CO2, light level
User feedback	Comfort-based services	Building/Apartment	database, csv	online questionnaire	Thermal sensation vote from the users (user preferences)
Temperature/Humidity/CO2	Flexibility services and heating control	apartment / zone	LoRaWAN	IoT sensor	Indoor climate measurements from apartments
Radiator Thermostats	Control of heating / Flexibility services	apartment / zone / radiator	LoRaWAN	IoT sensor	Data from radiator thermostats and control settings
Electricity consumption	Flexibility services	Apartment and building	analog	Electricity Meter	Electricity consumption from apartment including apartment ventilation system
Heating consumption	Heating control and flexibility services	Apartment	analog	Heating Meter	Heating and hot water consumption on apartment level
Ventilation	Flexibility services and indoor climate	Apartment	Modbus	Ventilation system	Parameters from ventilation system and set points for air change rate
Heating consumption	Heating control and flexibility services	Building	analog	Heating Meter	Main heating and domestic hot water consumption of building
Hot water consumption	Heating control and flexibility services	Building	analog	Heating Meter	Main-hot water consumption of building
Heating controller	Control heating for whole building	Building	Modbus	Danfoss ICL	Controller data
Thermal zone	Thermostats algorithms	Building, apartment, zones	BIM	BIM	Identified thermal zones of a building either from BIM or manual
Buildings	Collection of buildings	Building	Location	Address	Collection of buildings for aggregator services

This project has received funding from European Union's Horizon Europe Research and Innovation programme under the Grant Agreement No 101103998

Once all the data requirements summarized in the previous tables were available, a matching among them and the dCO ontology was done, and the entities/attributes/relationships (mainly entities) to be added to this ontology were identified. This information is described in Table 3.

Table 3: dCO extensions needed

Entity	Sub_entity	Ontology	Description
Key Performance Indicator	Single K P I Assessment	BIGG	Entity for a single KPI calculation (not just the measurement).
	Aggregated K P I Assessment	BIGG	Entity for an aggregated KPI by using an analytical model in time series format.
Device	WeatherStation	BIGG	Entity that represents the concept of the WeatherStation, even composed by sensors.
	Inverter	--	Entity for representing the inverter that is in any solar installation.
Building	Zone	IFC/BIGG	
	City	Saref4City	Entities used to represent the region (i.e. location) to be used within the climate services
	Country		
Units of measurement	Radiation	--	Entity for the solar radiation measurement.
User	UserProfile	Interconnect	Entities that represent the concept of Person in order to obtain the behaviour, preferences and activities for any specific user.
	UserPreference		
	UserActivity		

5.3. DEDALUS ontology

This section is devoted to the final DEDALUS ontology, obtained after applying the aforementioned ontology development process (see 5.2).

As detailed in the domOS project deliverable outlining the dCO ontology (D3.5⁵⁴), the dCO core module reuses the SAREF core ontology, and allows the integration of the other modules in order

⁵⁴ <https://www.domos-project.eu/documents>

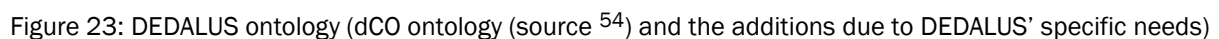


to ensure semantic interoperability. The following dCO ontology modules are the ones that will be used within the DEDALUS ontology (Things description module is not needed up to now in the scope of DEDALUS):

- Device (entities: Gateway, Sensor, Relay, Meter, Appliance, Actuator and Controller): this module represents the set of devices in an IoT ecosystem. Devices are structured in categories (subclasses), that reflect different types of devices (e.g., dco: Meter).
- dCO core (entities: Device, Service, property and measurement): this module reuses the SAREF core ontology and allows for the integration of the other modules to ensure semantic interoperability. This module is integrated with the device module using a subsumption relationship. Besides, it is integrated with the building description module and the units of measurement module respectively by using the property dco: isLocatedAt and dco:isMeasuredIn.
- Units of measurement (entities: unitOfMeasurement, powerUnit, areaUnit, energyUnit, electricUnit, temperatureUnit). This module represents all the units of measurements of the IoT devices. This module reuses the Units of Measurement Ontology (described in [4]).
A saref:Measurement represents a measurement using a unit of measurement (saref:unitOfMeasurement), which is structured in categories that reflect different types of units of measurements (e.g. om:energyUnit).
- Building structure (entities: Space, Room, Building, Apartment, Storey). This module contains the representation of building-related metadata and building topology. Some concepts from BOT have been used in the dCO ontology here. A dco:Device can be located at a dco:Space, and five types of spaces have been defined: dco:Building, dco:Storey, dco:Apartment, dco:Stairway and dco:Room. A building must contain at least one floor, which can contain at least one apartment. Additionally, an apartment can contain one or more rooms, and a room can be categorized into several types (e.g. kitchen, living room...). For the DEDALUS project, and as stated before, something to represent the concept of Zone should be added, because it is not addressed within dCO ontology.
- Power, electricity and heating flow (entities: Hub, Storage, Producer, Consumer). This module represents the flow of power, electricity, and heating between devices and spaces in a Building.
- Energy flexibility (entities: FlexOffer, Schedule, FlexOfferProfileConstraint, FlexOfferTariffConstraint). This module reuses and extends SAREF4ENER in order to model FlexOffers.

Figure 23 shows a summary of the different modules of the dCO ontology that are going to be used within DEDALUS, and also the needed additions (as shown in Table 3). All the components outside the dotted squares are additions to dCO.





Besides, and as explained before, entities to represent users were also needed, so the ones defined within InterConnect ontology have been added to the DEDALUS ontology. In this case, the entity InterConnect:User has been linked with saref:task. According to the definition of the dCO:Task entity represents "the goal for which a device is designed (from a user perspective). For example, a washing machine is designed for the task of "washing". These tasks are accomplished by a device. In this sense, the tasks involve the user perspective; therefore, the relationship with InterConnect:User is linked. Moreover, a dCO:FlexOffer should be accepted or rejected by the users. Even though this entity is not in the schema above, the user preferences and priorities should be considered. Notably to say that the dCO:Schedule is composed by dCO:Slice which divides the schedule in dCO:Slot to apply a demand response result. At the end, the flexibility would be applied to a device with a specific task (e.g., Comfort, Cooling, Lighting or Space Heating, among others), related thus to the user entity.

As ontology refinement is required, additional information will be incorporated into upcoming WP3 deliverables, providing further details on how the ontology will be used by other services.

Additionally, once the ontology is finalised, the next steps will include initiating its use to develop the specific data models for each pilot site. This will involve describing metadata, adding references to measurement and control points and generating specific queries to extract information from the ontology once it is populated.

With the aforementioned pilots' data models, the services would be able to semi-automatically configure themselves in order to be executed based on the specific features of each demo-site.



6. Multi-energy flexibility modelling for data-driven residential DR – 1st technology release

TE-4, led by NTUA, is dedicated to devising a methodology for estimating the flexibility of heterogeneous multi-energy loads within buildings or districts. The primary goal is to develop a tailored flexibility prediction model for Demand Response (DR) services, thereby optimizing energy efficiency. Leveraging advanced Long Short-Term Memory (LSTM) models, the methodology focuses on predicting energy consumption and production dynamics, including baseline levels and maximum and minimum flexibility, particularly on weekdays. The overarching goal is the development of a flexibility tool which will serve as a comprehensive flexibility modeling solution, integrating and expanding upon these predictive models. Additionally, it will incorporate features to facilitate comparison and evaluation of aggregation levels and the effectiveness of aggregated flexibility across the various pilots in the project.

6.1. Overview

The field of energy flexibility has made significant advances in recent years, yet there are still gaps in fully characterizing and quantifying this essential aspect of energy systems. While existing literature provides valuable insights, there are notable areas that require further exploration. Reynders et al. [5] have contributed by evaluating various methods for quantifying building energy flexibility, with a specific focus on thermal storage systems. However, this focus limits the scope of their analysis. Similarly, Bampoulas et al. [6] proposed an ensemble learning framework that considers the flexibility of both thermal and electrical systems within residential buildings. Their framework, encompassing components such as heat pumps, photovoltaic systems, and battery units, offers potential for electricity aggregators to optimize energy flexibility across building portfolios. Chen et al.'s [7] classification of flexibility measures has added depth to our understanding, although there remains a noticeable gap in reviewing flexibility quantification at broader levels, including whole-building and building cluster levels. Despite these valuable contributions, a comprehensive review spanning diverse applications and scopes is still lacking. To address this gap, TE-4, led by NTUA, is undertaking the task of developing a multifaceted approach to quantifying energy flexibility. Their aim is to encompass considerations at both the building and device levels, providing a more holistic understanding of energy flexibility across various applications.

6.2. General Architecture

The general architecture (see Figure 24) comprises two primary flows: one focusing on building-level analysis and another on device-level analysis. These flows aim to predict electricity consumption and production, calculate flexibility metrics, and deliver insights through digital interfaces.



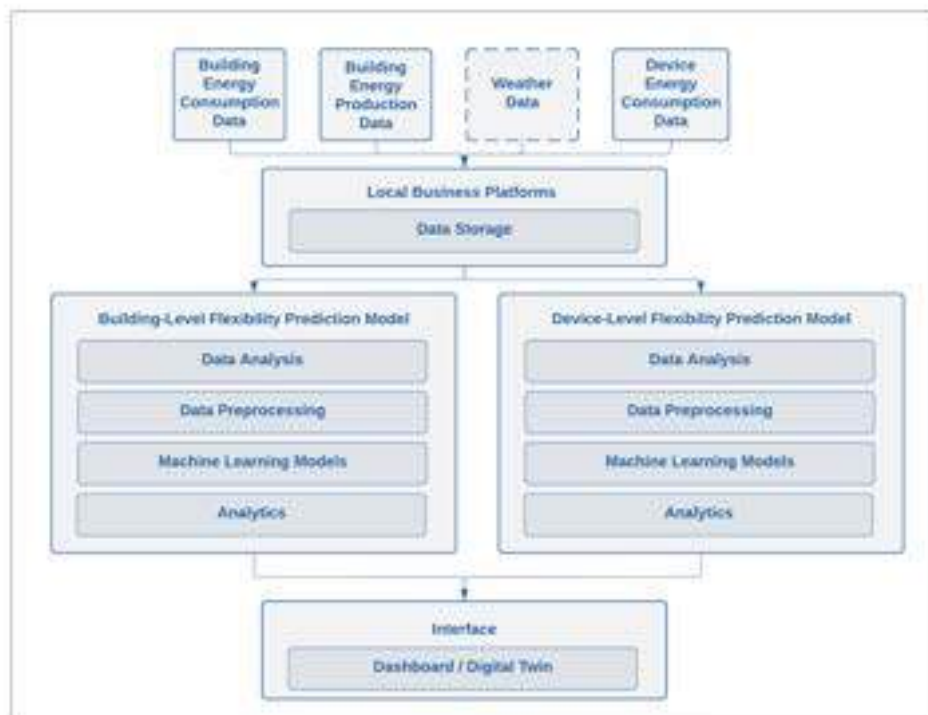


Figure 24: Workflow of the proposed methodology

6.2.1. Building-Level Analysis:

At the building level, the process begins with aggregating data from various sources, including building energy consumption data, building energy production data and weather data (where available). These datasets are stored in the local business platforms of the pilots, and API calls will be facilitated to retrieve the necessary information. The first step in the building-level flow involves data analysis and preprocessing to ensure data quality and suitability for modeling. Exploratory data analysis (EDA) is conducted to understand data distributions and patterns. Subsequently, Long Short-Term Memory (LSTM) models are employed for predicting the next day's electricity consumption and production of the building. After prediction, the flexibility baseline, maximum, and minimum flexibility levels are calculated to determine the upper and lower ranges of consumption and production for each hour of the target day. Analytics are performed to derive insights from these flexibility metrics. The output of this process is then delivered to an interface, which may include the building's digital twin or a dedicated dashboard provided by the service.

6.2.2. Device-Level Analysis:

At the device level, a parallel approach is adopted, focusing on predicting the electricity consumption of individual devices for the forthcoming day. This involves data analysis, preprocessing, and the application of machine learning models tailored to each device. Following prediction, the device electricity consumption baseline, maximum, and minimum flexibility are



calculated for each device, providing insights into their operational flexibility throughout the target day. Similar to the building-level flow, the output of this process is delivered to the pilots' digital twin or to a dedicated dashboard developed for this purpose.

6.3. Building-Level Flexibility Modelling Description

Following the comprehensive development of the methodology, this initial technology release is focused on testing the aggregated building flexibility modelling aspect. Through rigorous validation processes and extensive testing, the functionality of the Building-Level Flexibility Prediction Model was evaluated within the Austrian pilot conducted by OurPower, GreenPoint, situated in Wiener Neustadt. The pilot was selected due to its robust infrastructure, including solar panels, battery storage, EV charging stations, and a Building Energy Management System (BEMS). Leveraging two years of one-hour granularity data, encompassing both consumption and production data from dedicated PV panels, NTUA conducted thorough analyses to accurately assess the building's flexibility. This testing phase serves as a critical milestone, laying the foundation for subsequent refinements and optimizations, ultimately enabling more effective energy management practices within the pilot site and beyond.

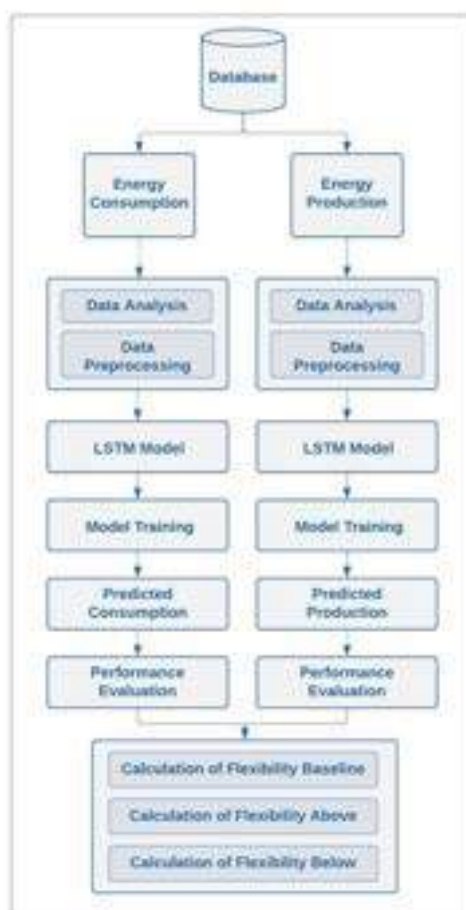


Figure 25: Workflow of the building-level methodology



The initial phase of the proposed methodology, depicted in Figure 25, focuses on predicting the consumption and the production of the pilot building equipped with a photovoltaic (PV) panel as an energy source for the upcoming day. To achieve accurate predictions, we employ LSTM models, a type of recurrent neural network (RNN) known for effectively handling variability in time-series data. LSTM models are particularly suited for capturing and learning long-term patterns within the context of energy consumption and production time series ensuring a more precise estimation of the building's energy dynamics.

To develop the models firstly there is the data preprocessing phase. The structured data-related issues, such as missing and duplicated data are unavoidable during the data collection, and they can negatively affect the model's performance if not appropriately corrected. Data cleansing is therefore conducted in order to detect and fix error records in regard to the hourly electricity consumption and production data. Various data cleaning processes were employed. In our case, the primary procedures involved removing duplications and addressing missing values. The dataset is loaded as a data frame using pandas ⁵⁵, a famous data manipulation and analysis library. After that, data inconsistencies can be automatically detected using pandas-supported functions. Records with values stuck and repeated more than three times were removed. For null values, a different approach was employed being replaced by the mean of neighboring values to maintain data integrity and coherence. This ensures that the dataset remains reliable for subsequent analysis and modeling. The dataset is then segmented into distinct time-series, focusing specifically on weekdays to capture the different energy consumption and production variations.

Feature engineering also plays a pivotal role, as it provides Machine Learning (ML) algorithms with informative and discriminative features that can help them better understand the underlying patterns and relationships in the data. For the consumption prediction we have identified and incorporated relevant time factors such as the month, hour and day of week. These features are crucial for building a comprehensive understanding of the energy consumption dynamics. As for the production dynamics we have incorporated the month and hour as features. One main process in the feature engineering is the standardization/normalization. Structured data can negatively impact regression model fitting and learned functions, particularly when numerical features with varying scales are involved, leading to bias. To mitigate this, normalization or standardization techniques are essential to normalize input features. Common approaches include min-max normalization and standardization. In our dataset concerning peak energy consumption and production at specific periods, outliers play a crucial role during training. Min-max normalization can help mitigate the impact of outliers by scaling features to a range between 0 and 1. Meanwhile, standardization adjusts features to have a mean of zero and a standard deviation of 1, aiding in improving model performance.

Data partitioning is a critical preparatory step preceding model training and evaluation. Following preprocessing, the dataset is divided into two subsets: the training set and the testing set. The training set is employed to train and refine the model, while the testing set serves to gauge the model's performance under diverse conditions. The dataset was portioned such that 80% of the

⁵⁵ <https://pandas.pydata.org/>



data was assigned to the training set, while the remaining 20% comprised the testing set. The LSTM model is configured to capture long-term dependencies and optimizing key hyperparameters. To capture temporal dependencies, a common lookback window of 24 hours was applied to both consumption and production models.

Training and validation processes involve meticulous dataset splitting and cross-validation to ensure robust model performance. Three standard evaluation metrics were computed, which included the coefficient of determination (R^2), mean squared error (MSE), and mean absolute error (MAE) in order to evaluate the electricity consumption and production forecasting. MSE is computed by averaging the squared difference between the predicted values and actual values for all the training samples. On the other hand, MAE is the average of the absolute differences between the predicted values and true values. While MSE measures the standard deviation of residuals, MAE calculates the average of the residuals in the dataset. R^2 is computed by determining the proportion of the dependent variable's variance predicted by the algorithm. The lower the MSE and MAE scores, the better the model's performance. A higher R^2 value is indicative of better performance..

6.3.1. Building-Level Electricity Consumption Modelling Description

The model for the prediction of the consumption, constructed through the Keras Sequential API, consists of three LSTM layers with progressively decreasing units (40, 20, 10), employing Rectified Linear Unit (ReLU) activation functions. A Dense layer with a single neuron serves as the output layer for forecasting purposes. The model is compiled with the Adam optimizer and mean squared error loss function, fitting the data over 50 epochs with a batch size of 16. To enhance its generalization capabilities, early stopping is implemented, allowing the model to cease training if the validation loss does not improve for five consecutive epochs, with the restoration of the best weights. This approach aims to capture and leverage sequential dependencies within the input consumption data, providing a robust framework for predicting energy consumption on weekdays. Figure 26 (together with Table 4) illustrates the performance of the LSTM model for energy consumption.

