



European Common Energy Data Space Framework Enabling Data Sharing - Driven Across – and Beyond – Energy Services

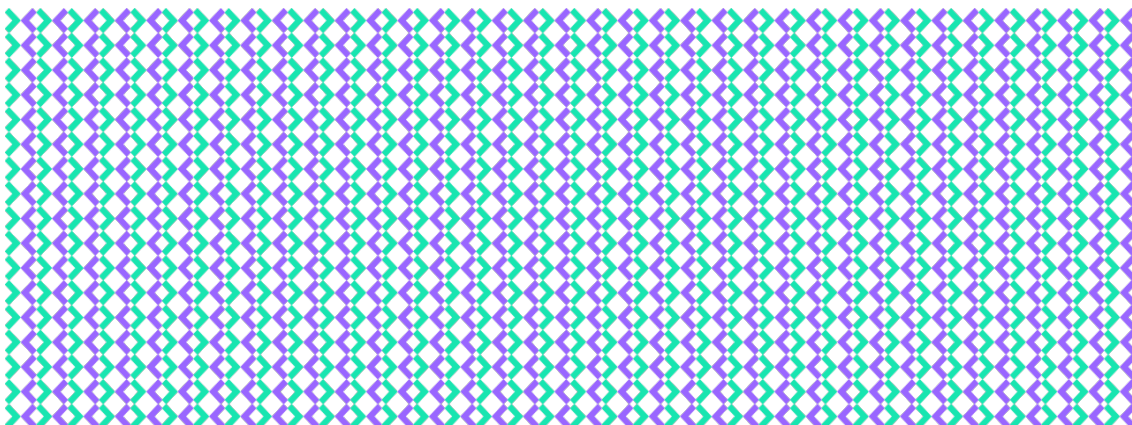
enershare.eu



ENERSHARE has received funding from European Union's Horizon Europe Research and Innovation programme under the Grant Agreement No 101069831

D6.2 Federated learning, data-driven services, data visualisation and Digital Twins

Beta version



Publication details

Grant Agreement Number 101069831

Acronym ENERSHARE

Full Title	European Common Energy Data Space Framework Enabling Data Sharing-Driven Across — and Beyond — Energy Services
Topic	HORIZON-CL5-2021-D3-01-01 'Establish the grounds for a common European energy data space'
Funding scheme	HORIZON-IA: Innovation Action
Start Date	Jul 1, 2022
Duration	36 months
Project URL	enershare.eu
Project Coordinator	Engineering
Deliverable	D6.2 – Federated learning, data-driven services, data visualisation and Digital Twins (Beta version)
Work Package	WP6 – Cross-value chain AI-based data-driven services
Delivery Month (DoA)	M18
Version	1.0
Actual Delivery Date	31 January, 2024
Nature	Report
Dissemination Level	PU



ENERSHARE has received funding from [European Union's Horizon Europe Research and Innovation programme](#) under the Grant Agreement No 101069831

Lead Beneficiary	INESC TEC
Authors	Ricardo Bessa (INESC TEC), José Villar (INESC TEC), Pedro Macedo (INESC TEC), Armando Moreno (INESC TEC), Ricardo Silva (INESC TEC), José Paulos (INESC TEC), Carla Gonçalves (INESC TEC), João Viana (INESC TEC), Ricardo Andrade (INESC TEC), Elena Lazovik (TNO), Selma Causevic (TNO), Shreshtha Sharma (TNO), Syrine ben Aziza (TNO), David Campo (FIWARE), Alberto Abella (FIWARE), Leonardo Carreras (RWTH), Chijioke Eze (RWTH), Francesco Bellesini (EMOT), Edoardo Mancinelli (EMOT), Oudom Kem (ENGIE), Weiqin XU (ENGIE), Andrej Čampa (COMS), Blaž Bertalanč (COMS), Ioannis Dimanidis (FORTUM), Rui Martins (SEL), Marco Ermidas (SEL), Adelaide Ambrósio (SEL), Vassilis Sakas (ED), Apostolos Kapetanios (ED), Konstantinos Kotsalos (ED), Ainhoa Pujana (TECNALIA), Antonio Kung (Trialog), Diana Jimenez (Trialog), Tine Mlač (ENVI)
Quality Reviewer(s)	Volker Berkhout (FhG) Ainhoa Pujana Goitia (TECNALIA)
Keywords	Federated learning, data-driven energy services, cross-sector services, visual analytics, digital twin



Document History

Ver.	Date	Description	Author	Partner
0.1	Sept 22, 2023	ToC	Ricardo Bessa	INESC TEC
0.2	Dec 22, 2023	With all contributions (for WP6 leader review)	All WP6 partners	All partners
0.3	Dec 27, 2023	Review of the WP6 leader	Ricardo Bessa	INESC TEC
0.4	Jan 15, 2024	Official review	Volker Berkhout and Ainhoa Pujana Goitia	FhG, TECNALIA
0.5	Jan 19, 2024	Revised version	Ricardo Bessa and WP6 partners	INESC TEC et al.
1.0	Jan 24, 2024	Final version	Ricardo Bessa	INESC TEC

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.



ENERSHARE has received funding from [European Union's Horizon Europe Research and Innovation programme](#) under the Grant Agreement No 101069831

Table of Contents

1	Introduction	29
1.1	About this document	30
1.2	Intended audience	30
1.3	Reading recommendations.....	30
2	Federated Learning Methods	32
2.1	Knowledge-driven federated learning.....	33
2.1.1	Development progress.....	33
2.1.2	Key features and documentation	35
2.1.3	End-to-end FL workflow.....	35
2.1.3	Next steps	37
2.2	Federated learning framework.....	38
2.2.1	Development progress.....	38
2.2.2	Documentation	45
2.2.3	Examples of running services on pilot 3 data	51
2.2.4	Integration with the Data Space	52
2.2.5	Next steps	54
2.3	Multivariate energy time series forecasting.....	54
2.3.1	Development progress.....	54
2.3.1.1	Initialization	56
2.3.1.2	Encryption.....	57
2.3.1.3	Model estimation.....	58
2.3.1.4	Forecast generation	60
2.3.1.5	Tests and illustrative results	61
2.3.2	Documentation	63
2.3.3	Integration with the Data Space	64
2.3.4	Next steps	66
2.4	Federated transfer learning flexibility potential assessment.....	67