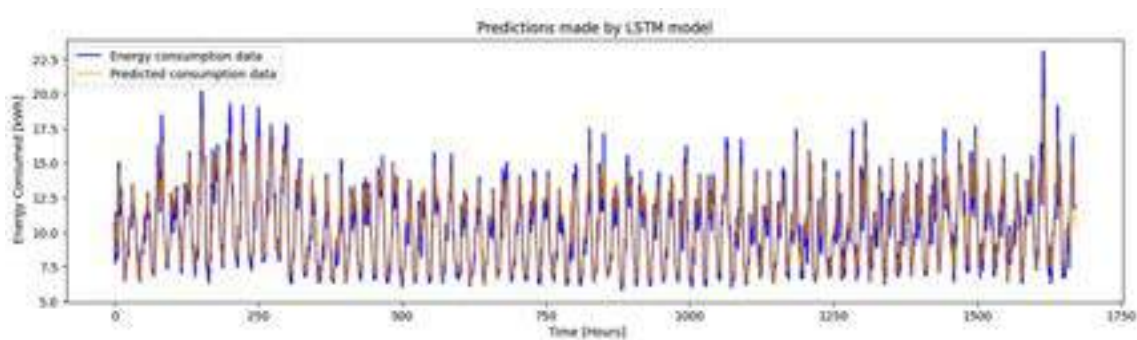


Figure 26: Performance of the LSTM model for energy consumption



Table 4 Electricity Consumption Prediction Performance

MAE	MSE	R ²
1.06	2.23	0.74

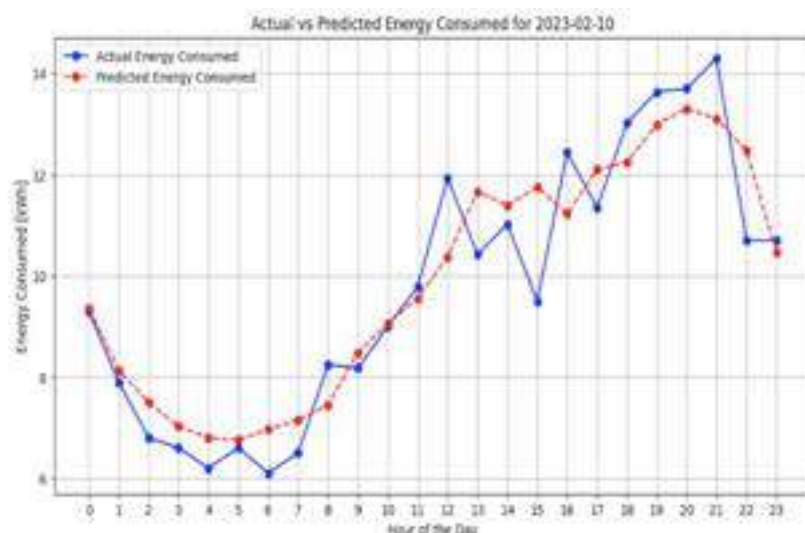


Figure 27: Comparison between Actual and Predicted Energy Consumption

Figure 27 illustrates the comparison between energy consumption values alongside the corresponding values forecasted by the predictive model. The figure provides valuable insights into the model's performance by showcasing the alignment or divergence between the predicted and actual consumption values, aiding in the evaluation of the model's efficacy in forecasting energy consumption trends.

6.3.2. Building-Level Electricity Production Modelling Description

Similarly, the model for the prediction of the production of the PV panel, implemented using the Sequential API in Keras, comprises three LSTM layers with diminishing units (40, 20, 10) and employs the ReLU activation function to capture sequential dependencies in the input data. The final layer consists of a Dense layer with one neuron, serving as the output for forecasting a singular value. The model is compiled with the Adam optimizer and utilizes the mean squared error loss function. To enhance generalization capabilities and mitigate overfitting, early stopping is again incorporated with a patience of 5 epochs and the restoration of optimal weights. Training the model on weekday energy production data involves 50 epochs with a batch size of 16, and a validation split of 20% ensures robust evaluation. Figure 27 depicts the behavior of the LSTM model designed for predicting energy production.



6.3.3. Calculation of Flexibility

In the next stage of the methodology, the calculation of the baseline energy consumption and production per hour is conducted for the specific day under examination. The analysis takes into account the day of the week for which the prediction is being made. This process allows for the establishment of a stable reference point for comparing energy consumption and production on the given day and identifies flexibility concerning this baseline. To estimate this baseline, the following procedure is followed:

- **Step 1:** Identify all days (e.g., Mondays if the day under prediction is Monday) within the last 30 days from the target day.
- **Step 2:** For each hour of the day, calculate the average consumption/production from all these days. This average serves as the baseline for the corresponding hour.

Next step of the methodology is the computation of Max Flexibility (Flexibility Above) and Min Flexibility (Flexibility Below) which is conducted for the specific day under examination, following the outlined procedure:

- **Step 1:** Identify all days (e.g., Mondays if the day under prediction is Monday) within the last 30 days from the target day.
- **Step 2:** For each hour of the day, determine the maximum consumption/production from all these days. This maximum value represents the Max Flexibility for the corresponding hour.
- **Step 3:** For each hour of the day, determine the minimum consumption/production from all these days. This minimum value represents the Min Flexibility for the corresponding hour.

This systematic process allows for the identification of the maximum and minimum flexibility levels, providing crucial insights into the potential variations in energy consumption and production during the target day. Max Flexibility signifies the upper limit of potential adjustments, while Min Flexibility represents the lower limit, aiding in the evaluation of the system's adaptability.

6.3.4. Visualization and Analysis

Visual representations, including graphs illustrating baseline, flexibility levels, actual, and predicted consumption and production values, are provided to facilitate a comprehensive analysis. These visual aids offer invaluable insights into energy dynamics, enabling informed strategies for energy management and optimization.



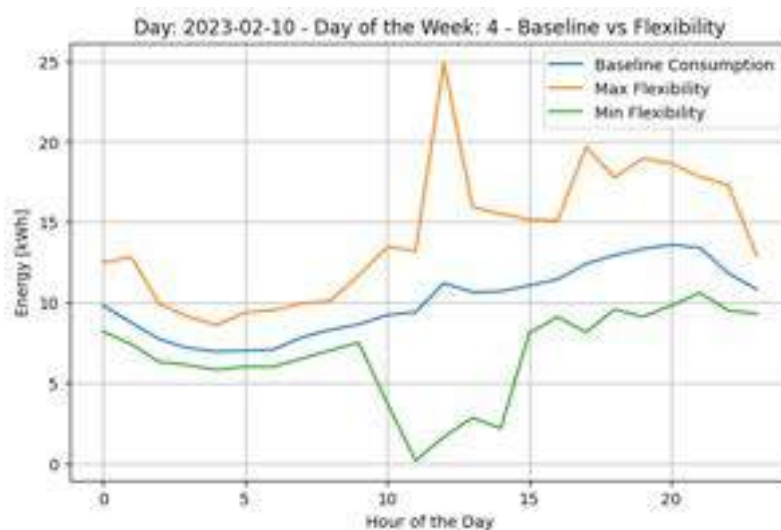


Figure 28: Baseline and Min/Max Flexibility for Energy Consumption for a weekday

Figure 28 represents the baseline energy consumption calculated for the target day Friday February 10th 2023, providing a stable reference point for comparison. Additionally, it includes the Min and Max Flexibility levels, indicating the lower and upper bounds of potential adjustments in energy consumption. Similarly, Figure 29 showcases the same values for the target day Saturday February 11th 2023. These visual representations aid in understanding the range of expected variability in consumption and highlights the flexibility limits within which the actual consumption values are likely to fluctuate. The figure serves as a valuable tool for assessing the adaptability and resilience of the energy consumption system, providing insights into the potential adjustments that can be accommodated based on historical patterns.

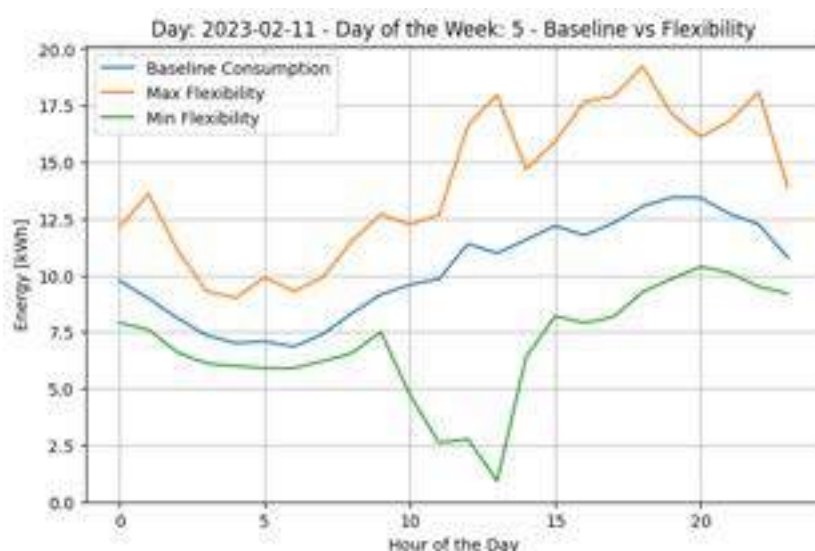


Figure 29: Baseline and Min/Max Flexibility for Energy Consumption for a weekend day



Figure 30 offers a comprehensive overview of multiple factors influencing energy consumption on the target day, Friday, February 10th, 2023. It encompasses actual consumption values, predicted values, baseline, and Min/Max Flexibility levels. Specifically, the Baseline of consumption is highlighted in green, Max Flexibility in orange, and Min Flexibility in mauve. Additionally, predicted consumption values are marked in red, while actual consumption values for the given day are represented in blue. Expanding upon this analysis,

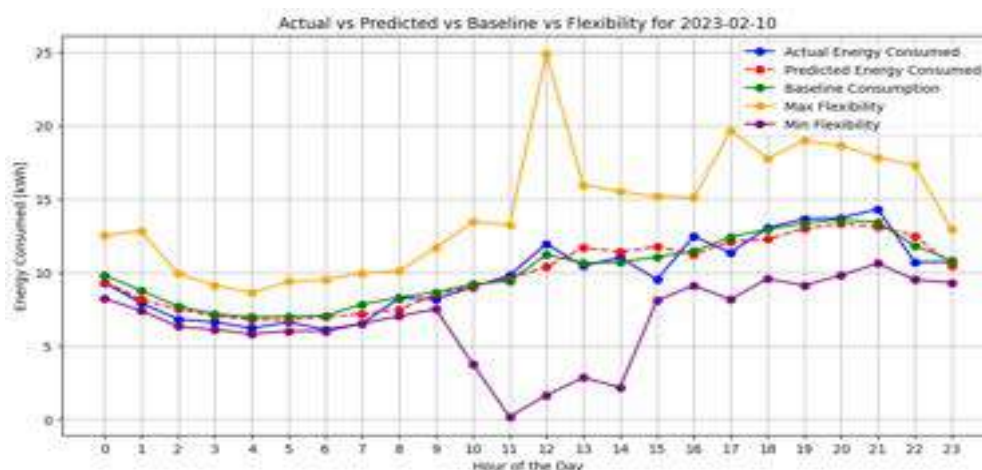


Figure 30: Overview Graph for Energy Consumption for a weekday

Figure 31 presents a parallel representation of comparable factors for Saturday, February 11th, 2023. This allows for a direct comparison between the two days, aiding in understanding patterns and variations in energy consumption over the weekend period.

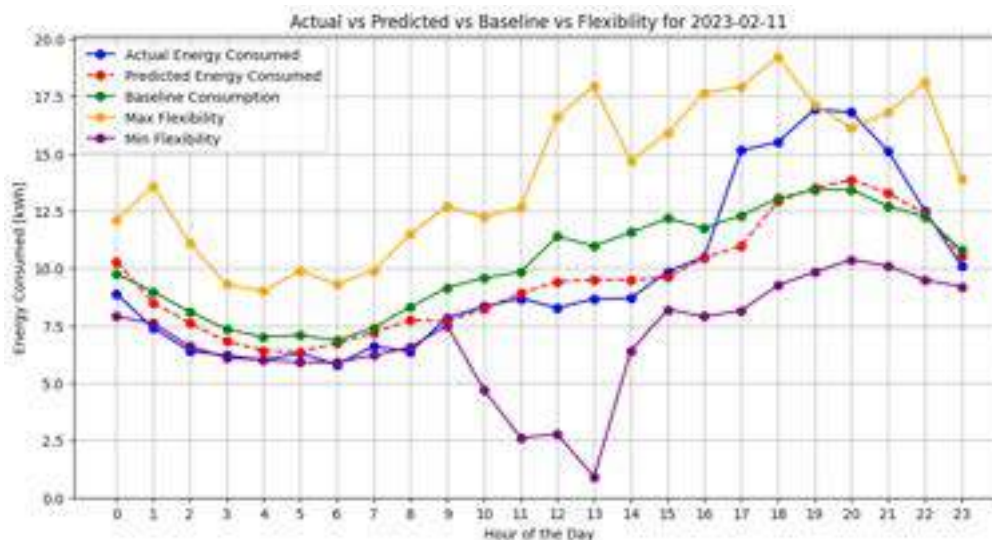


Figure 31: Overview Graph for Energy Consumption for a weekend day



Figure 32 illustrates the baseline energy production computed for the specified date, Friday, February 10th, 2023. It also includes the Min and Max Flexibility levels, denoting the potential range of adjustments in energy production with the lower and upper bounds. This depiction aids in understanding the expected variability in energy production and the flexibility within which actual production values are likely to fluctuate.

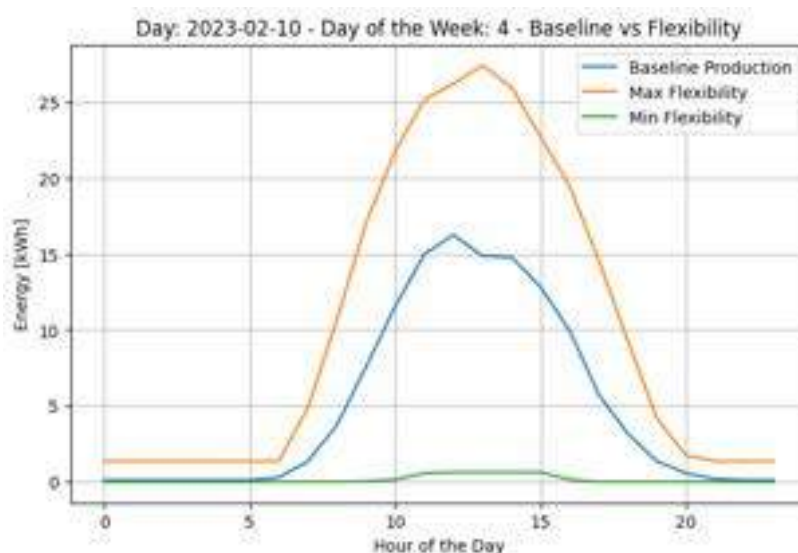


Figure 32: Baseline and Min/Max Flexibility for Energy Production for a weekday

Similarly, Figure 33 provides a corresponding representation for Saturday, February 11th, 2023. This allows for a comparative analysis between the two days, facilitating insights into the patterns and variations in energy production over the weekend period.

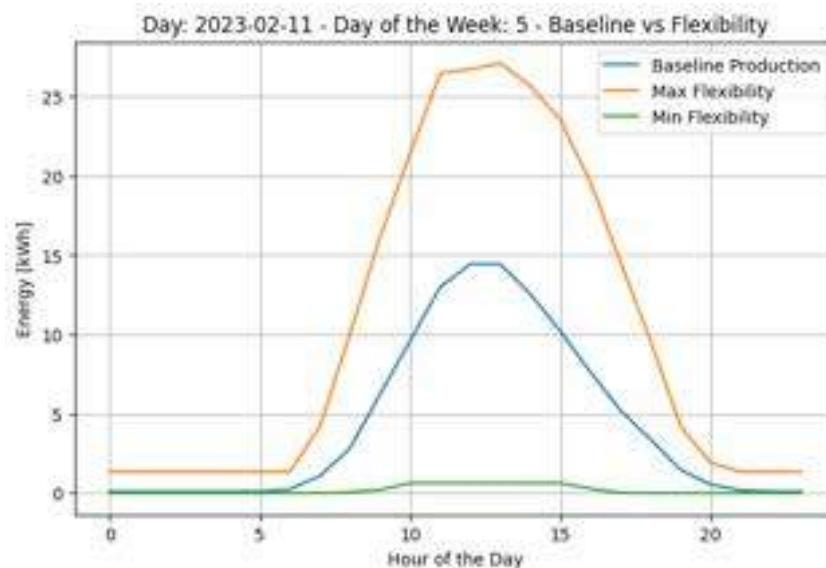


Figure 33: Baseline and Min/Max Flexibility for Energy Production for a weekend day



Figure 34 integrates key information related to both energy consumption and production, providing a holistic perspective on the dynamics of the energy system. The graph employs distinct colors to differentiate between different components, with, for instance, green representing consumption Baseline, orange denoting Max Flexibility for consumption, mauve indicating Min Flexibility for consumption, yellow representing Baseline production, brown denoting Max Flexibility for production and pink indicating Min Flexibility for production. The inclusion of both consumption and production data in a single graph facilitates a unified understanding of the overall energy landscape, allowing stakeholders to assess the interplay between these two critical facets of the energy system.

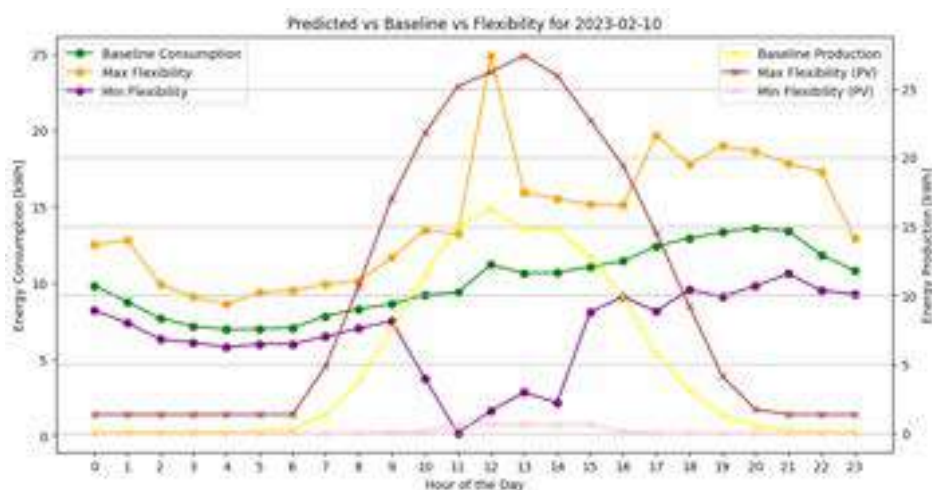


Figure 34: Consolidated Graph for both Energy Consumption and Production for a weekday

Likewise, Figure 34 provides a comparable representation for Saturday, offering insights into the energy consumption and production dynamics specific to that day. This parallel visualization allows for a direct comparison between Friday and Saturday, aiding in the assessment of any variations or trends across the two days.

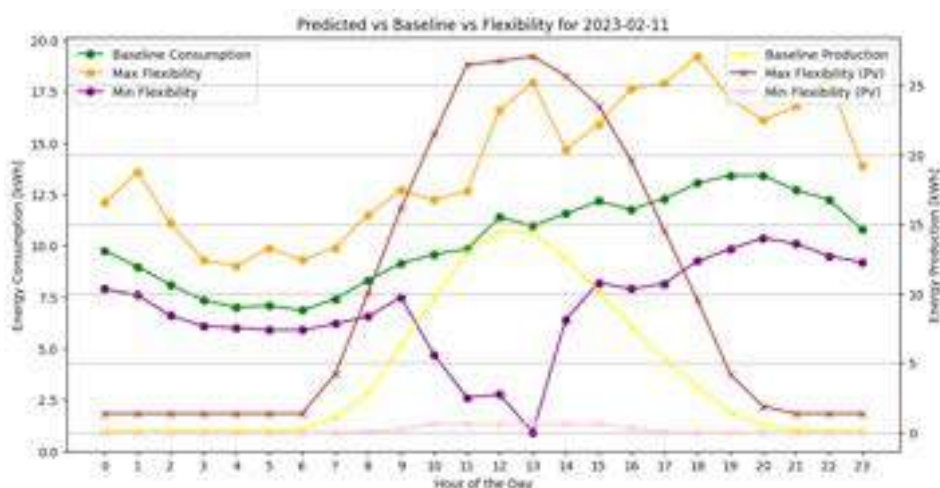


Figure 35: Consolidated Graph for both Energy Consumption and Production for a weekend day



Figure 36 offers a unified representation of actual energy consumption and production values, as well as predicted values, providing a comprehensive overview of the energy system, enabling stakeholders to assess the interdependencies and trends between these essential components.

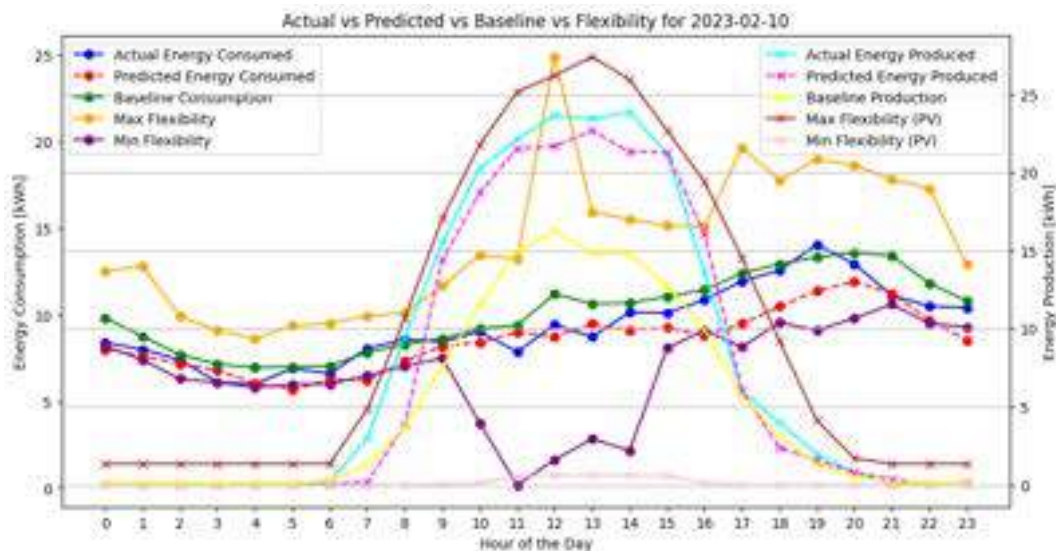


Figure 36: Comprehensive Graph combining both Energy Consumption and Production along with Predictions for a weekday

Similarly, Figure 37 offers a parallel representation for Saturday, providing insights into the energy consumption and production dynamics specific to that day. This allows for a direct comparison with the data presented in Figure 37, aiding stakeholders in assessing variations and trends across different time periods, such as weekdays and weekends.



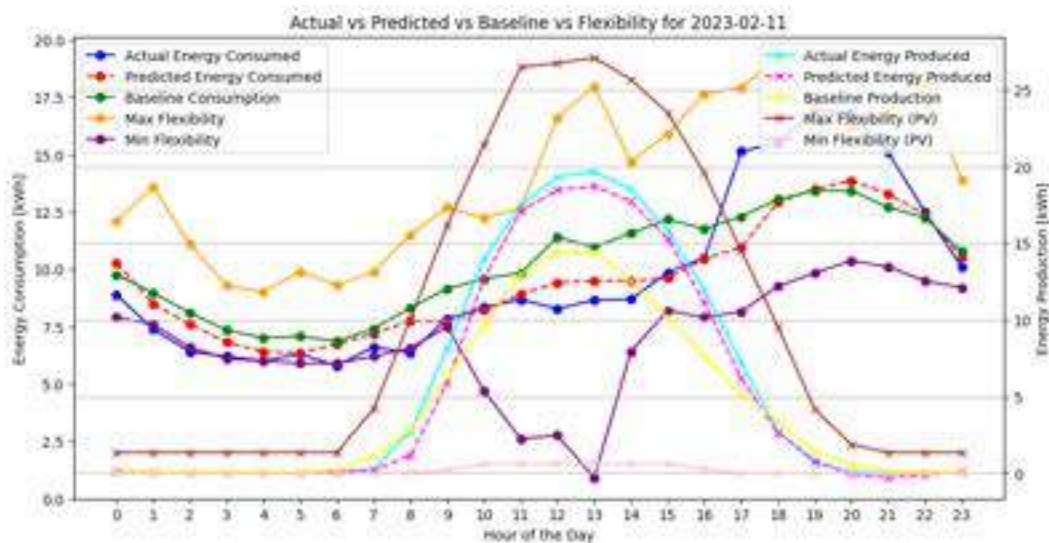


Figure 37: Comprehensive Graph combining both Energy Consumption and Production along with Predictions for a weekend day

6.4. Next Steps

In the upcoming second technological release, the methodology will undergo finalization through the incorporation of quantification of potential flexibility. This enhancement will lead to a better understanding and quantification of flexibility, enabling stakeholders to make more informed decisions regarding energy management. Specifically, the methodology will introduce specific Energy Flexibility Indexes (EFIs) to address both energy consumption and production. The Upper EFI will denote the percentage of potential increase in consumption/production compared to the Baseline for each hour of the target day, while the Lower EFI will reflect the percentage of potential decrease in consumption/production compared to the Baseline for each hour of the target day. These indices will provide a quantitative measure of the system's adaptability, offering insights into potential adjustments, both upward and downward, in energy consumption/production concerning the established baseline. Moreover, the methodology will be expanded to cover other pilot use cases, broadening its applicability and relevance.

The technical implementation of these enhancements will be executed, culminating in the delivery of results through a dedicated dashboard. The dashboard, envisioned as a centralized platform, will empower stakeholders to access and interpret insights derived from the methodology, facilitating seamless decision-making and optimizing energy management practices across diverse scenarios. Users will be able to select the type of flexibility they wish to calculate; aggregated (building-level) or device-level flexibility. Then, they will be able to upload their data through two main channels:

- **CSV Upload:** Users will be able to upload their data using a CSV file format. A sample CSV will be provided as reference, guiding users on the required format and structure of the data.



- **API Integration:** For users with existing business platforms supporting API calls, direct integration will be available. This streamlined approach ensures seamless data transfer and integration with the dashboard.

The dashboard features will include:

- **Flexibility Calculation:** The dashboard will facilitate the calculation of flexibility metrics based on user-uploaded data. This includes baseline, maximum and minimum flexibility levels and EFIs.
- **Visualization:** The dashboard will provide visual representations of calculated flexibility metrics, allowing users to gain insights into energy consumption and production dynamics.
- **Customization:** The dashboard will offer customization options, allowing users to tailor visualizations and analyses to their specific preferences and requirements.
- **Accessibility:** The dashboard will be designed with user-friendly interfaces and intuitive navigation, ensuring accessibility for users with varying levels of technical expertise.



7. Blockchain-based interoperability for flexibility tokenization & DR settlement – 1st technology release

TE-5 technological enabler for blockchain-based interoperability for flexibility tokenization & DR settlement (BIFTS) consists of several interconnected components that facilitate decentralized DR (Demand Response) and building flexibility aggregation by employing a blockchain-based distributed ledger to securely store and digitally fingerprint energy demand and/or production data. In the following subsections we will describe the 1st version for BIFTS highlighting the main data model, the general architecture and the implemented techniques and algorithms.

7.1. General architecture

The main actors of BIFTS are: (i) buyer – building manager and (ii) sellers – apartment owners who can place energy bids and offers on the market and have them matched into energy transactions. However, the design is flexible enough to be easily adapted to other contexts such as an energy community, in which case the buyer being the community manager and the sellers the community members (i.e. prosumers).

Figure 38 presents the architectural view for BIFTS. The energy data from IoT devices is integrated into scalable database compliant with the DEDALUS core ontology to ensure interoperability. A blockchain-based distributed ledger is leveraged to digitally fingerprint using hashes and securely storing the energy data on hourly bases. This architecture allows us to operate in a decentralized manner and to record information on a blockchain network in an immutable and tamper-proof manner, without any single entity managing it, which in turn, enhances trust among participants through this technology's main features, transparency, and traceability. A forecasting component utilizes monitored data from the database to predict energy demand and flexibility for the day-ahead. The prediction information is used by the building manager and apartment residents to define bids for energy flexibility (i.e. manager) and offers for energy flexibility (residents) on the energy flexibility market implemented using smart contracts. A P2P trading mechanism is defined and used at market level for level flexibility aggregation to meet the bid of the manager. As a result, transactions among building manager and residents are generated and stored in the energy ledger and used to define decentralized DR programs by injecting DR signals into the smart contracts associated with apartments. A DR settlement component uses monitored data to trigger the appropriate financial remuneration for flexibility providers, according to fulfilling or not their energy delivery commitment. Smart contracts compare the monitored data against these commitments, and generate the actual settlement based on these values. The platform also offers tokenization for flexibility integration, encompassing complex schemes that can go beyond just financial remuneration.

The main flow of functionalities for the platform is given briefly as steps below:

1. Building managers can place a bid for energy flexibility on a blockchain trading platform.



2. Apartment residents can place offers for energy flexibility on a blockchain trading platform.
3. A flexibility trading component selects, and aggregates flexibility offers to match the flexibility bid.
4. Flexibility transactions are generated and stored on the blockchain based on the matching.
5. Energy monitored data is used during the next day to settle transactions based on the actual delivery of flexibility and to distribute tokens to apartment residents' wallets.

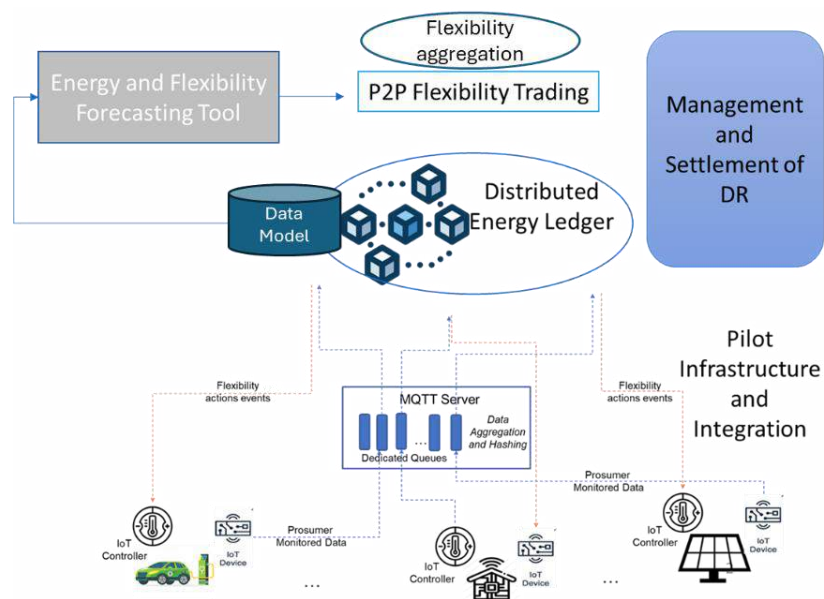


Figure 38: Architectural overview of BIFTS.

7.2. DEDALUS ontology compliant data model

The participants, asset configurations as well as monitored and predicted energy data are organized in the data model depicted in Figure 39. The main goal is to efficiently manage this data and support various configurations of the assets while adhering to the DEDALUS core data model/ontology to ensure interoperability with other project components and services.



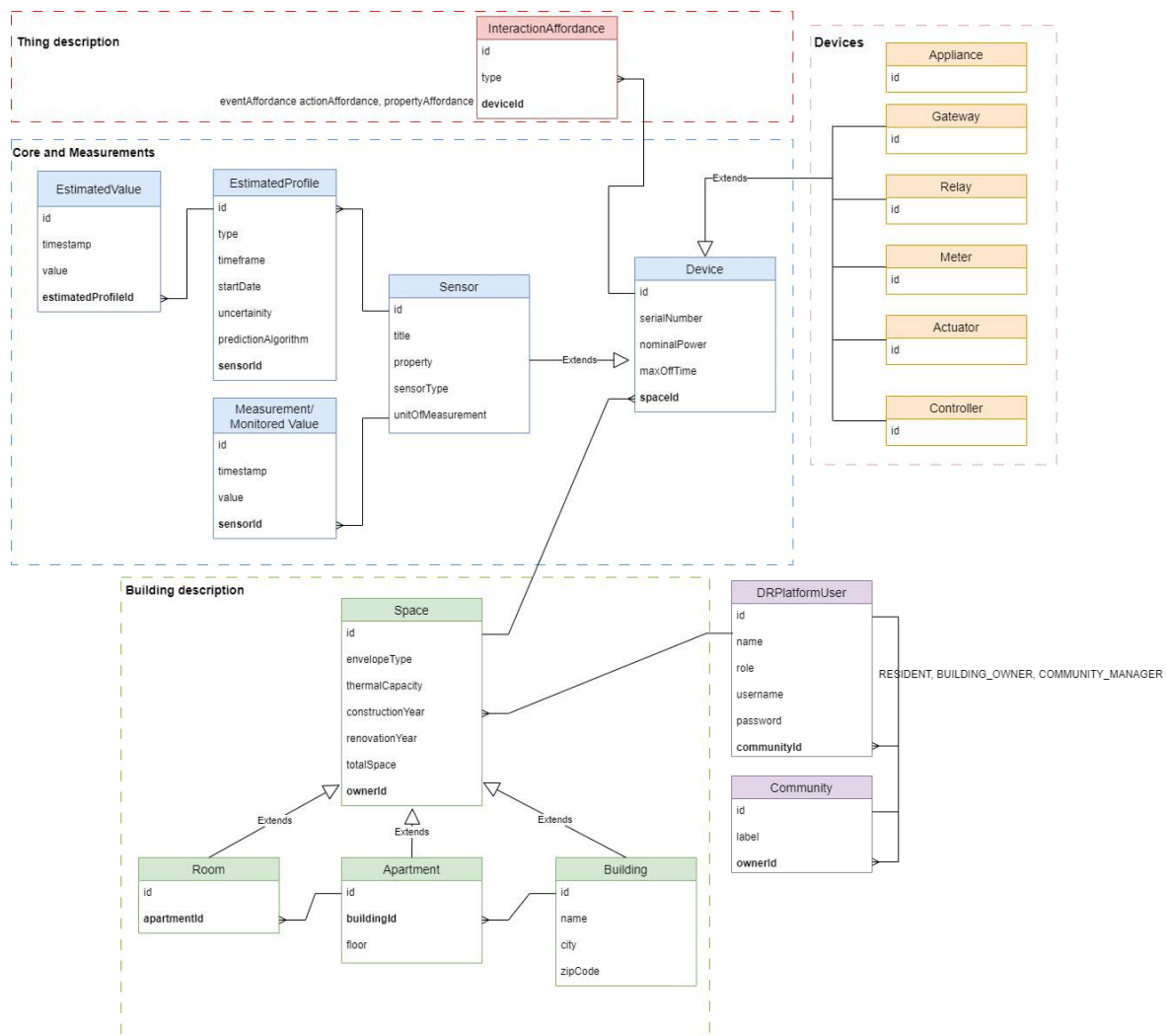


Figure 39: BIFTS Data Model

The relationships among data model entities are established based on their interactions and dependencies. In the case of buildings, participants may own the entire building or only an apartment/floor/room which are equipped with various devices with different capabilities. The devices can be installed at different levels such as building, apartment and room level. Monitoring sensors are devices that send monitored data and may also have predicted values obtained through forecasting algorithms.

The participants in the DR Platform are represented by the entity **DRPlatformUser** having the roles in the system of building owner or residents. They have associated username and password for authentication and identification purposes. The participants can be part of a **Community** or can be individual participants. This enables greater flexibility in how the participants with different roles may interact with the platform.

The data model is organized around four main components (same as the ones depicted in Figure 23) to represent different aspects in a structured manner. These components are described in the following sections.



7.2.1. Building description

The building component enables the representation of the building structure through entity relationships the associated data storage and management. For more flexible representation, we have defined a generic **Space** entity that can be used to represent the entire building, an apartment, or a room. This can be further extended to represent different kinds of spaces like an entire floor. The Space entity contains general information that all spaces may have like total area, thermal capacity, information about space construction or renovation. Each space is owned by a DRPlatformUser, and the entity contains the *ownerId*.

The more specific representation of the spaces is realized through the **Room**, **Apartment** and **Building** entities that extend the Space. Each Room has associated an apartment, and each apartment is in a building. The space is further connected to devices for core and measurements details. Thus, this representation enables fine grain configuration of the devices room, apartment or building level.

7.2.2. Core and measurements.

The core and measurements component has the **Device** entity which is a general representation of different types of devices associated with spaces. The device is identified by an id and serial number, has a nominal power and a maximum offline time.

Sensor entity is a specific representation used for measurement devices that extends the device entity. Each sensor may have multiple measurements that are represented by the **Monitored Value** entity, and also some estimated profiles generated by forecasting algorithms. The unit of measurements is stored in the sensor entity together with information about the property measured. Each sensor has a type that can be for energy consumption, production, or both. This representation supports integration of more types of sensors and different measurement units.

The estimated profile representing the predicted values are in the same measurement unit as the data collected from the sensors. Each **EstimatedProfile** is associated with a Sensor device and has a defined timeframe (day-ahead, intraday), a start date which represents the start of the estimated profile, the prediction algorithm used, and the uncertainty of the algorithm obtained from the evaluation of the prediction model. The estimated profile is composed of multiple **EstimatedValue** entities based on the specified timeframe. Each **EstimatedValue** is composed of timestamp, value and the *estimatedProfileId*.

7.2.3. Devices

Devices component consists of different specific types of devices with additional details for each type. Besides the sensors that belong to the core and measurement component, this data model supports devices like **Appliance**, **Gateway**, **Relay**, **Meter**, **Actuator** and **Controller**. These entities also extend the device entity.



7.2.4. Thing description

Thing description is a component defined for IoT representation of the devices and has the objective of describing their capabilities based on WoT Thing Description (TD). This component includes an **InteractionAffordance** entity that enables integration of different interaction types on IoT devices:

- Property (*propertyAffordance*): internal state or configuration of the device that can be exposed and, in some cases, updated (e.g. Set point temperature property of a thermostat)
- Action (*actionAffordance*): actions that can be performed by the device on command that make changes on the state (eg. Turn on/off a device)
- Event (*eventAffordance*): trigger events for observable actions or state changes within the IoT device (eg. change in temperature, motion detection, a button press)

7.2.5. Data model API

We have developed an API for the data model (Resource API) as a Spring Boot Java Application created over the data model presented. It is connected to a database (resources-db) which contains all configuration, monitored and estimated data for the buildings using the data model structure. The architecture of the API is presented in Figure 40. It is organized in three layers.

The controllers are exposing endpoints that can be used to gather, insert, or manage data. The services are responsible for the application logic and data manipulation. The Data Transfer Objects (DTOs) used for data transfers have associated builders and validators for easier mapping and to ensure that data follows a certain format. The repositories layer is responsible for managing data persistence and interaction with the underlying data storage system, resources-db. This layer encapsulates all the logic related to database operations and defines custom queries for retrieving more efficient monitored data.