2.4.1	Development progress.....	67
2.4.2	Integration examples of the running service on pilot’s data.....	68
2.4.3	Integration with the Data Space.....	72
2.4.4	Next steps	72
2.5	Comparison between federated learning methods	73
2.6	Collaboration with Eclipse privacy model working group	74
3	Data-driven Energy Services	75
3.1	Energy community sizing with assets sharing	77
3.1.1	Development progress.....	77
3.1.2	Documentation updates.....	78
3.1.3	Integration with the Data Space.....	78
3.1.4	Next steps	86
3.2	Flexibility modelling of thermoelectric water heaters	87
3.2.1	Development progress.....	87
3.2.1.1	Built-in load to binary usage converter	87
3.2.1.2	Parameters and converter refinement using real data	89
3.2.1.3	Service outputs	89
3.2.2	Documentation	90
3.2.3	Integration with the Data Space.....	90
3.2.4	Next steps	94
3.3	Community market pool to estimate energy price of internal transactions.....	94
3.3.1	Development progress.....	94
3.3.2	Documentation	95
3.3.3	Integration with the Data Space.....	97
3.3.4	Next steps	101
3.4	Data-driven failure detection algorithms for wind turbine components.....	101
3.4.1	Development progress.....	101
3.4.1.1	Anomaly detection of gearbox	101
3.4.1.2	Anomaly detection of electric generator.....	102





- 3.4.1.3 Anomaly detection of hydraulic pitch system 103
- 3.4.2 Documentation 104
- 3.4.3 Integration with the Data Space 104
- 3.4.4 Next steps 105
- 3.5 Substation Load forecasting tool 106
 - 3.5.1 Development progress..... 106
 - 3.5.2 Documentation 109
 - 3.5.3 Integration with the data Space 109
 - 3.5.4 Next steps 110
- 3.6 Energy Usage Prediction Service 111
 - 3.6.1 Development progress..... 111
 - 3.6.2 Documentation updates 114
 - 3.6.3 Plan to integrate service into Data Space 115
 - 3.6.4 Next steps 116
- 3.7 Service local load forecasts and estimation of electrical grid status..... 117
 - 3.7.1 Development progress..... 117
 - 3.7.2 Documentation 119
 - 3.7.3 Integration with the Data Space 119
 - 3.7.4 Next steps 120
- 3.8 Flexibility Analytics and Register (FAR) service..... 121
 - 3.8.1 Development progress..... 121
 - 3.8.2 Documentation updates 121
 - 3.8.3 Tests of the services with the Data Space 121
 - 3.8.4 Next steps 122
- 3.9 Aggregation of flexibility from end users 123
 - 3.9.1 Development progress..... 123
 - 3.9.1.1 Overview of developments..... 123
 - 3.9.1.2 New subsystems 124
 - 3.9.1.3 Technology Readiness Level 127



3.9.2	Documentation updates	128
3.9.3	Data Space integration plan.....	129
3.9.3.1	RESTful Data Space integration	129
3.9.3.2	Stream-based Data Space integration	130
3.9.4	Next steps	131
3.10	Data-driven management of surplus RES generation in distribution systems	132
3.10.1	Description of the service	132
3.10.2	Innovation.....	134
3.10.3	Functions.....	134
3.10.3.1	Machine Learning Module.....	134
3.10.3.2	Application Module	136
3.10.4	Input and Output Data Formats	138
3.10.5	Integration with the Data Space	138
3.10.6	Next steps	139
4	Data-driven Cross-Sector Services.....	140
4.1	Multi-energy flexibility potential assessment	141
4.1.1	Development progress.....	141
4.1.2	Documentation updates.....	143
4.1.3	Integration with the Data Space	144
4.1.4	Next steps	145
4.2	Cross-operators' portal (COP).....	145
4.2.1	Development progress.....	145
4.2.2	Documentation updates.....	145
4.2.3	Tests of the services with the Data Space	146
4.2.4	Next steps	147
4.3	Emissions and ecological footprint.....	147
4.3.1	Development progress.....	147
4.3.2	API documentation	148
4.3.3	Tests of the services with the Data Space	148



4.3.4	Next steps	149
4.4	EV charging monitoring and remote management	149
4.4.1	Development progress.....	149
4.4.2	Documentation updates	150
4.4.3	Integration with the Data Space	153
4.4.4	Next steps	154
4.5	ML-based models for assessing renovation actions in residential buildings (AI4EF)	154
4.5.1	Purpose and overview of AI4EF	154
4.5.2	Development progress.....	156
4.5.2.1	ML model developments	156
4.5.2.1.1	Data preprocessing	156
4.5.2.1.2	Model Training.....	159
4.5.2.1.3	Results.....	159
4.5.2.2	Software architecture	161
4.5.2.3	Front-end	163
4.5.2.3.1	Homepage.....	164
4.5.2.3.2	Investment Planning page	164
4.5.2.3.3	Photovoltaic Installation.....	165
4.5.2.4	Data model.....	166
4.5.3	Documentation	166
4.5.3.1	API description	166
4.5.3.1.1	Services API.....	166
4.5.3.1.2	Database API.....	169
4.5.4	Integration plan with the Data Space	171
4.5.4.1.1	MVP deployment	172
4.5.4.1.2	Internal Deployment.....	173
4.5.5	Next steps	173
4.6	Health insurance alarms for senior living alone	173
4.6.1	Development progress.....	173



4.6.2	Implementation details.....	174
4.6.3	Integration with the Data Space.....	178
4.6.4	Next steps	178
4.7	Appliances maintenance or retrofit.....	179
4.7.1	Development progress.....	179
4.7.2	Documentation updates.....	183
4.7.3	Tests of the services with the Data Space	183
4.7.4	Next steps	184
5	Data Visualisation	184
5.1	Visualisation Engine demo.....	184
5.1.1	Connection with external databases	185
5.1.2	File uploading.....	186
5.1.3	Dashboard creation	187
5.1.4	User and Role management	189
5.1.5	Deployment configuration.....	189
5.1.6	Connection with Data Spaces	191
5.1.7	User Testing and Feedback.....	193
5.2	Tiny spatial visualisations	195
5.2.1	Description of the service	195
5.2.2	Functions.....	196
5.2.2.1	Kernel smoothing.....	196
5.2.2.2	Grid aggregation	197
5.2.2.3	Contour lines or polygons.....	198
5.2.3	Connection with Data Space.....	200
5.2.4	Next steps	200
6	System-of-systems Integrated Digital Twins	201
6.1	System-of-system Digital Twins: ENERSHARE Approach.....	202
6.2	Digital Twin for optimal data-driven power-to-gas optimal planning.....	203
6.2.1	Changes since deliverable D6.1	204