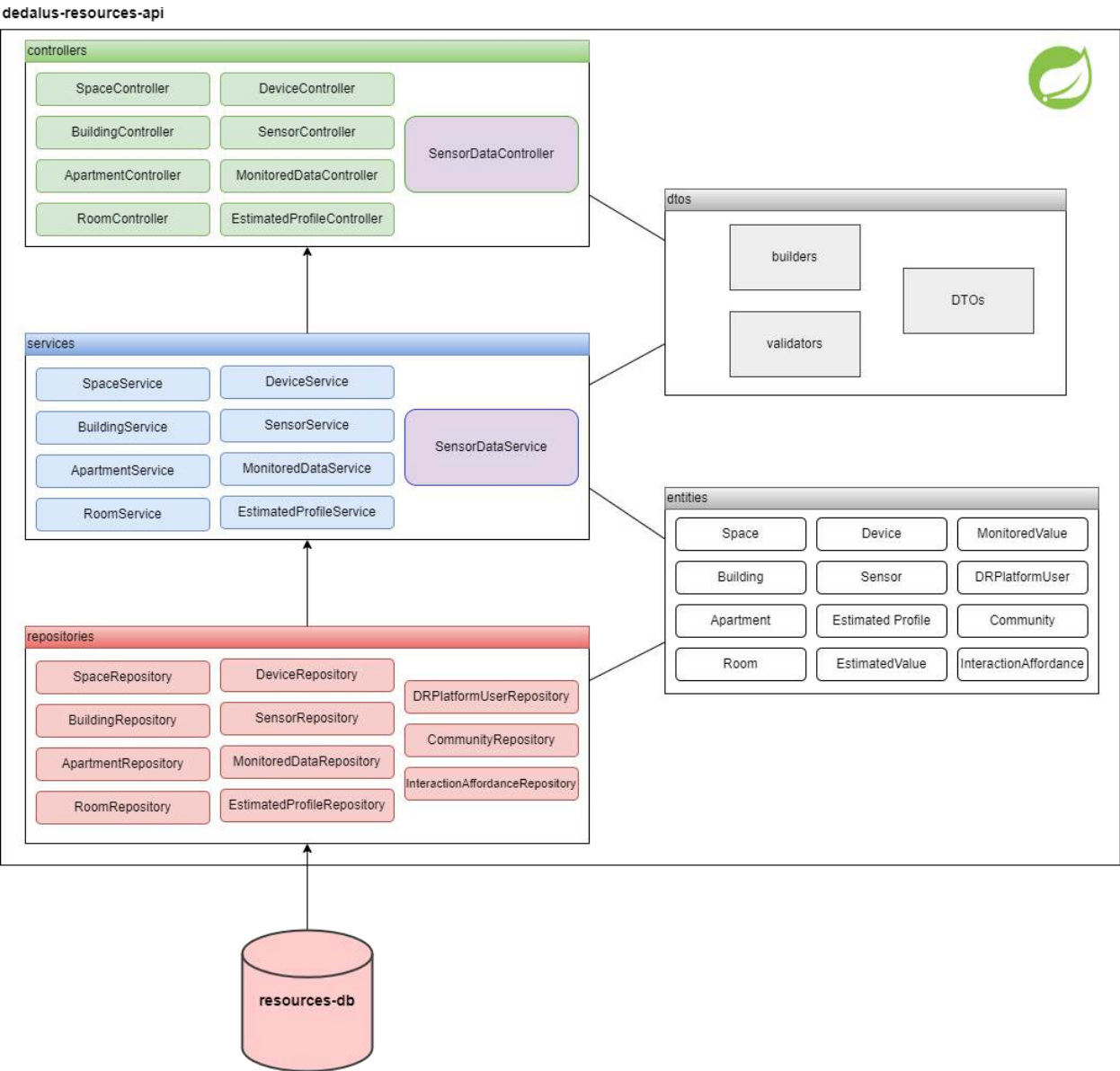


Figure 40: Data model API architecture.

The tables below present the main endpoints used for asset configuration, monitoring and estimating, data insertion and retrieving training dataset and prediction input.

Table 5: Insert DR Platform users API

	POST /dedalus-resource-api/dr-platform-user/community-manager
Description	Inserts a new DR platform user with COMMUNITY_MANAGER role
Parameters	-
Body	{ "id": uuid, "name": string, "username": string, "password": string,



	"ethAddress": string, "ethPassphrase": string }
Response	200 – user inserted successfully: uuid
POST /dedalus-resource-api/dr-platform-user/building-owner	
Description	Inserts a new DR platform user with BUILDING_OWNER role
Parameters	-
Body	{ "id": uuid, "name": string, "communityId": uuid "username": string, "password": string, "ethAddress": string, "ethPassphrase": string }
Response	200 – user inserted successfully
POST /dedalus-resource-api/dr-platform-user/resident	
Description	Inserts a new DR platform user with RESIDENT role
Parameters	-
Body	{ "id": uuid, "name": string, "communityId": uuid "username": string, "password": string, "ethAddress": string, "ethPassphrase": string }
Response	200 – user inserted successfully: uuid

Table 6: Insert community API

POST /dedalus-resource-api/community/{label}/{manager_username}	
Description	Inserts a new community with specified manager username
Parameters	label – community label manager_username – username of the community manager
Body	{}
Response	200 – community inserted successfully: uuid

Table 7: Configure spaces, apartments, and buildings API

POST /dedalus-resource-api/space	
Description	Inserts a space
Parameters	-
Body	{ "id": uuid, "label": string, "envelopeType": string, "thermalCapacity": string, "constructionYear": int, "renovationYear": int, "totalSpace": in,



	"ownerId": uuid }
Response	200 – space inserted successfully
POST /dedalus-resource-api/building	
Description	Inserts a new building and the corresponding space for specified owner
Parameters	-
Body	{ "id": uuid, "label": string, "ownerID": uuid "envelopeType": string, "thermalCapacity": string, "constructionYear": int, "renovationYear": int, "totalSpace": int, "name": string, "city": string, "zipCode": string }
Response	200 – building inserted successfully
POST /dedalus-resource-api/apartment	
Description	Insert a new apartment and the corresponding space for specified building
Parameters	-
Body	{ "id": uuid, "label": string, "ownerID": uuid, "envelopeType": string, "thermalCapacity": string, "constructionYear": int, "renovationYear": int, "totalSpace": int, "floor": int, "buildingID": uuid }
Response	200 – apartment inserted successfully

Table 8: Configure sensors API

POST /dedalus-resource-api/sensor/insert	
Description	Inserts a new sensor and corresponding device for the specified space
Parameters	-
Body	{ "sensorId": uuid "serialNumber": string, "nominalPower": int, "maxOffTime": int, "spaceUUID": uuid, "title": string, "property": <i>PROPERTY</i> , "sensorType": <i>SENSOR_TYPE</i> , "unitOfMeasurement": <i>MEASURE_UNIT</i> "maximumValue": 0 }



	Property: <i>ELECTRICAL_POWER</i> , <i>VOLTAGE</i> , <i>ACTIVE_ELECTRICAL_POWER</i> , <i>REACTIVE_ELECTRICAL_POWER</i> , <i>APPARENT_ELECTRICAL_POWER</i> , <i>TEMPERATURE</i> , <i>TOTAL_HORIZONTAL_IRRADIATION</i> , <i>TOTAL_DIFFUSE_IRRADIATION</i> , <i>BEAM_IRRADIATION</i> , <i>HUMIDITY</i> , <i>AIR_PRESSURE</i> , <i>WIND</i> , <i>CLOUD_COVER</i> MEASURE_UNIT: <i>KWh</i> , <i>Wh</i> , <i>KW</i> , <i>W</i> , <i>VOLT</i> , <i>PERCENTAGE</i> , <i>hPa</i> , <i>METRE_PER_SECOND</i> , <i>CELSIUS</i> SENSOR_TYPE: <i>CONSUMPTION_SENSOR</i> , <i>PRODUCTION_SENSOR</i>
Response	200 – sensor inserted successfully: uuid

Table 9: Insert Monitored Values and get input for training and prediction API

POST /dedalus-resource-api/monitored-value/insert	
Description	Insert a new monitored value for the specified sensor
Parameters	-
Body	<pre>{ "timestamp": long, "sensorId": uuid "monitoredValue": double, "userId": uuid "hourMinute": string }</pre>
Response	200 – monitored value inserted successfully
POST /dedalus-resource-api/monitored-values/insert/energy-profile	
Description	Inserts multiple monitored energy values for the sensor identified by sensor title
Parameters	-
Body	<pre>{ "energyValues": [{ "timestamp": YYYY-MM-DDTHH:MM:SS.SSSZ "value": double }, ..], "sensorTitle": string }</pre>
Response	200 – monitored value inserted successfully
POST /dedalus-resource-api/forecasting/input-for-sensor-selection/{spaceId}/{timestamp}/{noDays}	
Description	Returns the prediction input for the specified sensors (monitored values)
Parameters	spaceId – uuid of the space timestamp – long timestamp in milliseconds noDays – number of days used for baseline calculation
Body	<pre>{ "sensorProperties": ["ELECTRICAL_POWER", ...], "weatherForecast": true }</pre>
Response	InputDTO: { "day": int, "month": int, "year": int, "dayOfWeek": int, "holiday": int, "baseline": [...],



	<pre> "monitoringsPerDayPerSensor": { "sensor1": [0], "sensor2": [0],... }, "sensorDTOMap": { "sensor1": { "uuid": uuid, "title": string, "property": PROPERTY, "sensorType": SENSOR_TYPE, "unitOfMeasurement": MEASURE_UNIT, "maximumValue": int, "spaceUUID": uuid }, }, "dateTime": YYYY-MM-DDTHH:MM:SS.SSSZ } </pre>
	POST /dedalus-resource-api/forecasting/dataset-training/{spaceId}/{start}/{end}
Description	Returns dataset for training for the specified sensors and time interval (monitored values)
Parameters	spaceId – uuid of the space start, end – long timestamp in milliseconds
Body	<pre> { "sensorProperties": ["ELECTRICAL_POWER", ...], "weatherForecast": true } </pre>
Response	List of InputDTOs

Table 10: Insert Estimated Profile API

	POST /dedalus-resource-api/estimated-profile/insert
Description	Inserts an estimated profile for the specified sensor
Parameters	-
Body	<pre> { "sensorID": uuid, "timeframe": TIMEFRAME "uncertainty": double, "insertTime": YYYY-MM-DDTHH:MM:SS.SSSZ, "profileType": PROFILE_TYPE "predictionAlgorithm": ALGORITHM, "profileValues": [0], "label": 0 } </pre> TIMEFRAME: DAYAHEAD, INTRADAY PROFILE_TYPE: ENERGY_CONSUMPTION, ENERGY_PRODUCTION, FLEXIBILITY_BELOW, FLEXIBILITY_ABOVE, BASELINE, FLEXIBILITY_REQUEST



	ALGORITHM: <i>MLP, LSTM, NONE, SVR, GENETIC</i>
Response	200 – estimate profile inserted successfully

7.3. Data Ingestion Component

The data ingestion component integrates different types of information ingestion mechanisms to accommodate various data sources. We have defined two data collectors. Firstly, an FTP Data Collector that can retrieve sensor data from an external FTP server (as in the case of DEDALUS Pilot 7) and feed the information in the data model. Secondly, it is a MQTT based data integration with a producer that sends data to a queue at a fixed rate (as is the case of other DEDALUS pilots). The MQTT consumers collect data from the queue and make the needed energy conversions.

The FTP Data Collector is responsible for gathering data from the FTP server. Sensors from different buildings send energy consumption data to the server, which is then stored as CSV files. A new CSV file is created for each monitoring session with the file name containing date information. Each file line represents monitoring data from a sensor identified by the id on the first column. The second column represents the timestamp of the monitoring, and the third column contains the energy consumption measurement in kWh.

Figure 41 shows the architecture of the FTP data collector as well as the interactions with the FTP server and the component that is responsible for storing energy data. The collector has a timer task (Collector Timer Task) which can be configured to collect data at the specified time interval. Since we use hourly data monitoring, the timer is set to one-hour intervals. Once the timer task is scheduled, it will create a thread at each execution, that is responsible to establish a connection with the FTP server (1), copy the new files from the server (2), read monitored data from files and make energy conversions (aggregate data to obtain one monitored value for each sensor on the selected period, e.g. 1 hour) (3), and send the monitored data to the dedalus-resource-api that is responsible for data storage (4).

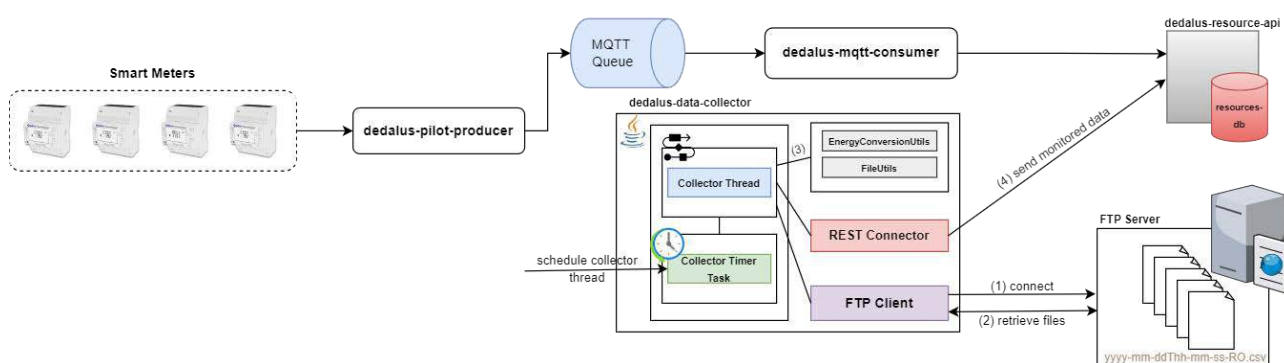


Figure 41: Data ingestion mechanisms.

Figure 41 shows the general flow for acquiring data from sensors devices such as smart meters. For testing purposes, energy data can be generated from CSV files by an estimator. The MQTT consumer subscribes to a queue and is responsible for data processing, energy conversions and



sending data to the database. The **dedalus-pilot-producer** receives energy consumption data from smart meters and sends it through an MQTT queue to which the **dedalus-mqtt-consumer** has subscribed. The **dedalus-mqtt-consumer** will further process the energy consumption data and stores it into the data model.

7.4. Energy Forecasting Component

The forecasting component is responsible for prediction model training, storing the trained models and making predictions on request or scheduled (Figure 42). It is generic so it can integrate different types of prediction models through an API. The training phase is triggered by an API call. Firstly, the **dedalus-forecasting-models** module requests training data through an API call from **dedalus-resource-api** which is responsible for data storage. Data is collected from the database and then the dataset for training is constructed and sent to the forecasting application.

After training the models, they are saved locally on files to enable the subsequent energy predictions. Predictions can be made on request by specifying the timestamp, or a thread can be scheduled to make predictions at the specified time interval.

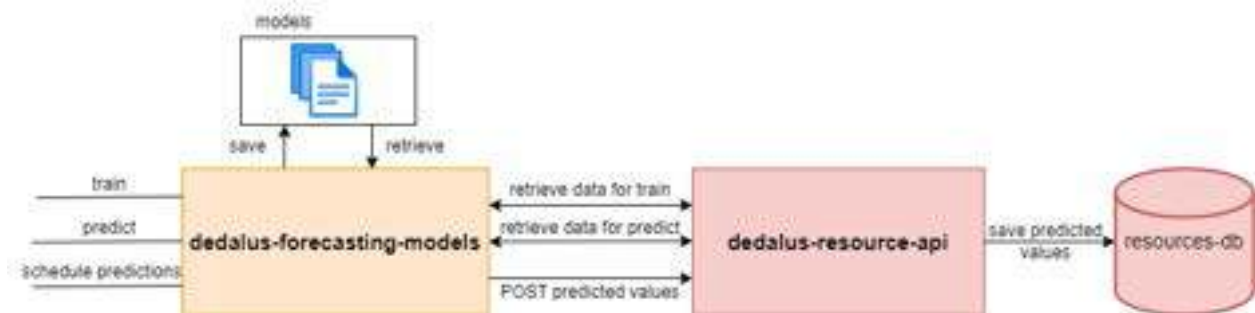


Figure 42: Forecasting tool – General flow.

The architecture of **dedalus-forecasting-models** is presented in Figure 43. It is a Python Flask application with resources defined for model training, scheduling and making predictions. Both **TrainResource** and **PredictionResource** expose endpoints for model training and for energy predictions, respectively. It creates a scheduled thread at the specified running period, that gathers input data from the **dedalus-resource-api** and makes energy predictions. Also, if a specific flag is set, the prediction will be sent to the **dedalus-resource-api** to be stored.

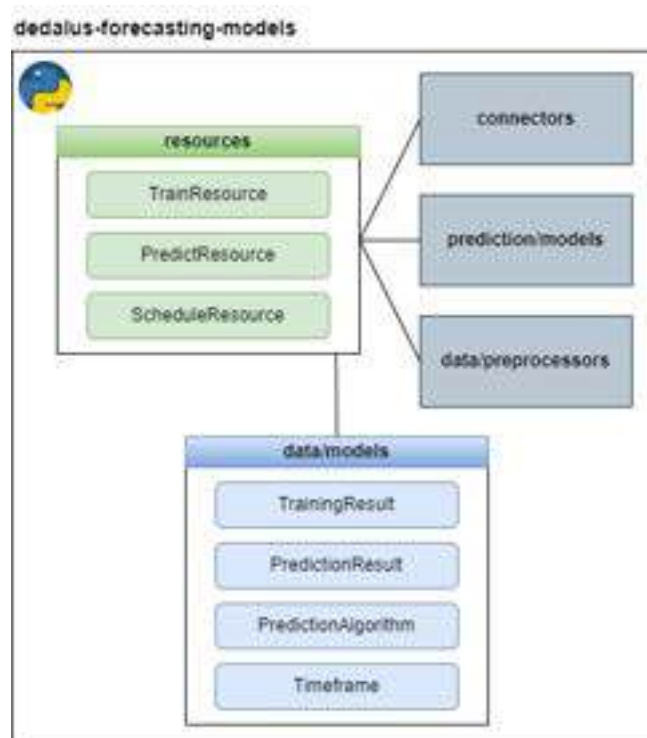


Figure 43: Forecasting component – dedalus-forecasting-models integration architecture.

To ensure easy seamless integration with the other decentralized forecasting tools developed in the project we have developed an integration API (see Table 11). This allows for replacing the currently implemented prediction models with alternatives that will be developed in DEDALUS.

Table 11: Forecasting component API

	POST /dedalus-forecasting-models/train/{spaceLabel}/{start}/{end}
Description	Train prediction models for <i>space</i> using train data in specified time interval
Parameters	spaceLabel – String start – long (timestamp in milliseconds) end – long (timestamp in milliseconds)
Body	{ sensorProperties: [ELECTRICAL_POWER, VOLTAGE, ACTIVE_ENERGY, REACTIVE_ENERGY, APPARENT_ELECTRICAL_POWER, TEMPERATURE, TOTAL_HORIZONTAL_IRRADIATION, TOTAL_DIFFUSE_IRRADIATION, BEAM_IRRADIATION, HUMIDITY, AIR_PRESSURE, WIND, CLOUD_COVER], weatherForecast: <i>boolean</i> }
Response	TrainingResult: { accuracy: <i>number</i> , min_MAE: <i>number</i> , min_MASE: <i>number</i> }
	POST /dedalus-forecasting-models/predict-day-ahead/{spaceLabel}/{timestamp}/{save}
Description	Predict day-ahead energy for <i>space</i>
Parameters	spaceLabel – String



	timestamp – long (timestamp in milliseconds) save – boolean (save prediction in database)
Body	{ sensorProperties: [ELECTRICAL_POWER, VOLTAGE, ACTIVE_ENERGY, REACTIVE_ENERGY, APPARENT_ELECTRICAL_POWER, TEMPERATURE, TOTAL_HORIZONTAL_IRRADIATION, TOTAL_DIFFUSE_IRRADIATION, BEAM_IRRADIATION, HUMIDITY, AIR_PRESSURE, WIND, CLOUD_COVER], weatherForecast: <i>boolean</i> }
Response	PredictionResult: { inserted: <i>boolean</i> , predictedValues: Array(24) }
POST /dedalus-forecasting-models/schedule-prediction/day-ahead /{spaceLabel}/{period}	
Description	Train prediction models for <i>space</i> using train data in specified time interval
Parameters	spaceLabel – String period – int (running periodicity for thread in minutes)
Body	{ sensorProperties: [ELECTRICAL_POWER, VOLTAGE, ACTIVE_ENERGY, REACTIVE_ENERGY, APPARENT_ELECTRICAL_POWER, TEMPERATURE, TOTAL_HORIZONTAL_IRRADIATION, TOTAL_DIFFUSE_IRRADIATION, BEAM_IRRADIATION, HUMIDITY, AIR_PRESSURE, WIND, CLOUD_COVER], weatherForecast: <i>boolean</i> }
Response	ScheduleResult: { started: <i>boolean</i> , }

7.5. Building Flexibility Trading

This section describes the BIFTS energy flexibility trading component, that uses blockchain network and smart contracts. The developments were done on top of the P2P energy trading solution proposed on Bright project for energy communities, using the Hardhat framework, which is an environment for Ethereum blockchain. Hardhat has many useful features, from developing and deploying smart contracts and decentralized applications, to optimizing the development chain by providing tools to run a local Ethereum network that is customizable and greatly enhances the development process. Additional business logic was implemented in Node.js, a modern, open-source, JavaScript runtime environment together with the hardhat.js framework that is used for interfacing with the blockchain network.

Figure 44 presents a sequence diagram that shows how each step is performed and how various types of DEDALUS related smart contracts are interacting with each other.