- 6.2.1.1 Database development..... 205
 - 6.2.1.1.1 Database architecture 205
 - 6.2.1.1.2 User Interface of the database system..... 206
 - 6.2.1.1.3 Data model 208
 - 6.2.1.1.4 API..... 208
- 6.2.2 Experimental setup and challenges 209
 - 6.2.2.1 Additional P2G scenarios 209
 - 6.2.2.1.1 Datasets 210
 - 6.2.2.1.2 TwinP2G Configurations 211
 - 6.2.2.1.2.1 Configuration 1: No P2G (Baseline) 211
 - 6.2.2.1.2.2 Configuration 2: P2G1 211
 - 6.2.2.1.2.3 Configuration 3: P2G2 212
 - 6.2.2.1.3 Selection of simulation parameters..... 212
 - 6.2.2.1.4 Results..... 215
 - 6.2.2.1.4.1 Scenario 1: 2023 results and conclusions..... 215
 - 6.2.2.1.4.2 Scenario 2: 2033 projection results and conclusions..... 218
- 6.2.3 Interface specifications 222
- 6.2.4 Front-end 222
 - 6.2.4.1 User roles 222
 - 6.2.4.2 Demonstration 222
- 6.2.5 Requirements and deployment strategy for pilot feedback 224
- 6.2.6 Integration with ENERSHARE Data Space and Services 225
 - 6.2.6.1.1 MVP deployment 226
 - 6.2.6.1.2 Internal Deployment..... 227
- 6.3 Digital Twin based O&M algorithms and generation of synthetic failures data 227
 - 6.3.1 Changes since deliverable D6.1 227
 - 6.3.2 Interface specifications 228
 - 6.3.3 Requirements and deployment strategy for pilot feedback 229
 - 6.3.4 Integration with ENERSHARE Data Space and Services 229



6.4	Digital Twin for flexible energy networks	230
6.4.1	Changes since deliverable D6.1	231
6.4.2	Experimental setup and challenges	233
6.4.3	Interface specifications	236
6.4.4	Front-end	237
6.4.5	Requirements and deployment strategy for pilot feedback	237
6.4.6	Integration with ENERSHARE Data Space and Services	238
7	Final Remarks.....	240
8	References	241
9	Appendix	243
9.1	Flexibility Analytics and Register (FAR) service.....	243
9.1.1	FAR service User Interface features.....	243
9.1.2	FAR service integration with OneNet Connector.....	245
9.2	Cross-operators' portal (COP) service.....	246
9.2.1	COP User Interface features	246



List of Figures

Figure 1: Architecture of knowledge-driven federated learning platform.....	33
Figure 2: End-to-end workflow of the knowledge-driven FL method	36
Figure 3: Integration of the knowledge-driven FL method with the Data Space	37
Figure 4: Predictions of the version 1 according to the target values.....	38
Figure 5: Sample of the dataset from the three households for the period form 01-01-2021 till 01-01-2023.....	39
Figure 6: The configuration of the client with all its parameters	40
Figure 7: New hyper parameters for LSTM model for local energy consumption predictions ..	41
Figure 8: The training process of the LSTM model according to the configuration parameters	41
Figure 9: The predictions of the energy usage for the next 24 hours according to the day in a week for a specific household	42
Figure 10: The intermediate and final results from the global model based on the new version of the predictive local models.	43
Figure 11: Separation of the low-density population area to the zones within which the data swap can be performed.....	44
Figure 12: Docker commands to start the FL platform.....	45
Figure 13: FL strategy for the transformer overloading predictions with the configurable values for every FL parameter	46
Figure 14: FL parameter loading and converting dataset to framework dataframe.....	47
Figure 15: Converting the data to the time series format, providing the target window and the lookback and the steps.....	48
Figure 16: The initialization of the local client with the model and dataframe	48
Figure 17: Configuration of the returned results from local clients	48
Figure 18: Initialization of the simulation.....	49
Figure 19: Loading and initializing the client for InfluxDB to write the final results from the simulation	50
Figure 20: Writing process to InfluxDB.....	50



Figure 21: Screenshot of the table of InfluxDB with the results from the run of the simulation of the electrical grid	51
Figure 22: Estimated overloads of the electrical transformers of the global model running on the new version of the LSTM model.....	52
Figure 23: Sequence diagram of the integration of the Federated Learning platform and IDS connectors from different organizations.....	54
Figure 24: Log messages of the participants' registration.....	56
Figure 25: Log messages of the steps taken to perform collaborative data encryption	57
Figure 26: Log messages of the first peer in the chain interacting with the server API and with the other peer.....	58
Figure 27: The second peer in the chain interacting with the server API and receiving request from the first peer in the chain. In the end, closes encryption cycle	58
Figure 28: Log messages of the beginning of the model estimation task	59
Figure 29: First model estimation log messages from the first peer in the chain.....	59
Figure 30: A model estimation process still hasn't reached any stopping conditions and carries on	60
Figure 31: A model estimation process finishes by reaching the maximum number of iterations	60
Figure 32: Log messages of the beginning of the forecasting generation task	61
Figure 33: Log messages for the forecasting process on the client side	61
Figure 34: Example of forecasts for 5 wind parks using three different approaches for the FL approach.....	62
Figure 35: Plan for integration of Multivariate energy time series Federated Learning forecasting service with Data Spaces (server integration and options for client integration).....	64
Figure 36: Example of a digital twin model for federated transfer learning	68
Figure 37: An example of a pipeline in the digital twin of Federated TL framework visualized with ZenML	69
Figure 38: Example of the forecasting performance for flexibility assessment service	69
Figure 39: An example of execution of automated pipelines for federated transfer learning ..	70
Figure 40: An example of built pipeline for ZenML	71
Figure 41 Data spaces integration diagram	72
Figure 42: Energy community sizing, REC pricing and EWH flexibility services integration with the ENERSHARE Data Space (high level overview)	78
Figure 43: Sizing Service: interaction with Data Space	81