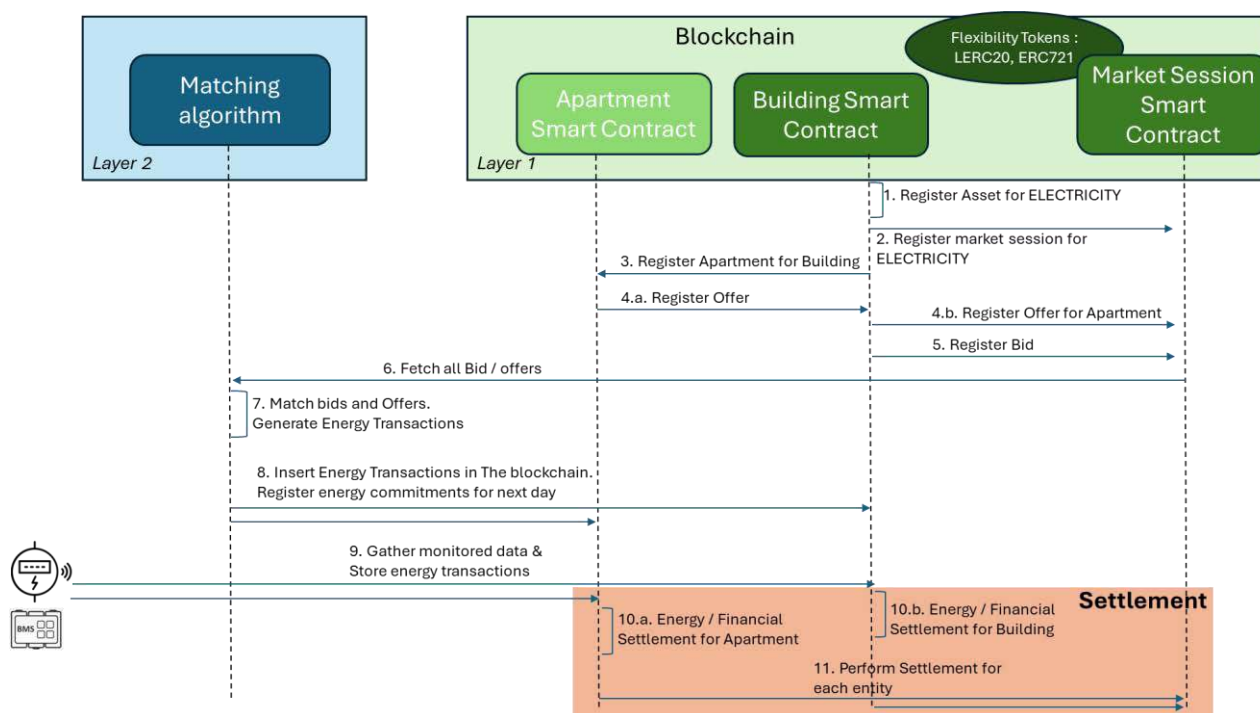


Figure 44: Interaction flow for building flexibility aggregation

7.5.1. The blockchain network

To assure a secure medium for building managers and apartment residents, we will use a private blockchain network in the Hardhat network, which is a local Ethereum network node. Hardhat allows to perform a multitude of activities, from deploying smart contracts, to running tests and debugging and has many customizable properties that enable its users to tune their network based on their needs. As default setting, a block is mined with each new transaction, in their receiving order, with no delays. That way, we are enhancing the development process without experiencing the common issues of an Ethereum network regarding the speed of block generation which is fixed for the entire network to around 10 seconds.

Specifically, for setting up the private Ethereum network for BIFTS, using Hardhat, we have used all the basic settings of this network that mimic how a public Ethereum would work, such that by migrating the setup to a public testnet or even to the mainnet, this would be seamless and quickly executed. We wrote, compiled and deployed in the network smart contracts (described in the following sub-sections) corresponding to the building and apartments. Hardhat automatically generates a set of folders, each smart contract getting its own folder, with the JSON file representing its compilation. These folders and implicitly, the corresponding JSON file of each smart contract, are accessible by their contract name.

We aim to develop DR programs that transform buildings into flexible assets by using their non-grid owned assets' flexibility or the flexibility of the residents' assets. The blockchain based ledger aims to store in a tamper proof manner the hourly energy collected by smart meters, based on the raw data stored in the data model. For this purpose, smart contracts were defined.



The building managers will act as flexibility buyers, and residential apartment owners, will act as flexibility sellers. Each such actor will own a specific smart contract deployed in their wallets. For energy trading between a building and multiple apartments, each building contract will deploy a Market Session smart contract, having the role of depositing bids and offers for the day ahead energy market session. Finally, each smart contract of a system actor (building manager or apartments residents) will save smart meter energy monitored data and programmatically compute rewards or penalties based on their committed values from the previous day. We have tokenized the process to allow intelligent mapping of remuneration and energy assets by using custom ERC20 and ERC721 token implementations.

For implementing the smart contracts we are using the Solidity programming language, an object-oriented, high level programming language, designed to be compiled and run on the Ethereum Virtual Machine – EVM. More specifically, we are using the latest stable version, 0.8.23, of the Solidity compiler, with the target evmVersion – Paris. When compiling our smart contracts, we have enabled the optimizer setting with 200 runs, which greatly reduces the deployment gas cost for smart contracts, due to reducing the stack size it allocates to each method for repeated method calls.

7.5.2. Flexibility Tokenization

Energy flexibility is represented in the blockchain-based distributed ledger by a customized ERC721 token whose role is to digitize the energy flexibility delivered in DR programs. ERC721 is the Ethereum-based standard for creating non-fungible tokens. In general, fungibility describes the property of assets to be readily interchanged for another asset alike meaning that any two tokens are identical. As the apartments or building assets are not interchangeable, thus unique, they are non-fungible. We use non-fungible tokens to quantify and substitute energy flexibility that was generated or consumed by the energy flexibility assets or apartments of the buildings. In this way we may quantify and associate the flexibility delivered with the apartment or asset thus facilitating the settlement process. We have used the latest version of ERC721 provided by Open Zeppelin library, which is the most widely used token standard in the Ethereum ecosystem. The main properties of the ERC721 energy flexibility token we used for BIFTS can be seen below.

```
1 library GridOperableLibrary {
2     struct EnergyTokenMetadata {
3         uint startTimeToken;
4         uint endTimeToken;
5         uint meterID;
6     }
7     ...
8 }
9
10 contract ERC721 {
11     string private _name;
12     string private _symbol;
13
14     mapping(uint256 => address) private _tokenOwner;
15     mapping(uint256 => address) private _tokenApprovals;
```



```

15
16 mapping(address => mapping(address => bool)) private _operatorApprovals;
17 mapping(uint256 => GridOperableLibrary.EnergyTokenMetadata) private _tokensDetails;
18 mapping(address => mapping(uint256 => uint256)) private _ownedTokensBalances;
19 mapping(address => mapping(uint256 => uint256)) private _ownedTokensBalancesArray;
20
21 mapping(address => uint[]) private ownerTokens;
22 mapping(uint => uint) private ownerTokensIndexPosition;
23
24 uint private tokenIndex;
25 ...
26 function mintWithTokenMetadata(address to, GridOperableLibrary.EnergyTokenMetadata
27 memory tokenMetadata, uint quantity) public returns (uint) {
28     tokenIndex++;
29     _mint(to, tokenIndex, quantity);
30     _setTokenMetadata(tokenIndex, tokenMetadata);
31
32     uint length = ownerTokens[to].length + 1;
33     ownerTokens[to].push(tokenIndex);
34     ownerTokensIndexPosition[tokenIndex] = length - 1;
35     return tokenIndex;
36 }
37 ...
38 function _transferFrom(address from, address to, uint256 tokenId, uint quantity) internal {
39     require(ownerOf(tokenId) == from);
40     require(to != address(0));
41     require(_ownedTokensBalances[from][tokenId] >= quantity);
42
43     _ownedTokensBalances[from][tokenId] -= quantity;
44     tokenIndex++;
45     _ownedTokensBalances[to][tokenIndex] = quantity;
46     _tokenOwner[tokenIndex] = to;
47
48     emit Transfer(from, tokenId, to, tokenIndex, quantity);
49 }
50
51
52

```

Figure 45: ERC721 energy flexibility smart contract

The first major adaptation from the classic ERC721 standard is the information held by the token. We have designed a custom metadata structure, `EnergyTokenMetadata`, that will hold characteristic data for each token instance, removing the normal token URI from OpenZeppelin ERC721 implementation (lines 2-7). Our goal is to ensure that once registered, each token will be the equivalent of an energy flexibility amount (i.e. decrease of energy demand), storing specific information that will make it distinguishable and traceable against other similar tokens



(of other apartments or at different time). Thus, for each token we are saving on-chain the start and end hour, which means the time during which the flexibility will be available, as well as the id of the meter associated with the apartment or asset that will deliver the flexibility. This adaptation makes the token unique, not only because of its owner, but also due to the time and place of generation or consumption. We also store information related to the owned token balances of each building manager or apartment resident, as well as a special data structure that accounts for the owned tokens of different types. We are using the main methods that of the ERC721 standard, such as *mint*, *transfer*, *transferFrom*, *setApproval* or *burn*. To identify the transfer of energy flexibility tokens to multiple participants, we are incrementing the global counter representing the token index, for each token generation or token transfer. Moreover, as each owner may hold more than one non-fungible tokens, we are using a map-based structure that associates several tokens to each address. This functionality means that the energy token marked by the same metadata, once transferred, could have its quantity split among more than one account.

7.5.3. Payments Tokenization

For paying for the tokens in smart contracts we use an adaptation of the standard ERC20 Ethereum token, which is the classic implementation of a fungible asset in the Ethereum network. This token can be minted when necessary, and thus, increasing its total supply available. We have custom ERC tokens to support trades of heterogenous assets, and thus, we will have a deployed ERC token for each possible type of energy asset / apartment that may participate. Moreover, to block the tokens in transactions until the actual flexibility delivery and settlement, a lockable feature was added on top of the ERC20 standard. As ERC20 standard, we are using the latest version of this smart contract provided by Open Zeppelin to manage commitments of energy flexibility delivery and allowing the validation of registered transactions using monitored data. As a result, the Lockable ERC20 contract provides the owners with an option to lock some of their own tokens, specifying a time-lock – a period after which the tokens can be unlocked, and an escrow party account – a third party, which can be a smart contract too, responsible to unlock the tokens when certain conditions are met. It has a safety mechanism that ensures that participants of the flexibility trading or other similar programs implemented at a building level, really own the tokens they offer. Although tokens are committed during program implementation, their spending is restricted. We need a decentralized management of building energy flexibility, thus a time-lock governed escrow between the owner (spending account-building manager) and the recipient (receiving account-apartments residents) needs to be implemented. This time lock should expire after energy transaction settlement, and the tokens are transferred either to the recipient or back to the owner. The main methods which we will use and extend to implement the Lockable ERC20 feature for BIFTS are shown below.

```

1 totalSupply () public view returns (uint256)
2 balanceOf (address _owner) public view returns (uint256 balance)
3 transfer (address _to, uint256 _value) public returns (bool success)
4 transferFrom (address _from, address _to, uint256 _value) public returns (bool success)
5 approve (address _spender, uint256 _value) public returns (bool success)
6 increaseAllowance (address spender, uint256 addedValue)
7 decreaseAllowance (address spender, uint256 subtractedValue) public virtual returns (bool)
8 allowance (address _owner, address _spender) public view returns (uint256 remaining)

```



Figure 46: Lockable ERC20 methods

In our solution, a lock is added over a given quantity of tokens, usually equivalent with a promised quantity of energy flexibility, by specifying: the locker (the transaction signing address), the number of tokens, the locking period, and the unlocking responsible. The *Locking Period* is the time interval during which the tokens are locked. When the flexibility trades are established, or flexibility is delivered, the tokens are released from their lock, thus the escrow party can check the order status and proceed with the unlocking phase. After the matching phase, the market contract holds all the necessary information to unlock the tokens and transfer them. In case an order has not been matched, the tokens return to the initial owner, otherwise, the escrow will initiate a transfer to the recipient (the selling party that has provided the asset).

To provide these additional functionalities, LockableERC20 extends the ERC20 contract implementation. The evidence of a locked balance for each account is kept by using two additional mappings. The *_locked* mapping links the address of the one responsible for that lock with the locker address (the locker can be the owner or an address that received approval from the owner to lock) and stores a *LockedEntity* data structure. The *_lockedBalance* mapping stores the locked balance for each account.

```

1 State:
2   address tokens_owner
3   MAP (address unlocking_address, MAP(address locking_address, LockedEntity lockedEntity)) _locked
4   MAP (address account_address, uint256 balance) _lockedBalance
5 Constructor:
6   Input: msg.sender, amount
7   Output: -
8   Begin:
9     ERC20("LockableERC20", "LERC20")
10    tokens_owner ← msg.sender
11    mint (tokens_owner, amount)
12  end
13 Struct: _Locked Entity
14 Members:
15   address owner
16   uint256 amount
17   uint blockNo
18   bool isActive
19

```

Figure 47: LockableERC20 definition

7.5.4. Building Flexibility Decentralized Management

This section describes the main smart contracts that were used to manage and aggregate the building energy flexibility and how entities interact. We have designed three main smart contracts to model the decentralized energy flexibility aggregation based on trading: *Apartment*, *Building* and *Market Session*. Each actor who decides to join the energy market needs to be granted



permission to join, since we have designed a closed environment where all participants need to be verified prior to interacting with the platform. A building manager needs to verify all apartments before joining the network, and thus, after the offline verification, the building manager or apartment resident will have a specific smart contract deployed for them. The smart contracts are deployed from the wallets of the building manager or apartments who will be the contract's owners and as a result, will gain access to all contract functionalities. A building manager can create a market session and the associated energy asset type directly from their smart contract. The market session is the main tool for modeling possible interactions for the energy flexibility trading and aggregation process. They are day-ahead market sessions, which means that they host transactions between buying and selling flexibility orders one day prior to the actual delivery day. Each apartment resident owner or building manager can register offers, or bids, for the market session created by the building manager. These orders are placed for the next day and are based on energy flexibility predictions.

7.5.4.1. Apartment/Asset associated Smart Contract

An apartment smart contract models all the actions regarding apartment or building energy assets participation and offerings of their flexibility in decentralized energy flexibility trading and aggregation model. It needs to be able to keep track of multiple data related to each apartment or asset they are associated with. The most relevant data is the following (see Figure 48):

- Building address: address of the building where the apartment/asset is located and the id of the associated smart meter.
- Baseline: baseline values for the apartment for each hour of the day. It is computed as the average of the previous 7 days.
- LERC20 address: address of the LERC20 smart contract, saved when deploying the apartment contract.
- Commitments: array of values representing the committed values for each hour. These committed values are recorded when registering energy transactions and represent the values that each apartment must respect when delivering monitored data.
- Buying / Selling address commitments: addresses that were involved in energy transactions for each hour of the next day.

```

1 function registerOffer(GridOperableLibrary.Order memory order) external {
2     ERC721 erc = ERC721(order.certificateAddress);
3     order.tokenId = erc.mintWithTokenMetadata(apartmentOwner, order.metadata, order.quantity);
4
5     emit MintedTokensBalance(order.metadata.startTimeToken, order.quantity, apartmentOwner,
6     DateTime.getDayTimestamp(order.timestamp));
7
8     uint256 unlockAmount = order.quantity * order.price;
9     order.unlockAmounts[order.metadata.startTimeToken] = unlockAmount;
10
11     if (LockableERC20(LERC20Address).lockFrom(apartmentOwner, unlockAmount, blockNrForADay,
12     order.orderHandler)) {

```



```

13     Building(buildingAddress).registerOffer(order);
14     emit LockedBalanceOrder(order.metadata.startTimeToken, unlockAmount, order.orderHandler,
15     DateTime.getDayTimestamp(order.timestamp));
16 }
17     emit OrderRegistered(order, DateTime.getDayTimestamp(order.timestamp));
18 }
19
20 function updateCommitments(GridOperableLibrary.Trade memory trade, uint256 price, uint256
21 timeIndex) public {
22     commitments[timeIndex] += trade.quantity;
23     priceCommitments[timeIndex] = price;
24     sellingAddressCommitments[timeIndex] = trade.sellingAddress;
25     buyingAddressCommitments[timeIndex] = trade.buyingAddress;
26     handlerCommitments[timeIndex] = trade.buildingAddress;
27     ercCommitments[timeIndex] = trade.certificateAddress;
28
29     uint256 tradeUnlockAmount = trade.quantity * price;
30     uint256 orderUnlockAmount = orderUnlockAmounts[timeIndex];
31     uint256 transferAmount;
32
33     uint256 hourDeposit = realTimePrice;
34     transferAmount = orderUnlockAmount - (hourDeposit * trade.quantity);
35     if (transferAmount != 0) {
36         LockableERC20(LERC20Address).unlockWithoutTransfer(apartmentOwner, transferAmount,
37         address(this));
38     }
39     emit CommitmentResult(timeIndex, transferAmount,
40     DateTime.getDayTimestamp(trade.timestamp));
41
42     emit ClearingRecorded(
43         apartmentOwner,
44         orderUnlockAmount,
45         tradeUnlockAmount,
46         trade.apartmentAddress,
47         trade.buildingAddress,
48         DateTime.getDayTimestamp(trade.timestamp));
49 }
50
51 function monitor(uint timestamp, uint value) external {
52     require(msg.sender == apartmentOwner, "Apartment owner is not the method caller");
53
54     DateTime._DateTime memory dt = DateTime.parseTimestamp(timestamp);
55     uint timeIndex = dt.hour;
56
57     uint256 promisedValue;
58     uint256 settlementPayment;
59     uint256 unlockAmount;
60
61     uint256 flexibilityValue = 0;

```



```

62  if (value > baseline[timeIndex]) {
63      flexibilityValue = value - baseline[timeIndex];
64  }
65
66  promisedValue = commitments[timeIndex];
67  if (flexibilityValue >= promisedValue) {
68      if (promisedValue * realTimePrice > 0) {
69          if (unlockAmount > 0)
70          {
71              LockableERC20(LERC20Address).unlockWithoutTransfer(
72                  apartmentOwner,
73                  unlockAmount,
74                  address(this)
75              );
76          }
77      }
78  } else {
79      settlementPayment =
80  availableMonitorings[GridOperableLibrary.MonitoringType.PERCENT].monitor(realTimePrice,
81  flexibilityValue, promisedValue);
82      if (settlementPayment > 0) {
83          //at the building or be transferred further to the market session
84          LockableERC20(LERC20Address).unlockTransfer(
85              apartmentOwner,
86              settlementPayment,
87              buildingAddress,
88              address(this)
89          );
90      }
91      if (settlementPayment < promisedValue * realTimePrice) {
92          unlockAmount = (promisedValue * realTimePrice) - settlementPayment;
93      } else {
94          unlockAmount = 0;
95      }
96      if (unlockAmount > 0) {
97          LockableERC20(LERC20Address).unlockWithoutTransfer(
98              apartmentOwner,
99              unlockAmount,
100             address(this)
101         );
102     }
103 }
104 emit SettlementResult(
105     timeIndex,
106     value,
107     promisedValue,
108     unlockAmount,
109     settlementPayment,
110     DateTime.getDayTimestamp(timestamp)

```



111	//	<i>objectHash</i>);
112	);	
113	}	
114		

Figure 48: Apartment smart contract model

The smart contract has a specialized method of registering offers for the day ahead of the market session. The first step is to mint our ERC721 token in a quantity equal to the quantity of the order – lines 2-3. This will ensure that the apartment owns tokens which represent the order quantity placed in the market that can be traded before the actual energy delivery from the next day. Each important action is followed by an event emitted on the blockchain, to help us keep track of these actions over time. The unlocked amount of the order is computed as the product between the order quantity and order price – lines 8-9 followed by a lock from the LERC20 smart contract of that amount, on behalf of the apartment owner and can be unlocked by the apartment smart contract itself – lines 11-19. If the lock succeeds, proper events are emitted, and the order is sent to the building that will forward it to the market session smart contract. The update commitments method is called when an energy transaction is recorded between a buy order of a building and a sell order or an apartment – lines 20-49. It has the role of recording on-chain the values agreed upon in the energy transaction, which need to be respected the next day during the energy delivery phase. After saving the energy transaction data in specific data arrays – lines 22-27, the next step is to check if the initial order placed by the apartment is equal or higher to the energy transaction values. If it is higher, the remaining energy quantity should be unlocked from the LERC20 smart contract and the tokens returned to the apartment while the transacted amount remains locked – lines 29-38. Again, the appropriate events are emitted with this information such that we can track them later.

The last important method of the apartment smart contract is the monitoring method. During energy delivery, data monitored from smart meters will be injected directly on chain which triggers the monitor method which is described below. The method can be called only by the apartment owner wallet and takes the monitored value and its timestamp as arguments. After extracting the monitoring hour from the timestamp, we proceed by computing the flexibility value as the monitored value minus the baseline for that hour of the day – lines 61-64. The next step is to compare the flexibility value to the commitments saved during the energy transaction registration phase. If the flexibility value is higher or equal to the promised value and the unlock amount is higher than zero, then we will proceed to unlock all the remaining locked amount. If not, we compute the settlement payment penalty using a factory method which allows us to select between multiple methods of computing this penalty. In this version of the market, we are using the percent penalty method which computes the penalty only if the flexibility quantity is lower than 90% of the committed value penalty being computed as the difference multiplied by the price required to buy the excess energy from the real time market – lines 79 – 103.

7.5.4.2. *Building associated Smart Contract*

A building smart contract models all the actions a building manager can take in the decentralized flexibility market to aggregate the flexibility of the residents and participate in DR programs. The



building smart contract is responsible for creating day ahead market sessions between itself and the apartments it owns and placing buy flexibility orders on the market while apartments / assets that belong to the building may place selling orders. A building associated smart contract needs to save the following important variables for creating and managing decentralized energy trading (see Figure 49).

- Owned apartments' addresses: list of apartments or energy assets part of the building.
- Baseline: baseline values for the apartment for each hour of the day. Computed as the average of the previous 7 days.
- LERC20 address: address of the lerc20 smart contract, saved when deploying the apartment/asset associated contract.
- Commitments: array of values representing the committed energy values for each hour. These committed values are recorded when registering energy flexibility transactions and represent the values that each apartment or asset must deliver being tracked based on the energy monitored data.
- Buying / Selling address commitments: addresses that were involved in energy flexibility transactions for each hour of the next day.
- Market sessions: mapping of address to address representing market sessions that are linked to certain energy certificates, representing different types of energy: electricity, thermal etc.
- Energy certificates: mapping from string to address, representing the links between the name of each energy type, to the associated address of the ERC721 token.

Regarding market participation functionality, a building associated contract has similar features with the apartment/asset associated contract. The contract is comprised of the three main components: order registration, commitment saving and energy monitoring. In the order registration, the building will place a buy flexibility order, without minting ERC721, since it intends to buy and aggregate energy flexibility. As a result the contract will only lock up the buy order amount corresponding to the order's values. It also has a method to forward the offer placed by the apartment, since the building stores the market sessions for all certificates and knows the address of the Market Session contract that represents the energy certificate, they want to place orders for. Regarding the commitment phase, the building will unlock and transfer all the amount of tokens that represent the energy that was agreed upon in the energy transaction, to the actual apartment/ asset seller, and this would be the only difference compared to the commitment set up in the apartment contract, where it just has to unlock the tokens that represent the excess energy when subtracting the trade values from the initial order values. While the apartment has already delivered the energy equivalent to what it is committed to, the building must pay for the flexibility energy. As a result, this leads to the buyer owning the token ID corresponding to the energy which it will receive the next day (ERC721 token), and the apartment has been paid for what it will produce the next day (LERC20 tokens). In the monitoring phase, the building will transfer to the market session contract a penalty in the form of LERC20 tokens, only if the promised value is lower than the flexibility value by more than 10%.

Finally, the building associated smart contracts are responsible for creating and managing the market sessions that will take place between the buildings and their apartments.



```

1 function setEnergyCertificate(address _address) public {
2     ERC721 certificate = ERC721(_address);
3     string memory name = certificate.getName();
4     energyCertificates[name] = _address;
5     emit EnergyCertificateRegistered(name, _address, DateTime.getDayTimestamp(block.timestamp));
6 }
7
8 function registerMarketSession(uint _startHour, uint _endHour, string memory _energyType, uint
9 timestamp) public {
10     address energyCertificateAddress = energyCertificates[_energyType];
11     require(energyCertificateAddress != address(0), 'no energy certificate on this energyType');
12     MarketSession marketSession = new MarketSession(buildingOwner);
13     setupMarketSession(_startHour, _endHour, energyCertificateAddress, marketSession, timestamp);
14     emit EnergyTypeOfMarketSession(_energyType, DateTime.getDayTimestamp(timestamp));
15 }
16
17 function updateMarketSession(
18     uint _startHour,
19     uint _endHour,
20     string memory _energyType,
21     uint timestamp) public {
22     address energyCertificateAddress = energyCertificates[_energyType];
23     require(energyCertificateAddress != address(0), 'no energy certificate on this energyType');
24     require(marketSessions[energyCertificateAddress] != address(0), 'no market session with this
25 address');
26     MarketSession marketSession = MarketSession(marketSessions[energyCertificateAddress]);
27     setupMarketSession(_startHour, _endHour, energyCertificateAddress, marketSession, timestamp);
28 }
29 function setupMarketSession(
30     uint _startHour,
31     uint _endHour,
32     address energyCertificateAddress,
33     MarketSession marketSession,
34     uint timestamp) internal {
35     GridOperableLibrary.MarketSessionConfiguration memory sessionConfiguration =
36     GridOperableLibrary.MarketSessionConfiguration(address(marketSession), _startHour, _endHour,
37     timestamp);
38     marketSession.setSessionConfiguration(sessionConfiguration);
39     marketSessions[energyCertificateAddress] = address(marketSession);
40     emit
41     MarketSessionRegistered(marketSessions[energyCertificateAddress], marketSession.getSessionId(),
42     sessionConfiguration);
43 }

```

Figure 49: Building smart contract model



7.5.4.3. Trading Session associated Smart Contract

The market session smart contract main purpose is to store the flexibility bids and offers for each session, as well as transactions resulting from their matching – lines 1-6. Instead of mappings, we have chosen to store them in arrays due to better iteration possibilities in the associated methods. However, when we need to query the stored elements by array position, we need to use mappings queried by elements unique ids. . The methods for placing bids and offers, which are called by the building smart contract that owns each market session, along with the support data structures are shown in lines 1 – 20.

```

1  GridOperableLibrary.Order[] private bids;
2  GridOperableLibrary.Order[] private offers;
3  GridOperableLibrary.Trade[] private trades;
4
5  mapping(bytes32 => uint) private orderPosition;
6  mapping(bytes32 => uint) private tradePosition;
7
8  function placeBid(GridOperableLibrary.Order memory _order) external {
9      bids.push(_order);
10     orderPosition[_order.id] = bids.length - 1;
11
12     emit PublishedBid(DateTime.getDayTimestamp(_order.timestamp), _order);
13 }
14
15 function placeOffer(GridOperableLibrary.Order memory _order) external {
16     offers.push(_order);
17     orderPosition[_order.id] = offers.length - 1;
18
19     emit PublishedOffer(DateTime.getDayTimestamp(_order.timestamp), _order);
20 }
21
22 function registerIndividualTrade(GridOperableLibrary.Trade memory _trade) public {
23     uint positionBid = orderPosition[_trade.buyOrderId];
24     uint positionAsk = orderPosition[_trade.sellOrderId];
25
26     GridOperableLibrary.Order memory buyOrder = bids[positionBid];
27     GridOperableLibrary.Order memory sellOrder = offers[positionAsk];
28
29     require(sellOrder.metadata.startTimeToken == buyOrder.metadata.startTimeToken, "matched
30 orders do not have the same star hour");
31     require(sellOrder.metadata.endTimeToken == buyOrder.metadata.endTimeToken, "matched
32 orders do not have the same end hour");
33     require(sellOrder.certificateAddress == buyOrder.certificateAddress, "matched orders do not have
34 the same certificate address");
35
36     address buildingAddress = buyOrder.orderHandler;
37     Building buildingContract = Building(buildingAddress);

```



```

38     address apartmentAddress = sellOrder.orderHandler;
39     Apartment apartmentContract = Apartment(apartmentAddress);
40
41     ERC721(_trade.certificateAddress).transferFrom(_trade.sellingAddress, _trade.buyingAddress,
42 _trade.tokenId, _trade.quantity);
43     emit EnergyTokensTransfer(sellOrder.metadata.startTimeToken, _trade.quantity,
44 _trade.sellingAddress, _trade.buyingAddress, apartmentAddress, buildingAddress,
45 DateTime.getDayTimestamp(_trade.timestamp));
46
47     buildingContract.setQuantityCommitmentForTimeIndex(buyOrder.metadata.startTimeToken,
48 _trade.quantity);
49
50     buildingContract.updateCommitments(_trade, buyOrder.price,
51 buyOrder.metadata.startTimeToken);
52     apartmentContract.updateCommitments(_trade, hourDeposit,
53 sellOrder.metadata.startTimeToken);
54
55     buyOrder.quantity -= _trade.quantity;
56     sellOrder.quantity -= _trade.quantity;
57
58     bids[positionBid] = buyOrder;
59     offers[positionAsk] = sellOrder;
60
61     buyOrder.quantity += _trade.quantity;
62     sellOrder.quantity += _trade.quantity;
63 //
64     emit MarketTrade(sessionId, DateTime.getDayTimestamp(_trade.timestamp), _trade.price, _trade,
65 buyOrder, sellOrder);
66 }
67
68 function refundUnmatchedBids() private {
69     uint256 unlockedAmount = Building(owner).unlockAllTokens (prosumerOwner, owner);
70     emit UnlockedBalanceUnmatchedBidOrder(unlockedAmount, address(this),
71 sessionConfiguration.sessionTimestamp);
72 }
73
74 function burnTokensUnmatchedAsks() private {
75     for(uint i = 0; i < offers.length; i++) {
76         if(offers[i].quantity > 0) {
77             GridOperableLibrary.Order memory sellOrder = offers[i];
78             ERC721(sellOrder.certificateAddress).burn(sellOrder.tokenId);
79             emit BurnedTokensBalance(sellOrder.metadata.startTimeToken, offers[i].quantity,
80 sellOrder.orderHandler, DateTime.getDayTimestamp(sellOrder.timestamp));
81         }
82     }
83 }

```

Figure 50: Trading session smart contract model



The next step is to insert the energy flexibility transactions that were generated by the matching algorithm which is a layer 2 solution. For each blockchain-based flexibility trade, the participating apartments or assets flexibility orders are accessed by their ID, to be further processed. To register energy flexibility transactions, we need to perform the commitment phase and token transfer. After making sure that the bids and offers involved in energy flexibility transactions were placed for the same hour of the day – lines 29-35, the first step to successfully perform the commitment update is for the flexibility seller to transfer the ERC721 tokens representing the energy it will sell the next day, to the flexibility buyer – lines 41-42. As a result, instead of transferring energy, the blockchain system will transfer the equivalent token, considering the reading from the smart energy meters and the associated blockchain-based energy transactions. The token index generated from the transfer, which holds the trade quantity, which was transferred to the buyer, is saved, and stored in the building which buys that energy.

Next, the market session will call the update commitments methods from the Building and Apartment associated smart contracts that we have described in the previous sections, such that these contracts will store the state of the commitment phase – lines 50-53. Now that each contract stores how much energy and at what prices they have registered the commitments, the data inside the market session contract is updated such that we know which orders were fully matched and which were not – lines 55-62. This information will be used later for refunds. Finally, after all energy transactions were registered on-chain, we need to refund buyers who did not have their entire orders matched, with the LERC20 tokens they have locked when placing bids – lines 68-72, and the sellers will burn the ERC721 tokens they have minted for the sell orders that were not matched – lines 74-80.

7.5.5. Flexibility Aggregation and Settlement

This section describes the implementation of the flexibility aggregation algorithm, which generates energy flexibility transactions from flexibility bids and offers by prioritizing the offers with the highest energy flexibility quantity. Thus, the market will favor a higher total energy flexibility transfer rate over the market session. Then, the algorithm will match the submitted orders (sorted by energy quantity) until one of the two categories has no more left. A complete pseudocode can be seen in Figure 51.

Algorithm: Order matching

Input: $Apartment_{Flex}^{offer}[N]$, - flexibility offers ordered by the amount of flexibility and price:

$Building_{Flex}^{bid}$ – flexibility bid of the building manager.

Outputs: pair of matched energy flexibility bids and offers: $Flex_{Tr}[index_{offer}][bid]$ with the value 1

Begin

1. $Sort_{descending}(Apartment_{Flex}^{offer}[N])$
2. $index_{offer} = 0$
3. $trades = []$
4. **while** ($index_{offer} < N \ \&\& \ Apartment_{Flex}^{offer}[index_{offer}].quantity < Building_{Flex}^{bid}.quantity$) **do**
5. $Building_{Flex}^{bid}.quantity = Building_{Flex}^{bid}.quantity - Apartment_{Flex}^{offer}[index_{offer}].quantity$
6. $trade = P[index_{offer}][bid]$
7. $trade_{quantity} = Apartment_{Flex}^{offer}[index_{offer}].quantity$



```

8.   ApartmentsofferFlex[indexoffer].quantity = 0
9.   indexoffer ++
10.  tradeprice = ApartmentsofferFlex[indexoffer].price
11.  trades.add(trade)
12.  end while
13.  if indexoffer < N then
14.    ApartmentsofferFlex[indexoffer].quantity = ApartmentsofferFlex[indexoffer].quantity
15.                                           - BuildingbidFlex.quantity
16.    trade = P[bid][indexoffer]
17.    tradequantity = BuildingbidFlex.quantity
18.    BuildingbidFlex.quantity = 0;
19.    tradeprice = ApartmentsofferFlex[indexoffer].price
20.    trades.add(trade)
21.  end if
22.  return trades
End

```

Figure 51: Greedy algorithm pseudocode for decentralized matching of orders

After sorting the offers descending by their energy quantities – line 1, we analyze them in this order until the sum of all selected offers exceeds the bid quantity, it means we have filled up the bid with the appropriate offers – lines 4-20. Until there are offers available, and the remaining bid quantity is higher than the current offer, we decrease the bid quantity with the quantity of the offer that is currently matched with it – line 5. We create an energy transaction using the quantity and the price of the apartment offer and increase the offer index, such that we can go to the next offer – lines 7-10. If we have not reached the end of the apartment offer list, the final energy transaction will be generated to fill the remaining bid quantity, partially matching the apartment offer – line 14. Each energy bid and offer that are being matched will be traded at the same price between the buyer and the seller. The price of the energy transaction will be the price of the offer that was included in that transaction – lines 10 and 19.

The algorithm is integrated through an API allowing us to easily implement, integrate, use and test multiple flexibility aggregation and trading algorithms (see Table 12).

Table 12: Matching algorithm API endpoint

POST /match-orders/{algorithm}	
Description	Matched bids and offers based on an algorithm
Parameters	Algorithm – the matching algorithm we want to specify for matching
Body	{ activeAsks: [], activeBids: [], }
Response	200 – Energy transactions registered successfully



For settlement, we have dealt with how the payments are computed for both apartments and buildings. In the case of electricity-based flexibility (see Figure 52), we have opted for a settlement method shown below which computes the percentage represented by the difference between the promised energy flexibility value and the monitored energy value. The design is flexible to allow for integration of other settlement methods. If there is a difference higher than a threshold set for 10% (see line 4), then the apartment is not delivering the agreed flexibility amount stored in the blockchain transactions thus alternatives should be sought or penalties on the agreed payment can be applied (see lines 9-12).

```

1 contract ClassicSettlement is SettlementInterface{
2     function monitor(uint price, uint value, uint promisedValue) override(SettlementInterface) public pure
3         returns (uint){
4         uint deviationThreshold = 10;
5         int32 _value = int32(value);
6         int32 _promisedValue = int32(promisedValue);
7         uint diff = SafeMath.abs(_value - _promisedValue);
8         int32 promisedPrice = int32(price);
9         if (SafeMath.abs(100 - (_value * 100) / _promisedValue) > deviationThreshold) {
10             return diff * uint(promisedPrice);
11         } else {
12             return 0;
13         }
14     }
15 }
16 }

```

Figure 52: Settlement smart contract



8. Privacy, security and data protection framework for trusted data sharing and AI

This section serves a dual purpose: firstly, it introduces the platform intended to be used within DEDALUS in the scope of privacy, security and data protection issues (the Universal Secure Gateway (USG)). In DEDALUS, and due to the computational limitations of the existing USG hardware platform and the needed support for local, on-premises private data processing as specified by IDSA, a software solution will be developed by reusing and enhancing the existing USG building blocks, and this will enable a plug & play solution for deployment of pre-aggregation, orchestration, and activation of flexibility. Following this, the section delves into the development of the project's specific framework for privacy, security and data protection issues.

8.1. Universal Secure Gateway

Universal Secure Gateway is a technology enabler that was developed in the scope of previous H2020 projects: BRIGHT and PHOENIX. USG in its original form was a combination of hardware and software. Hardware platform is based on a TI AM335x Cortex-A8 processor used in a family of BeagleBone prototyping and development boards. It is packed in a DIN rail enclosure that can be mounted in household fuse boxes or in industrial environment. The operating system running on the USG is Debian GNU/Linux supporting standard GNU/Linux applications with computational capabilities limited by the low-power ARM processor on a system and limited amount of memory available.



Figure 53: Universal Secure Gateway device.



The software installed on the USG enabled integration of field assets and legacy systems with cloud backend and data analytics services deployed in cloud and on an edge. The USG software provided standardised APIs for technical interoperability with enabled security over legacy protocols.

In DEDALUS, and due to the computational limitations of the existing USG hardware platform and the needed support for local, on-premises private data processing as specified by IDSA, a software solution will be developed by reusing and enhancing the existing building blocks that will enable a plug & play solution for deployment of pre-aggregation, orchestration, and activation of flexibility. The ambition is to enable the energy community manager to gather real-time data from various sources (assets, grid, weather, market, etc.), and make informed decision based on analytics services developed in various tasks in DEDALUS project. At the time of writing the proposal, the plan was to base the USG software solution on the existing open-source projects such as LF Energy Fledge, FledgePower, Super Advanced Meter, FlexMeasures, OpenEMS, OpenHAB and HomeAssistant. The experience gained on the market and other projects motivated Consensus (COMS) to start developing new software platform that enables collecting data in a common format, exposing it to local optimisation and asset control services and exposing relevant data to third parties over GDPR-compliant APIs. That prevented the huge overlapping between the existing open-source software components to build up and overwhelm the development team with writing the connectors and interfaces between the specialised components but instead, focus on providing relevant functionality such as on demand response (DR) and local energy consumption management in a form of energy management system (EMS).

8.1.1. USG Software-Only Architecture

The USG software is deployed as a collection of Docker containers organised in a Docker Compose file providing a deployment recipe that can be replicated on any compatible personal computer, server machine or cloud-based computing instance preferably running GNU/Linux operating system with Docker Engine installed. Deployment formulation in a form of Docker Compose file enables easy reproducibility, deployment or use case-specific configuration modifications and deployment with integration of tools and services developed in DEDALUS project.

USG utilizes existing communication interfaces supported by the hosting platform especially the TCP/IP and UDP-based interfaces. By using the existing network interfaces, the USG connects to the user energy assets, DER resources, smart meters and other data sources using specialised IO microservices that support data retrieval and executing commands on devices that support the control action execution. MQTT broker enables unbuffered data collection and data streaming from outside data sources and internal IO microservices that are routed to the Kafka event streaming unit that supports data buffering in case data ingress pipeline temporarily stops working. Kafka also supports routing of data streams to specific local services by relying on topics and their partitions. Pre-processing service acts as an input data stream pre-processing unit that makes the necessary resampling of input data (in case meter data is stored every 15-min intervals). Data is stored in a local PostgreSQL database with TimescaleDB plugin extension to support the efficient time series data storage. Monitoring service provides the basic USG system performance information such as resource usage, service activity and system health. REST API



provides a unified data interface to all services deployed on an USG system where web app unit provides the visualisation of user household consumption and flexibility-related data and enables the community manager to assess the flexibility and expose it to relevant parties on energy and flexibility markets. The last part of the deployment is an identity and access management unit together with secure and privacy-preserving APIs to provide user management and user-based access management to relevant data by managing special resource-related consents and permissions that enable GDPR-compliant data exchange. The architecture of USG software platform is presented in Figure 54.