Figure 44: Parameters interaction with Data Space	82
Figure 45: Units and Formats of parameters.....	84
Figure 46: Default values and sources of parameters	85
Figure 47: Sizing API outputs	86
Figure 48: EWH Load to Usage converter visualisation example	89
Figure 49: Sequence diagram for the interaction with the EWH Flexibility REST API	91
Figure 50: GitLab repository where the Python library developed under the use case for internal REC transactions price estimation is stored	96
Figure 51: Install and test guide for the library	96
Figure 52: Interactive API documentation provided by Swagger UI	97
Figure 53: Sequence diagram for the interaction with the REC LEM prices API	99
Figure 54: Schematic of the Data Space integration	105
Figure 55: Pivoting of net-load time series into daily observations and quarter-hourly features	106
Figure 56: Clustering and prediction processes in the PSF algorithm (Martínez-Alvarez et al., 2010).....	107
Figure 57: Pattern Sequence matching and generation of forecasts (Martínez-Alvarez et al., 2010).....	107
Figure 58: Integration plan for the substation load forecasting tool in the Data Space	110
Figure 59: Program flow for training and forecasting data-driven user profiles.....	112
Figure 60: forecasted (blue curve) and measured (black dots) district heating profile	113
Figure 61: Daily forecasted heat profile of the house.	113
Figure 62: IDS Integration schema and possible interactions inside the pilot.	116
Figure 63: A graphical interface of the energy grid MapEditor with the example of the topology from the low-density populated geographical area	118
Figure 64: Architecture of the integration of the Federated Learning platform and IDS connectors from different organizations.....	120
Figure 65: Architecture of the integration of the FAR with Enershare data space using OneNet connector (NGSI-LD APIs)	122
Figure 66: Overview of components comprising the Demand Response Service	125
Figure 67: Simulated response to a triangular frequency signal of a charger fleet of 67 units with baseline consumption of 225 kW and a flexibility potential of 150kW.....	128



Figure 68: RESTful Data Space integration plan for the aggregation of flexibility from end-users service	130
Figure 69: Websocket-based Data Space integration plan for the aggregation of flexibility from end-users service	131
Figure 70: Sample net load and PV generation data	133
Figure 71: Comparison between the actual data and the predictions by the multivariate spatio-temporal graph model.....	133
Figure 72: Machine learning module of data-driven service for managing surplus renewable energy generation in distribution systems	135
Figure 73: Application module component of data-driven service for managing surplus renewable energy generation in distribution systems.....	137
Figure 74: Architecture of Integration of data-driven service for managing surplus renewable energy generation in distribution systems with the ENERSHARE Data Space	139
Figure 75: UI of Multi-energy flexibility potential assessment service with marked new features.	142
Figure 76: Comparing calculated profiles generated using historical weather data (PVGIS, left graphs) and weather forecast (right graph) for zone location of Slovenian pilot	143
Figure 77: IDS Integration schema and possible interactions inside the pilot	145
Figure 78: Architecture of the integration of the COP with Enershare data space using OneNet connector (NGSI-LD APIs)	146
Figure 79: Schematic of the Data Space integration for the emissions and ecological footprint service	149
Figure 80: On-Board Diagnostic Device (OBD) connected to EV diagnostic interface	150
Figure 81: Schematic for “EV charging monitoring and remote management” service integration with the Data Space.....	154
Figure 82: Count plots describing the frequency of observations in our features in service 1.	157
Figure 83: Count plots describing the frequency of observations in the "Region" feature in service 2	158
Figure 84: Architectural diagram of the designed technology (deployment)	161
Figure 85: AI4EF dashboard - Homepage (screenshot)	164
Figure 86: AI4EF - Investment Planning page (screenshot)	165
Figure 87: AI4EF - Photovoltaic Installation Planning page (screenshot)	166
Figure 88: Data Space integration plan for AI4EF and pilot 7.....	171



Figure 89: Data acquisition system architecture	175
Figure 90: Operations management system architecture.....	176
Figure 91: System diagram for “health insurance alarms for senior living alone” service integration with the Data Space	178
Figure 92: The proposed transformer-based architecture for energy disaggregation	181
Figure 93: Optimal window length estimation pipeline	182
Figure 94: System diagram of the integration of service “appliances maintenance or retrofit” with Data Space	183
Figure 95: Visualisation Engine - Database connection details required	186
Figure 96: Visualisation Engine - File upload form	187
Figure 97: Visualisation Engine - Dashboard with pilot 7 dataset.....	188
Figure 98: Visualisation Engine - New dashboard page.....	189
Figure 99: Visualisation Engine connection schema.....	193
Figure 100: Testing sample example 1	197
Figure 101: Testing sample example 2	198
Figure 102: Contour lines testing sample example.....	199
Figure 103: Contour polygons testing sample example.....	199
Figure 104: IDS integration schema.....	200
Figure 105: The high-level architecture of TwinP2G	203
Figure 106: Landing page of the database system's web UI.	206
Figure 107: Exploring the individual runs in Dagster.....	207
Figure 108: Run inspection page in Dagster	208
Figure 109: Different P2G configurations	211
Figure 110: Electric power generation timeseries for Scenario	216
Figure 111: Energy mix timeseries for Scenario 1	217
Figure 112: Natural gas imports or production for Scenario 1	217
Figure 113: CO2 emissions timeseries for Scenario 1.....	217
Figure 114: Electricity generation Scenario 2: 2033	219
Figure 115. Energy mix timeseries for Scenario 2	220
Figure 116. Green hydrogen storage charge and discharge timeseries for Scenario 2.....	220
Figure 117. Green hydrogen storage level timeseries for Scenario 2	220
Figure 118. Green hydrogen injection into natural gas grid timeseries for Scenario 2.....	221



Figure 119. Natural gas imports or generation timeseries for Scenario 2	221
Figure 120. CO2 emissions timeseries for Scenario 2.....	222
Figure 121: Definition of customizable use cases via Streamlit front end.	223
Figure 122. Adding buses, connect generators, and import generation timeseries.	223
Figure 123. Addition of power lines	223
Figure 124. Connection of power loads to buses and insertion loads timeseries.....	223
Figure 125. Insertion of P2G components and customization of their features	224
Figure 126. Simulation results timeseries selection	224
Figure 127: Data Space integration plan for TwinP2G and pilot 4	225
Figure 128: Simulink model for PMSG and hydraulic pitch system.....	228
Figure 129: High level overview for the DT for wind energy O&M integration with the Data Space	230
Figure 130: High-level architecture of the DT for flexibility in electrical networks.....	231
Figure 131: Example graphics using VILLASweb and VILLAScontroller	237
Figure 132: Screenshot of the OpenStack dashboard	238
Figure 133: Data Space integration of the Digital Twin for Flexible Energy Networks.....	239
Figure 134: User Interface for storing locally (i.e., at Aggregator or User system) data on resources/resource groups.....	244
Figure 135: Flexibility Analytics tab, visual dashboard and “Flexibility Analytics” button integrated with OneNet Connector.....	244
Figure 136: Architecture of the integration of the FAR with Enershare data space using OneNet connector (NGSI-LD APIs)	244
Figure 137: Consent management tab, Flexibility Service Provider representing resources...245	
Figure 138: OneNet Connector UI: Available services on the ecosystem (FAR and COP service providers).....	245
Figure 139: OneNet Connector UI: Available services on the ecosystem (FAR provider’s service offering)	246
Figure 140: Service request for subscription from user ed-new1 to FAR_provider.....	246
Figure 141: Available data analytics to be retrieved from FAR service	246
Figure 142: Grid Analytics tab on COP provided application.....	247
Figure 143: Resources groups tab on the electricity grid	247
Figure 144: Available resources on the electricity grid	247



List of Tables

Table 1: Overview of the different federated learning releases in WP6.....	32
Table 2: Configuration parameters for LSTM model for local predictions.....	40
Table 3: The results for the local predictions of 5 households after 5 training runs with the relevant MAE and the last improvement of the model as the next step will be normalization of the input data to decrease the loss (MAE value) to minimum.....	51
Table 4: Comparison (in terms of features) between the different ENERSHARE FL services.....	73
Table 5: Overview of the different data-driven energy services releases in WP6	75
Table 6: EWH Flexibility Data Space request detail	92
Table 7: EWH Flexibility Data Space output detail	93
Table 8: Information required by the REC LEM pricing service	100
Table 9: Overview of the different data-driven cross-sector services releases in WP6.....	140
Table 10: Features and targets of the two ML services developed.....	158
Table 11: Classification Report for service 1 model	160
Table 12: Evaluation metrics for service 2 model.....	160
Table 13: docker-compose file for PostgreSQL DB and FAST API backend	191
Table 14: Overview of the different digital twins releases in WP6	201
Table 15: Digital Twin and Data Spaces aspects.....	202
Table 16: P2G scenarios parameters	214
Table 17: Simulations investment results.....	215
Table 18: Digital Twin for flexible energy networks TRL status.....	232



List of Acronyms

Acronym	Description
AE	Autoencoder
AI	Artificial Intelligence
ALPG	Artificial Load Profile Generator
API	Application Programming Interface
BN	Bayesian Network
BTM	Behind the Meter
Cbc	Coin-or branch and cut
CEC	Citizens Energy Community
CM	Condition Monitoring
COP	Cross-sector Operators' Portal
CPO	Charging Point Operators
DER	Distributed Energy Resources
DIHN	Directly Influenced Hypernodes
DNN	Deep Neural Network
DP	Differential Privacy
DSO	Distribution System Operator
DT	Digital Twin
EV	Electric Vehicle
EWH	Thermoelectric Eater Heaters



FAR	Flexibility Analytics and Register
FL	Federated Learning
FSP	Flexibility Service Provider
gRPC	Google Remote Procedure Call
IDS	International Data Space
IEGSA	Interoperable European Grid Services Architecture
KG	Knowledge Graph
KPI	Key Performance Indicators
LSTM	Long Short-term Memory
MILP	Mixed-Integer Linear Problem
ML	Machine Learning
MMR	Mid-market-rate
MQTT	MQ Telemetry Transport
OBD	On-board Diagnostic Device
O&M	Operation and Maintenance
P2G	Power-to-gas
P2P	Peer-to-peer
PaaS	Platform-as-a-Service
PMSG	Permanent Magnet Synchronous Generator
PoD	Point of Delivery
POOL	Pre- / Post-delivery pool
REC	Renewable Energy Community



RES	Renewable Energy Sources
RPC	Remote Procedure Call
SDR	Supply and demand ratio
SM	Smart Meters
SQL	Structured Query Language
TCN	Temporal Convolutional Networks
TL	Transfer Learning
TRL	Technology Readiness Level
TSG	TNO Security Gateway
TSO	Transmission System Operator
VAR	Vector Autoregressive
WT	Wind Turbine



Executive summary

The ENERSHARE project developed data-driven services and System-of-System Digital Twin (DT) applications across multiple value chains in WP6 by leveraging privacy-preserving federated learning. These cutting-edge services address various energy sector challenges, including load flexibility estimation and optimizing renewable energy sources (RES). Additionally, the project enables cross-sector services, such as wellness alarms, energy poverty monitoring, and electric vehicle (EV) management. System-level DTs are developed to enhance interoperability and support complex energy planning and real-time operations. This report describes the Beta version of the WP6 data-centric services software, particularly the development progress compared to D6.1, APIs and documentation, and options for integration with the Data Space (for Deliverable D6.3, Final version). The following table presents a summary of the current development status. It outlines forthcoming contributions for the different federated learning (FL) services and the pilot and corresponding use case (from Deliverable D2.1).