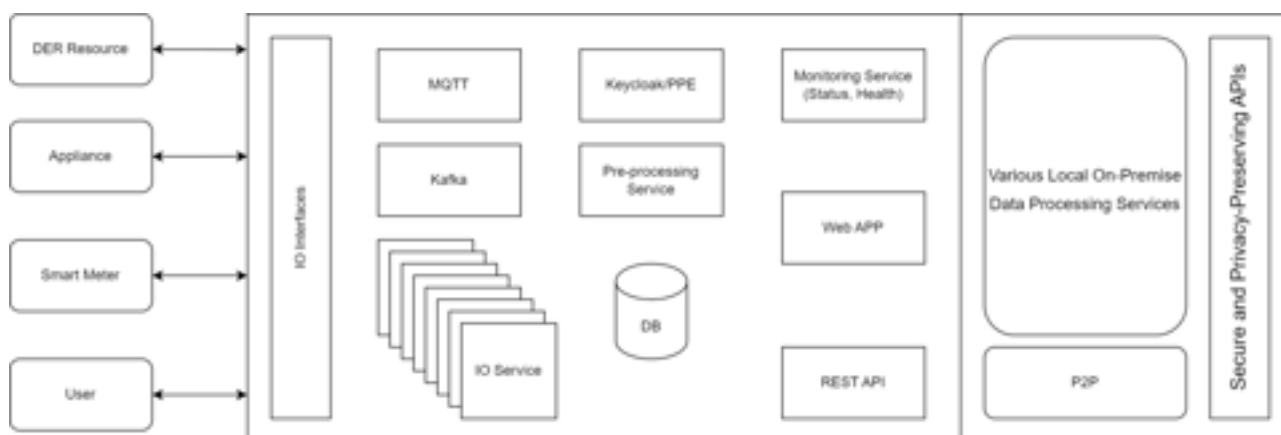


Figure 54: High-level architecture of complete USG software platform integration.

8.1.1.1. IO Services

IO services are custom and customisable microservices deployed as lightweight Docker Containers on UGW (Universal GateWay) software platform that enable virtualisation, data collection and control signal dispatching to individual energy assets such as HVAC units, battery storage devices, PV inverters, EV charging stations, energy meters and other devices and data sources. Each type of device or asset needs an individual IO service implementation to model the specific asset functionality. Each unit has to implement a translation of data and signals into a common USG data model, which is compatible with the DEDALUS data model and ontology developed in T3.3.

8.1.1.2. MQTT

MQTT is a publish-subscribe network protocol used for asynchronous data collection and data distribution with Internet of Things (IoT) systems. It is especially convenient for use when the lightweight communication protocol with a small code footprint and minimal network bandwidth is required. It is used to connect small remote devices on unreliable and low-bandwidth connections to the USG platform. The protocol is implemented on top of a TCP/IP network stack but can run on any network protocol that provides ordered, lossless and bi-directional communication.



In a MQTT network two device types exist: MQTT broker and MQTT client. MQTT broker is a server that receives all messages from the MQTT clients (publishers) and forwards them to the subscribed MQTT clients (subscribers). Information that flows through the MQTT broker is hierarchically organized into topics. Each MQTT client that connects to the MQTT broker can subscribe to one or more MQTT topics. When new data is published on a particular topic, the MQTT broker distributes it to all subscribed MQTT clients. If there is no MQTT client subscribed for a particular topic, the MQTT broker discards the published data. The flow of data in MQTT network is presented in Figure 55.

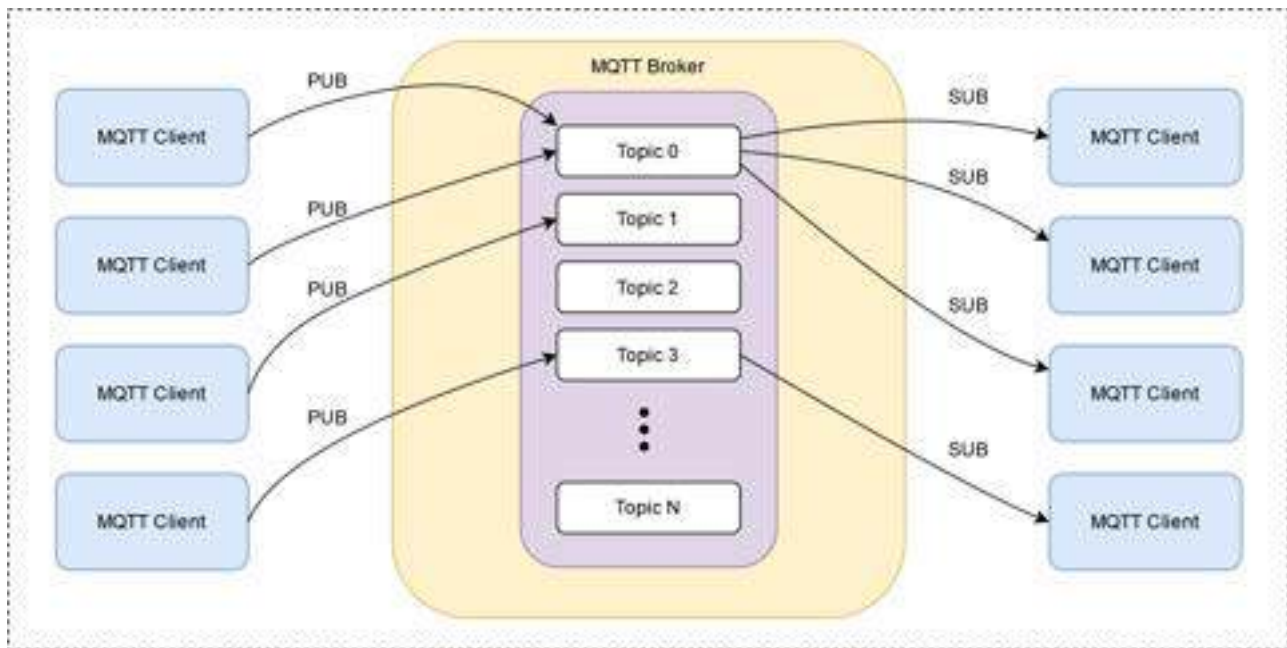


Figure 55: Visualization of data flow in MQTT system with MQTT clients and MQTT broker.

MQTT client is any device that runs MQTT protocol and can connect to a MQTT broker. MQTT client code footprint is very small and can be implemented on small, embedded devices with limited processing and memory resources.

MQTT protocol also specifies quality of service (QoS) measure, where for each connection between a MQTT client and a MQTT broker a level of QoS can be specified. Three QoS settings are specified, and they differ in the amount of communication overhead they impose according to the setting:

- Level 0: at most once – the message is sent only once without the delivery acknowledge (fire and forget).
- Level 1: at least once – the message is being retransmitted multiple times until acknowledge is received (acknowledged delivery).
- Level 2: Exactly once – the sender and receiver use a two-level handshake to ensure only one copy of the message is received (assured delivery).

There exist many implementations of MQTT brokers, such as Mosquitto MQTT broker, which is widely spread, especially amongst the hobbyists and less demanding deployments. In USG,

enterprise-grade MQTT broker EMQX is used, which provides great scalability, full MQTT standard compatibility with security and reliability. The main advantage of using EMQX in USG instead of Mosquitto is its high-performance capability even in non-scalable deployments, providing capability of handling of millions of MQTT messages on a single MQTT broker.

8.1.1.3. Kafka

Apache Kafka is a distributed streaming platform that is designed to handle large amounts of data in real-time. Apache Kafka is a message queueing system at its base, a powerful tool for building real-time data pipelines and streaming applications that can handle massive amounts of data. One of the key benefits of Kafka is its high throughput and low latency. It can handle millions of messages per second with very low latency, making it ideal for real-time applications where data needs to be processed quickly.

Kafka is useful for wide range of deployments because it is lightweight and easy to use. It can be set up quickly and easily on a single machine or a small cluster, and it can scale up as needed to handle larger workloads. Another advantage of Kafka is its ability to handle data streams from multiple sources and to integrate with a wide range of data processing and analytics tools. This makes it an ideal platform for building data pipelines that can collect and process data from a variety of sources, such as social media feeds, sensor networks, web applications, as well as from data collection gateways and devices.

At its core, similarly to MQTT, Kafka provides topics for channelized data exchange with possibility to split individual topics to partitions in case of high demand and dividing the data processing between several engines. In case of USG, it acts as a central messaging bus around which all existing and future internal microservices are built. It is flexible regarding maintainability and provides some temporary persistence. Temporary persistence can be set arbitrarily according to the needs in the system and can be set from the range of seconds to the range of months which is upwards limited by the memory and storage capacity of the USG host system. The persistence can act as a buffer in case of temporary increase of input data traffic with the pre-processing capabilities limitations or in case when, for example, pre-processing service is unavailable for a shorter period due to some errors or maintenance, the input data waits in a queue until the pre-processing service is able to consume it and therefore prevents the input data loss.

8.1.1.4. Data Pre-Processing Service

Data pre-processing service is currently just a microservice that is subscribed to all topics with input data on a Kafka broker and writes the data to the data base.



8.1.1.5. Database

Data is stored in a local relational PostgreSQL⁵⁶ database with TimescaleDB⁵⁷ plugin extension to support the efficient time series data storage. The relational data containing the user information, devices, device parameters, etc. are being saved into the PostgreSQL relational data storage, where periodic measurements from the smart meters, PV inverters, etc. are being stored as a time series data in a data storage managed by a TimescaleDB. The data storage model will be compliant with the data models and ontologies developed in task T3.3 to an extent to support the data collection and data processing needs deployed on an USG.

8.1.1.6. Web Application

The web application running on an USG provides user interfaces (UI) to each energy community member (user, consumer) and a community UI with administrative rights to the energy community manager. The web application works on data available in the USG database over the REST API for internal data access.

User UI gives the user an insight into his daily consumption and production statistics with the amount of imported energy from the grid, amount of exported energy to the grid, daily net energy export, energy costs, revenues, and CO₂ emissions. The daily statistics are complemented by the self-sufficiency and self-consumption performance parameters giving user information, how much of the energy consumed was provided by his own PV and how much of locally produced energy was consumed at the location. The part of user UI with daily statistical parameters is presented in Figure 56.

⁵⁶ <https://www.postgresql.org/>

⁵⁷ <https://www.timescale.com/>



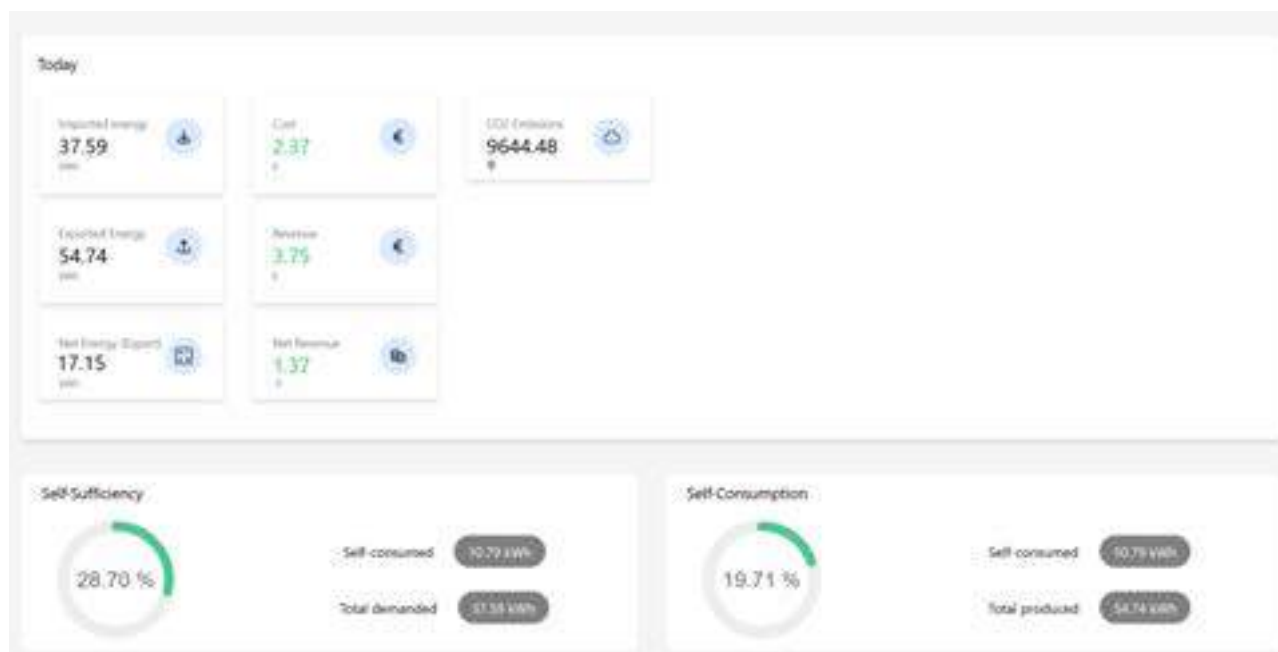


Figure 56: User UI presenting daily consumption and production statistics based on 15-minute smart meter data.

The next part of user's UI presents the user's prosumption profile against the forecasted prosumption profile for the current day which is presented in Figure 57, where data used for the preparation of prosumption profiles is based on prosumption measurements on a smart meter with a sampling interval of 15 minutes.

Community manager gets a similar UI but with aggregated values for all the members of the energy community. Community manager uses the aggregated values on a community-level to estimate the amount of the flexibility available for trading on energy and flexibility markets. Community manager also gets an admin UI part, where he/she can add new or remove existing energy community users.

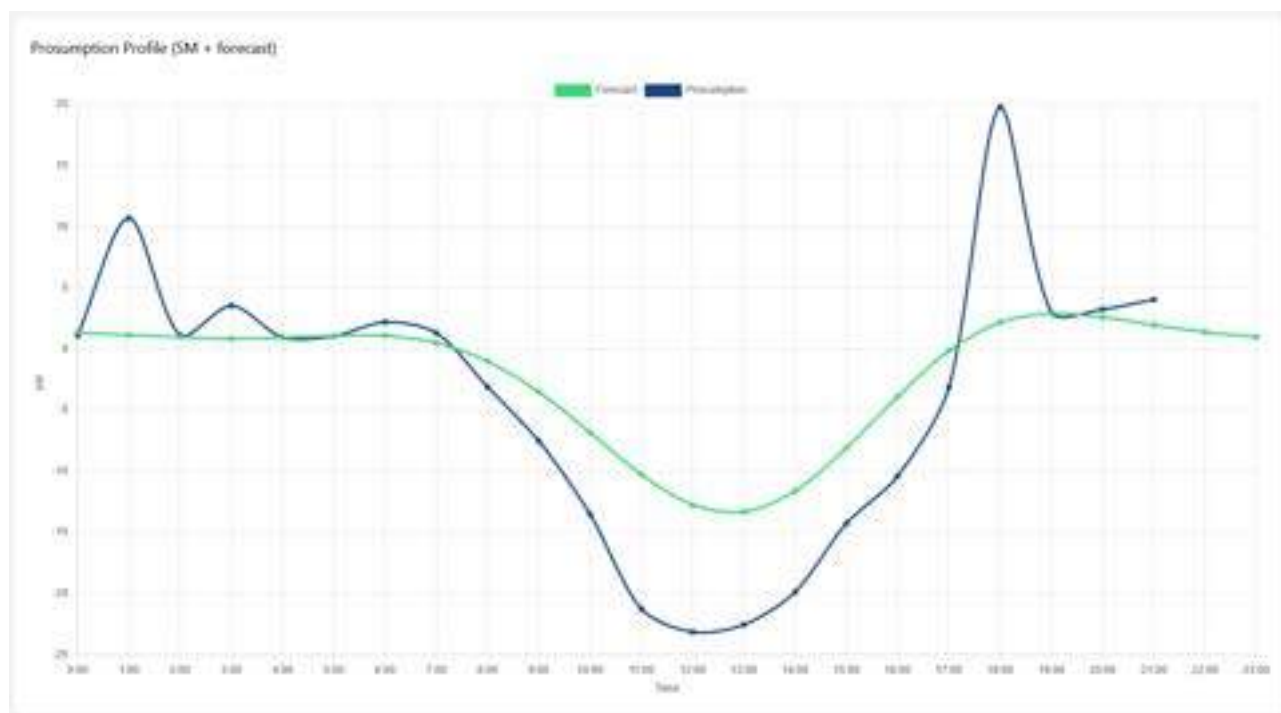


Figure 57: User UI presenting actual and forecasted prosumption profile for the current day based on the 15-minute smart meter data.

8.1.1.7. Monitoring Service

Monitoring service will provide information about the resource usage by the complete USG software package and performance parameters with status of individual USG microservices. This will enable the operator (e.g. community manager) to detect potential issues with individual components and potential lack of resources.

8.2. DEDALUS' privacy, security and data protection framework

Privacy and data protection are recognised as fundamental rights in EU, which makes the protection of privacy and personal data requirements that the project must comply with. According to article 4 of GDPR, personal data is defined as “any information relating to an identified or identifiable natural person (‘data subject’); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person”.

The identification of the privacy and data protection requirements should consider the following:

- Smart grid and smart meters belong to complex IT infrastructures that require not only to gather, collect, and analyse data, but also to exchange the same data with other IT



infrastructures. In that respect, considering the source from which such data will be gathered/collected/extracted it is possible that personal data, might be included in the information exchange cycle. Therefore, it is necessary to consider a trade-off among two conflicting interests (privacy and data protection on one side and security on the other side) is performed.

- Privacy and data protection rights are two fundamental rights recognized either at European level or at international level. In particular, the inclusion of the protection of personal data in EU primary legislative texts such as the Treaty of Functioning of the European Union (TFEU), as well as the Charter, enabled the same EU to provide for effective legislative instruments (e.g. among the other, GDPR) to protect personal data.

The interests or the values protected by the right to privacy and data protection rights for the sake of clarity are not the same. It is possible to say that the scopes of these two rights are different. Privacy can be considered as providing for a general prohibition of interference in the private life of an individual. On the other hand, the protection of personal data, can be instead intended as a complete system of rights, to be balanced and to be exercised and activated only when personal data, i.e. information that can directly or indirectly identify a person, are processed. This means also that data protection rules and requirements shall be held in check even when the processing personal data do not interfere with the privacy of the individual.

As of today, GDPR is the most relevant EU legislative source in terms of providing the rules and principles that should be respected when personal data are processed.

The main privacy and data protection requirements are presented in Table 13.

Table 13: Privacy and data protection requirements.