Federated Learning Service	New features in D6.2 (Beta version)	Future features for D6.3 (Final version)	Pilot / Use Case (D2.1)	TRL
Knowledge-driven federated learning (ENGIE)	<ul style="list-style-type: none"> - Usage of time Series with LSTM - REST API 	Connection to Data Space	Pilot 2 (Portugal) / Leveraging on consumer-level load data to improve TSO's operational and planning procedures	5
Multivariate energy time series forecasting (INESC TEC)	<ul style="list-style-type: none"> - Integrate non-linear (e.g., additive models) approaches - Validation with real wind power data - REST API developed 	<ul style="list-style-type: none"> - Integration with Data Space - Frontend 	Pilot 2 (Portugal) / Leveraging on consumer-level load data to improve TSO's operational and planning procedures	5
Federated learning framework (TNO)	<ul style="list-style-type: none"> - Validation with the real energy consumption data from Slovenia - Concept and architecture for the connection to Data Space 	<ul style="list-style-type: none"> - Demonstration and validation with data - Frontend - Running the framework on IDS connector 	Pilot 3 (Slovenia) / Optimal multi-energy vector planning - electricity vs heat	5
Federated transfer learning flexibility potential assessment (COMS)	<ul style="list-style-type: none"> - Flexibility potential ML model on actual pilot time series data - Initial federated learning experiments - Data Space integration plan 	<ul style="list-style-type: none"> - Federated transfer learning framework for model training - Federated learning integration with feature store - Data Space integration 	Pilot 3 (Slovenia) / Optimal multi-energy vector planning - electricity vs heat	5



10 data-driven energy services were improved, targeting different stakeholders (see the Figure below): local energy communities, TSOs, DSOs, multi-energy utilities, RES value chain, and consumer/prosumer. A summary of the current development state is:

Energy service	New features in D6.2 (Beta version)	Future features for D6.3 (Final version)	Pilot / Use Case	TRL
Energy community sizing with assets sharing (INESC TEC)	<ul style="list-style-type: none"> - Current formulation improvements to broaden its utility - Formulating, modeling, and testing business models - REST API 	<ul style="list-style-type: none"> - Integrate several price mechanisms within the business models - Data Space integration - Frontend 	Pilot 2 (Portugal) / Instantiation of energy communities and digital simulation of business models	6
Flexibility modeling of thermoelectric water heaters (INESC TEC)	<ul style="list-style-type: none"> - Validation of output flexibility measurement - Improvement of the service: parameters, variables cleaning, converter for load diagrams - REST API 	<ul style="list-style-type: none"> - Fine-tuning the REST API for integration with the data space - Converted into a standalone library for offline testing and analysis 	Pilot 2 (Portugal) / Instantiation of energy communities and digital simulation of business models	6
Community market pool to estimate energy price of internal transactions (INESC TEC)	<ul style="list-style-type: none"> - Formulation improvements to broaden its utility - Formulating, modelling, and testing business models - REST API 	<ul style="list-style-type: none"> - Integration with the Data Space - Inclusion of shared resources (e.g., batteries) 	Pilot 2 (Portugal) / Instantiation of energy communities and digital simulation of business models	6
Failure detection for wind turbines (TECNALIA, HINE, ENGIE)	<ul style="list-style-type: none"> - Data pre-process and features extraction - API docs - DS integration details 	<ul style="list-style-type: none"> - Failure hybrid model - Failure classifier based on explainable ML models - Refined the tool's API - Data Space integration 	Pilot 1 (Spain) / Wind farm integrated predictive maintenance and supply chain optimization	5
Substation Load forecasting tool (NESTER)	<ul style="list-style-type: none"> - Implementing the Pattern Sequence Forecasting (PSF) model - API documentation - DS integration plan 	<ul style="list-style-type: none"> - Inclusion of PSF models with different clustering algorithms - Refined the tool's API - Data Space integration 	Pilot 2 (Portugal) / Demonstrate service prosumer net-load forecasting	6
Energy usage prediction (COMS)	<ul style="list-style-type: none"> - Enhancing the model to heat forecasting - REST API - Dataspace integration plan 	<ul style="list-style-type: none"> - Evaluating the heat models on different houses and proposing changes - Data Space integration 	Pilot 3 (Slovenia) / Optimal multi-energy vector planning - electricity vs heat	5



Energy service	New features in D6.2 (Beta version)	Future features for D6.3 (Final version)	Pilot / Use Case	TRL
Service local load forecasts and estimation of electrical grid status (TNO)	<ul style="list-style-type: none"> - Visual API and MapEditor congruent with open-source geographical location tools - API docs 	<ul style="list-style-type: none"> - Visualisation API for the results - IDS Connector integration 	Pilot 3 (Slovenia) / Optimal multi-energy vector planning - electricity vs heat	6
Flexibility Analytics and Register (FAR) service (ED)	<ul style="list-style-type: none"> - Integration with COMS and TNO service - User database modification - API docs updates - User Interface - Consent management - 1st cycle of Data Space integration 	<ul style="list-style-type: none"> - Development of locational flexibility analytics - Development of a visualisation dashboard for analytics retrieved - Full cycle, e.g., actual compilation of demo data of analytics testing - Integration with COP service 	Pilot 3 (Slovenia) / Optimal multi-energy vector planning - electricity vs heat	6
Aggregation of flexibility from end users (FORTUM)	<ul style="list-style-type: none"> - Fairness and performance improvements - Potential Data Space integration plans - Adaptation to the new SVK rules 	<ul style="list-style-type: none"> - Frequency source integration - Prequalification-like end-to-end test - Data Space integration 	Pilot 6 (Sweden) / Aggregation of EV flexibility	6
Data-driven management of surplus renewable energy generation in distribution systems (RWTH)	<ul style="list-style-type: none"> - Development of the two stages ML approach - API development - Data Space integration plan 	<ul style="list-style-type: none"> - Finetune ML models with pilot data - Data Space integration 	Pilot 5 (Italy) / Flexibility provision for electricity grid with water pumps and predictive maintenance of the pumps	5

7 cross-sector data-centric services were improved to integrate data and business processes from the energy sector with other sector services such as water, transport, and finance. The following table presents a summary of the current services' development status.

Cross-sector service	New features in D6.2 (Beta version)	Future features for D6.3 (Final version)	Pilot / Use Case	TRL
Multi-energy flexibility potential	<ul style="list-style-type: none"> - Importing daily profiles from the user energy forecasting tool - REST API 	<ul style="list-style-type: none"> - Evaluating the integrated models - Data Space integration 	Pilot 3 (Slovenia) / Optimal multi-energy vector	5



Cross-sector service	New features in D6.2 (Beta version)	Future features for D6.3 (Final version)	Pilot / Use Case	TRL
assessment (COMS)	- forecasting capabilities	-Enhancing service with emissions and environmental footprint for different multi-energy mixes	planning - electricity vs heat	
Cross-operators' portal (COP) (ED)	- Integration with TNO substation load forecast service - COP user database modification - API docs updates to adapt on pilot custom needs - User Interface for registering assets - 1st cycle of DS integration	- Locational flexibility analytics - Development of visualisation dashboard for analytics retrieved - Full cycle e.g., actual compilation of demo data of analytics testing - Integration with COP service to allow cross-energy flexibility orders	Pilot 3 (Slovenia) / Optimal multi-energy vector planning - electricity vs heat	6
Emissions and ecological footprint (ENVI)	- Enhanced DNN model with much larger sample coupled with SCAM model - REST API to obtain emissions and EF - DS integration plan	- Integration with multi-energy flexibility potential service - Data Space integration	Pilot 3 (Slovenia)	5
EV charging monitoring and remote management (EMOT)	- EV real-time monitoring service implementation - In-lab service validation phase	- Prioritisation of charging and dynamic pricing aligned to the grid congestion level services implementation - Data Space integration	Pilot 5 (Italy) / Services for e-mobility CPOs, EVs drivers and DSO	5
ML-based models for assessing renovation actions in residential buildings (AI4EF) (NTUA)	- Full-stack technical solution, e.g., DB, User Interface - Data Sharing Infrastructure - API	- Dataset augmentation with generative ML - Data Space integration - Front-end, back-end refinements.	Pilot 7 (Latvia) / Cross-value chain services for energy-data driven green financing	5
Health insurance alarms for senior living alone (SEL)	- DB integration with existing backend - Frontend - Algorithm for pattern recognition - API	- Complete the development of the event detection component d - Finalize API - Data Space integration	Pilot 2 (Portugal) / Detect irregularities in energy consumption in households with seniors living alone	5



Cross-sector service	New features in D6.2 (Beta version)	Future features for D6.3 (Final version)	Pilot / Use Case	TRL
Appliances maintenance or retrofit (SEL)	<ul style="list-style-type: none"> - Validation phase with supervised classification - Internal architecture with backend service with NILM algorithms, local DB, and API for third-party data access 	<ul style="list-style-type: none"> - Final version of the NILM algorithms - Data Space integration 	Pilot 2 (Portugal) / Suggest maintenance of appliances based on NILM data	5

3 DTs have reached beta versions with substantial advancements in data warehousing, simulation scenarios, and visualisation interfaces, namely:

Digital Twin	New features in D6.2 (Beta version)	Future features for D6.3 (Final version)	Pilot / Use Case	TRL
Power-to-Gas optimal planning (NTUA)	<ul style="list-style-type: none"> - First integration of simulator with DB - Inclusion of real data and costs in the simulation - Data ingestion mechanism MVP - First prototype of front-end - API docs 	<ul style="list-style-type: none"> - Final version of data ingestion mechanism and DB - Full interconnection with T6.4 (visual analytics) - Simulation results through the data space 	Pilot 4 (Greece) / Digital Twin for optimal data-driven Power-to-Gas planning	5
Wind turbine Digital Twin (TECNALIA)	<ul style="list-style-type: none"> - Follow-up of failure diagnosis independent services - More detailed architecture description - API docs 	<ul style="list-style-type: none"> - Creation of the DT by means of integration the failure diagnosis services of gearbox, electric generator, and hydraulic pitch system in ENERSHARE Data Space 	Pilot 1 (Spain) / Wind farm integrated predictive maintenance and supply chain optimization	5
Flexibility planning in electrical networks (RWTH)	<ul style="list-style-type: none"> - Basic configurations and parametrizations of the toolset - Selection of the best candidate technologies for the implementation of a deployable solution 	<ul style="list-style-type: none"> - First Proof-of-Concept of the software solution stack - API docs - Interconnection with services from other WPs - Data ingestion and Data Space integration 	Pilot 5 (Italy) / Cross-sector Flexibility Services for aggregators and DSO (to be confirmed)	4-5

To provide all the needed functionalities that will allow the optimal presentation, exploitation, and understanding of the data produced and collected within ENERSHARE, an advanced visualisation dashboard is implemented. Moreover, a service that provides spatial data visualisation with diverse aggregation methods (e.g., kernel smoothing) is also available for environmental monitoring and urban planning data-centric services, allowing flexible and powerful insights into complex spatial information.



1 Introduction

The primary goal of the ENERSHARE project is to seamlessly integrate and tailor the Data Space paradigm within the energy sector. This involves converging data, storage, and computing infrastructures with widely used services and tools for data analysis and sharing, fostering the creation of interoperable resources. Beyond the conceptual and infrastructural framework, the project ambitiously develops data-driven services and System-of-System Digital Twin (DT) applications in WP6 ("Cross-value chain AI-based data-driven services"), spanning multiple value chains, and building upon advancements from previous and ongoing projects.

In month M18, the ENERSHARE project has made significant advances in the development of various data-driven services and DT applications across the energy sector. The beta versions of these services, spanning federated learning methods, data-driven energy services, and data-driven cross-sector services, showcase advancements and improvements.

Federated learning methods, such as knowledge-driven federated learning and federated learning frameworks, have been implemented with notable enhancements in the beta versions. These include privacy-preserving techniques, improved datasets, and the application of multivariate energy time series forecasting. Data-driven energy services cover a diverse range, including energy community sizing, flexibility modelling of thermoelectric water heaters, community market pool for internal transactions, and failure detection algorithms for wind turbine components. Data-driven cross-sector services, including multi-energy flexibility potential assessment and emissions/ecological footprint calculation, have expanded their scope and accuracy. The System-of-System DTs, such as DT for optimal Power-to-Gas planning and Digital Twin for flexible energy networks, have reached beta versions with substantial advancements in data warehousing, simulation scenarios, and visualisation interfaces. These DTs play a crucial role in optimizing renewable energy usage, improving O&M for wind turbines, and simulating flexible energy networks.

It is important to highlight that these data-centric services are the ultimate instrument to extract and showcase the value of data sharing to multiple stakeholders. Through these services, the ENERSHARE also expects to establish a solid foundation for cross-sector collaboration, underlining the pivotal role of shared data in optimizing efficiency, sustainability, and innovation across the energy landscape.