Req. ID	Requirement	Description
PR1	Transparency	The purposes of the data processing should appear clear and intelligible for the data subject. This can be ensured providing all the appropriate and necessary information to data subjects to exercise their rights, to data controllers to evaluate their processors, and to Data Protection Authorities to monitor according to responsibilities. The technology solutions and their data models should ensure that a data subject have easy access to that information at any time (also after the start of the data processing operations). All that information should be made available to the data subjects in a clear and intelligible way.
PR2	Lawful data collection	The data processing shall originate only from the personal data that have been collected with a lawful ground. Particular attention should be paid when implementing the components that will help to collect and get the data subject's consent. In this respect, the relevant partner should ensure the possibility to map the data flow. Particular attention should be given in case of secondary processing (even if, at the time of submission, this kind of operations are not foreseen).
PR3	Personal data collected are: • adequate,	The implementation of the principle of purpose limitation and data minimisation (core principles in GDPR) requires that the amount of data collected shall be proportionate to the purposes to be achieved, and at



	- proportionate, - relevant to the objectives of the system.	the same time, the purpose itself shall be legitimate. In this respect, data should be gathered if and only if it is strictly necessary for achieving the specified purpose and that data is “need to know”.
PR4	The personal data collected are accurate	The technology solutions should ensure that the data to be processed are accurate, i.e. data are correct and up to date in all details.
PR5	Storage limitation	The development team of the technology solution should define and implement an infrastructure to which it is possible to foresee for how long the personal data will be stored. Data subjects must be informed about it. In case the data are no longer necessary to fulfil the scope of technology solution such data should be immediately erased and/or anonymised according to the best standards and practices.
PR6	Procedures for granting individual rights	The components of the technology solutions should be designed taking also into consideration how concretely the relevant data subject might exercise his/her rights in connection with the data processing. The relevant partner should be aware of all the rights that GDPR grants to data subjects, and for each of them tailor a specific solution (e.g. data subjects have the right to rectify their data and to request their erasure).
PR7	Accountability principle and technical implementation	The implementation of the accountability principle entails that the technology solutions should allow a clear identification of the responsibilities related to the data processing. Examples of accountability measures are related to tracking of personal data access and tracking of communications with external systems. In addition, the accountability principle implies the set-up of internal audits and handle complaints procedures.
PR8	Implementation of security measures	Information security addresses integrity, confidentiality, and availability concerns. IT measures such as privacy enhancing technologies (PETs) represent an important tool for privacy and data protection, in terms of implementing technology solutions able to restrict access to personal data only to authorized people (e.g. permissions), and to ensure that the data is trustworthy and accurate (e.g. based on provenance information). The relevant partner should: (i) Regularly conduct privacy risk assessment and audit processes. (ii) Regularly run reviews of the security measures implemented. (iii) Design an ad hoc procedure to be followed in case of data breach.

8.2.1. Privacy Protection Enforcement

General Data Protection Regulation (GDPR) is a law regulation in European Union (EU) governing data protection, privacy and transfer of personal data. Privacy Protection Enforcement (PPE) component is the main component that will ensure the compliance with rules and policies. It serves as an authority and a mediator between entities with assigned roles such as data subjects, data controllers and data processors. The PPE’s position in the process of GDPR-compliant data exchange is presented in Figure 58.



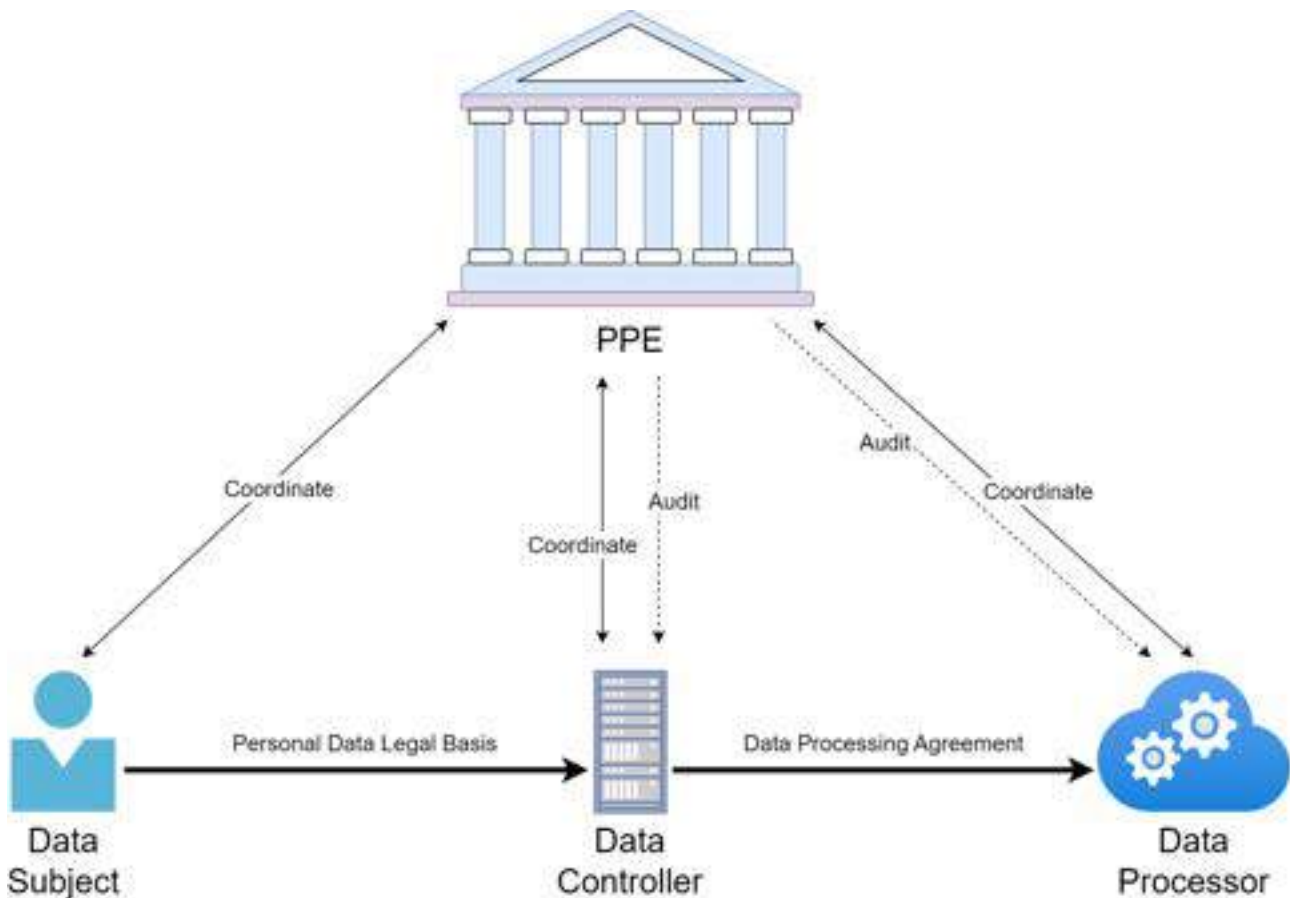


Figure 58: PPE as a main authority in the process of data exchange compliant to data protection and privacy preserving rules and policies.

Consider revising for clarity: The processing of personal data must either be explicitly allowed by law or require consent from the involved data subject. Consent is one of the legal bases for data processing introduced by a GDPR. A data subject must provide consent prior to any personal data collection and processing. Consent of the data subject as defined in Article 4 of GDPR is “any freely given, specific, informed and unambiguous indication of the data subject’s wishes by which he or she, by a statement or by a clear affirmative action, signifies agreement to the processing of personal data relation to him or her” [8].

Consent must be willingly and clearly given for it to be valid under GDPR. This means the person must freely offer it without any coercion, fully understanding what they're agreeing to, specifically what data will be used and how. They need to know who is collecting their data and why, including their right to withdraw consent anytime. Consent requires a clear, affirmative action, meaning it can't just be assumed. While there's no set duration for consent, revoking it should be as easy as giving it. If a third party provides consent, they must prove their authority to do so.

8.2.1.1. Privacy, Reputation and Auditability

Privacy can be represented by the ability to hide the information and disseminate information by choice, especially when the information is sensitive. Any data handling is associated with a certain degree of risk of data leakage. Data can be exposed both while in storage as well as during transit. Adequate security measures must be developed to prevent unauthorized data access. Data access policies may be used to define the responsibilities and roles of actors who are granted access to data, while data encryption may be used as an effective method for data protection. Data Encryption is a process of encoding information by converting a plaintext into a ciphertext usually performed by algorithms based on pseudo-random keys.

Privacy protection can also be achieved through anonymization, which involves removing identifying and quasi-identifying attributes from data. It can be also achieved by replacing the identifiers by pseudonymity but the permanent unique identifiers can still pose a privacy risk. To prevent identifying the devices or persons, changeable identifiers such as decentralized identifiers were proposed [9]. Decentralized identifiers have no central controlling authority and are managed by identity owners or their appointed trusted entity. This concept is known as self-sovereign identities [10].

The fundamental goal of leveraging reputation is formation of trust among unfamiliar entities which is directly proportional to the perceived risks. A reputation system assesses the trustworthiness of actor based on his past actions.

Auditability is defined as auditor's capacity to realize a comprehensive review of records. The rise of blockchain technologies enables improved accountability and audit capabilities. Blockchain technologies ensure authenticity of records and non-repudiation using public key infrastructure and digital signatures. The immutability of ledgers with the transactions provides integrity by preventing alteration of the records.

8.2.2. PPE Component Architecture

The Privacy Protection Enforcement (PPE) enables handling of data in compliance with EU Regulatory Framework and is built on top of a PPE DLT. The PPE component functionality is split into two separate subcomponents:

- **Privacy, Reputation Mechanism and Mutual Auditability (PRIMULA)** component enabling consent-driven exchange of data through the advanced consent among parties via smart contracts. PRIMULA provides consent auditing functionalities and reputation mechanisms to assess the party's reputation based on global perception of its behavior.
- **Compliant Regulatory Data Exchange (CoReDEX)** component enforces compliance rules and governance policies by acting as a mediator for data exchange among parties. CoReDEX implements appropriate mechanisms for data transfer between two parties without intermediary channels. The authorization depends on advanced consent and reputation mechanisms defined by PRIMULA.



8.2.2.1. PRIMULA Component

PRIMULA enables consent-driven exchange of personal and privacy-sensitive data through managing the advanced consent among parties via smart contracts on a common PPE DLT. It provides also consent auditing and reputation tracking mechanisms to assess the party's reputation. It supports the following parties:

- Data subject (DS) – data owner and consumer of provided services.
- Data controller (DC) – collecting and managing data.
- Data processor (DP) – data consumer and service provider.

PRIMULA is designed on top of a privacy-preserving blockchain for exchanging personal and confidential data. It is based on three ledgers for managing consents, assessing reputation and logging transactions related to the system. All entities in the system are defined by a pair of keys: private key and public key.

PRIMULA's architecture is composed of the following components:

- DLT on a blockchain on which PRIMULA is implemented:
 - Data and Advanced Consent Registry provides mechanisms for registering personal/confidential data and specification of consents.
 - Audit Logs ensure traceability by logging all interactions.
 - Reputation Assessment assesses the actors in the system and computes their reputation scores.
- Notification and Request System enables sending and retrieval of consent requests and various notifications.
- Advanced Consent API exposes PRIMULA's functionalities via specific API.

8.2.2.2. CoReDEX Component

The Compliant Regulatory Data Exchange (CoReDEX) provide functionalities the ensure the data is exchanged in compliance with the current EU Regulatory Framework which includes GDPR as well. The definition of the component is based on the analysis of the activities carried out by the data subject (DS) and the data controller (DC).

CoReDEX component is involved in the privacy protection enforcement process by providing three main features:

- Permission Management: it will agree or disagree on data exchange between parties based on consent status provided by PRIMULA.
- Notification: parties will be notified about specific events such as data access requests or data breaches.
- Data Access Tracking: data access will be constantly monitored and logged inside the PPE blockchain.

CoReDEX component is implemented using advanced, production-oriented technologies:



- CoReDEX Smart Contracts: PPE component interacts with the blockchain by creating and storing transactions through the use of a set of smart contracts (SCs). SCs are designed in such a way that they can be invoked from outside the blockchain.
- CoReDEX Logic: this component implements and exposes CoReDEX API.
- PPE Message Broker: message broker is used to facilitate communications between PPE subcomponents.

The diagram in Figure 59 describes the chronological sequence of messages exchanged between CoReDEX components and other components external to the PPE. It presents the events associated to a data exchange request coming from data processor component:

- Data Exchange Request (DEReq): the component sends an explicit request to CoReDEX component by calling specific API.
- Permission Request (PReq): CoReDEX Logic receives the request and uses the information contained within to contact the responsible smart contract.
- Permission Response (PRes): the responsible smart contract checks for the status of the consent object associated to the data and generates new permission object based on that status. The generated permission is stored on the blockchain and returned to CoReDEX Logic component.
- Data Exchange Response (DERes): once received from the smart contract, the permission object is forwarded to the component that asked for data exchange.

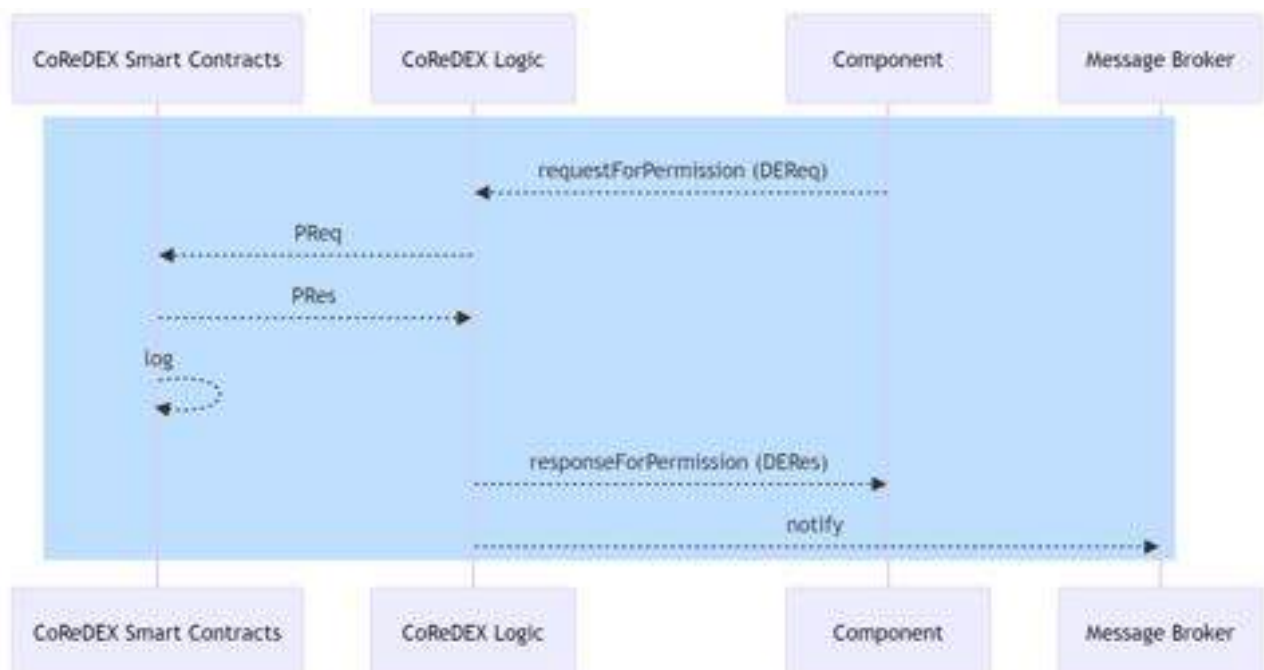


Figure 59: CoReDEX Sequence Diagram.



8.3. Next steps

During the following months of development of this task, the privacy, security and data protection framework will be integrated into the universal secure gateway, enabling trusted data sharing among components and actors. Universal secure gateway will be further adapted to support deployment of various DEDALUS services by partially adopting the DEDALUS data models and providing API for internal data access. Further, the blockchain technology developed in task T3.5 will potentially be adopted to replace the ledgers for the privacy protection enforcement component that will enable data and consent registration, transaction auditing and actors' reputation assessment currently used in the privacy protection enforcement component. The development can also provide the interface for identity and access management by hosting the fine-grained information on identities and consents for specific data sources used in DEDALUS project. Specific interfaces will be developed for identity and access management to provide human-readable interfaces for user and access management inside the energy community.



Conclusions

Throughout the first year of the DEDALUS project, WP3 partners have diligently worked on the different tasks, detailing not only the work done but also the next steps to be carried out in the coming months. The technologies to be used in each case have been studied, and a selection of the most appropriate ones has also been made. Subsequently, the development of each technology enabler has begun and progressed, with a proof of concept achieved in some cases. This deliverable includes very detailed information on all the work done in this sense, having one specific chapter for each task. Additionally, one additional chapter (chapter 2) has been added in order to contextualize each development within the current architecture.

Chapter 3 focuses on the work undertaken for T3.1, which is devoted to the ARC IoT platform, which will be deployed in two pilots (Greek and Spanish ones). Chapter 4 is devoted to T3.2, which is working on the definition of the Energy Data Space compliant interoperability layer. Chapter 5 is dedicated to T3.3, aimed at defining the DEDALUS ontology/data model and providing guidelines on its use across the different developments. From its side, chapter 6 is focused on T3.4, containing information on multi-energy flexibility modelling for data-driven residential DR. Chapter 7 explores T3.5, which is devoted to the use of blockchain for flexibility tokenization and demand response settlement. Finally, chapter 8 outlines the work carried out within T3.6, focused on establishing the foundation for privacy, security and the DEDALUS data protection framework for trusting data sharing and AI.

Moving forward, these developments will continue to evolve, with careful consideration of the needs from the different high-level services, as well as the requirements and specifications coming from the different pilots. Detailed information on these ongoing works will be included in coming deliverables (D3.2 and D3.3).

Considering all the activities described in this document, we can confidently state that we have successfully achieved MS4 – the completion of the 1st technology development cycle, marking a significant milestone in the project's progress.



References

- [1] Nagel, Lars & Lycklama, Douwe & Ahle, Ulrich & Bastiaansen, Harrie & Bengtsson, Kjell & Caballero, Mallku & Castellvi, Silvia & Dognini, Alberto & Ette, Frans & Faraldi, Marianna & Gelhaar, Joshua & Graziani, Alessio & Grguric, Andrej & Gusmeroli, Sergio & Helmholt, Kristian & Hierro, Juan & Hoppenbrouwer, Denise & Hülsmann, Thorsten & Krco, Srdjan & Valiño, Javier. (2021). Design Principles for Data Spaces – Position Paper, [Online]. Available: <https://www.opendei.eu/wp-content/uploads/2022/03/Position-Paper-Design-Principles-for-Data-Spaces.pdf>
- [2] Celesti, Antonio & Fazio, Maria & Galán, Fermín & Glikson, Alex & Mauwa, Hope & Bagula, Antoine & Celesti, Fabrizio & Villari, Massimo., “How to Develop IoT Cloud e-Health Systems Based on FIWARE: A Lesson Learnt,” Journal of Sensor and Actuator Networks. 8. 7, 2019, [Online]. Available: <https://www.mdpi.com/2224-2708/8/1/7>
- [3] Euzenat, J., Mocan, A. and Scharffe, F., 2008. Ontology Alignments. In: M. Hepp, P. De Leenheer, A. De Moor and Y. Sure, ed., Ontology Management, 1st ed. New York City: Springer, pp.177-206.
- [4] Hajo Rijgersberg, Mark van Assem, and Jan Top. 2013. Ontology of units of measure and related concepts. Semant. web 4, 1 (January 2013), 3–13.
- [5] Reynders G, Diriken J, Saelens D. Generic characterization method for energy flexibility: applied to structural thermal storage in residential buildings. Appl Energy 2017;198:192–202. doi:10.1016/j.apenergy.2017.04.061.
- [6] Bampoulas D, Pallonetto F, Mangina E, Finn D. An ensemble learning-based framework for assessing the energy flexibility of residential buildings with multicomponent energy systems. Applied Energy, Volume 315, 2022, 118947, ISSN 0306-2619, doi:10.1016/j.apenergy.2022.118947.
- [7] Chen Y, Xu P, Gu J, Schmidt F, Li W. Measures to improve energy demand flexibility in buildings for demand response (DR): a review. Energy Build 2018;177:125–39. doi:10.1016/j.enbuild.2018.08.003
- [8] B. Wolford, “What are the GDPR consent requirements? - GDPR.eu,” [Online]. Available: <https://gdpr.eu/gdpr-consent-requirements/>. [Accessed 14 December 2020].
- [9] Y. Kortessniemi, D. Lagutin, T. Elo and N. Fotiou, “Improving the privacy of IoT with decentralised identifiers (DIDs),” Journal of Computer Networks and Communications 2019, 2019.
- [10] C. Allen, “The Path to Self-Sovereign Identity,” 25 April 2016. [Online]. Available: <http://www.lifewithalacrity.com/2016/04/the-path-to-self-sovereign-identity.html>