1.1 About this document

This deliverable will report about the Beta version of the federated learning (Task 6.1), data-driven services (Tasks 6.2 and 6.3), data visualisation (Task 6.4) and DTs (Task 6.5). The objective of this document is to describe the development progress in comparison to D6.1, APIs and documentation, option for integration with the Data Space (for Deliverable D6.3, Final version), as well as its level of maturity and readiness of each service. The software components described in this document completed the second technology development cycle, which means that have completed the main software functions and API (and documentation), and the next steps are the integration with the Data Space components.

1.2 Intended audience

The intended audience for this deliverable is:

- Data providers and consumers of the Data Space, willing to know the potential of the data-centric energy and cross-services, as well as DT capabilities to generate synthetic data.
- Potential end-users of the services, namely: consumers, energy communities, transmission system operator (TSO), distribution system operator (DSO), multi-energy utilities, O&M companies, RES power plants developers, non-energy service providers (i.e., mobility, healthcare, buildings).
- Data Space operators interested in creating value from data by enabling data-centric service provision.

1.3 Reading recommendations

This document is divided into 7 chapters. Chapter 1 is this introduction. Chapter 2 describes the enhancement and adaption of Federated Learning (FL), capable of considering these strong constraints when it relates to private or confidential data in the energy domain. Chapter 3 describes the last development progress on the energy services, covering the following stakeholders group: local communities, RES value chain, multi-energy utilities, TSO, DSO, end-user/consumer. Chapter 4 describes the recent developments of cross-sector services that integrate energy data with other sector data spaces (mobility, healthcare, finance, buildings, cross-sector flexibility). Chapter 5 describes the Visualisation Engine, which uses Apache Superset to serve as the basic visualisation dashboard and reporting tool, and a spatial visualisation service. Chapter 6 describes the system-level DTs covering wind turbine model,



power-to-gas optimal planning and flexibility planning in electrical networks. Finally, Chapter 7 presents final remarks, particularly the level of fulfilment of the key performance indicators (KPI) defined in the ENERSHARE objectives for WP6.



2 Federated Learning Methods

Task 6.1 aims at leveraging FL to propose innovate learning approaches that are, on the one hand, respecting privacy constraints on data and, on the other hand, tailored to specific requirements of the pilots and use cases of ENERSHARE. This section describes the development progress of the different FL approaches and their future integration with ENERSHARE’s data space. Table 1 presents a summary of the current development status and outlines forthcoming contributions.

Table 1: Overview of the different federated learning releases in WP6

FL Framework	Features in deliverable D6.1 (Alpha version)	New features in D6.2 (Beta version)	Future features for D6.3 (Final version)
Knowledge-driven federated learning (ENGIE)	Knowledge driven Semantic engine for Automated FL with differential privacy	- Usage of time Series with LSTM - REST API	- Connection to Data Space
Multivariate energy time series forecasting (INESC TEC)	- Main encryption, model training and forecasting components - Collaboration simulator (Python script)	- Integrate non-linear (e.g., additive models) approaches - Validation with real wind power data - REST API developed	- Integration with Data Space - Frontend
Federated learning framework (TNO)	- Local non-linear forecasting components - Graphical simulator of the grid with predictive capabilities on the status	- Validation with the real energy consumption data from Slovenia - Concept and architecture for the connection to Data Space	- Demonstration and validation with data - Frontend - Running the framework on IDS connector
Federated transfer learning flexibility potential assessment (COMS)	- Transfer learning GUI application concept - Initial flexibility potential assessment ML model	- Flexibility potential ML model on actual pilot time series data - Initial federated learning experiments - Data Space integration plan	- Federated transfer learning framework for model training - Federated learning integration with feature store - Data Space integration



2.1 Knowledge-driven federated learning

2.1.1 Development progress

The implementation of the knowledge-driven federated learning platform (see Figure 1) is at TRL 5. The reasoning engine and the FL engine have been implemented and tested. Moreover, a proof of concept has been developed and model trained using open data.

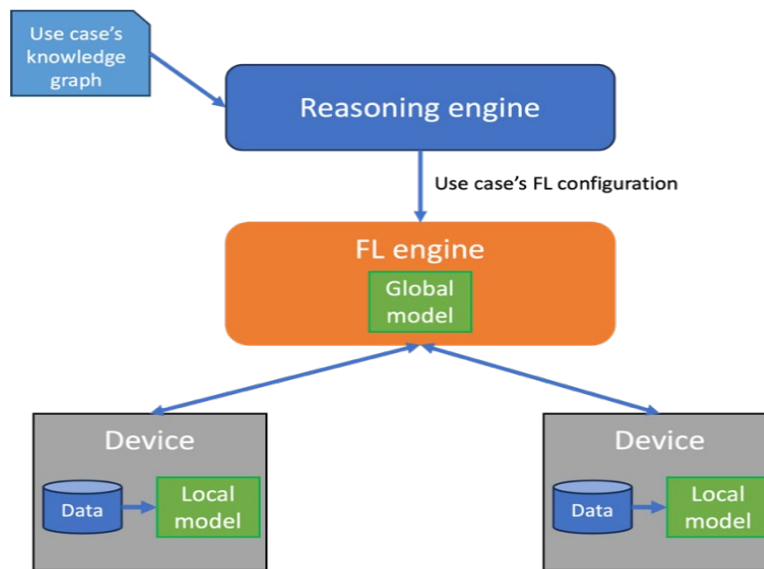


Figure 1: Architecture of knowledge-driven federated learning platform

For the beta version (deliverable D6.2), ENGIE developed a time series model using Long Short-Term Memory (LSTM) with differential privacy, achieving an MSE loss of 0.0002. Utilizing LSTM networks for wind energy forecasting proves powerful as LSTMs adeptly capture temporal dependencies in time series data. Wind energy forecasting plays a pivotal role in optimising wind farm operations and effectively integrating wind energy into the power grid.

However, using a non-interpretable model like LSTM to predict power generation of wind turbines without prior knowledge of physical and business constraints might be inefficient. To address this, accuracy improvement can be guaranteed by employing interpretable physics-based models capable of generalizing equations governing power generation for simulating various future scenarios. Challenges arise when governing equations are unknown or overly complex, and the acquired signals provide limited information about power generation. In such cases, power generation can only be modelled through a data-driven transfer function between past physical and numerical parameters. Injecting domain knowledge and physics aids in calculating relevant features, augmenting the informational content of the input dataset.



Below, we outline the modelling steps for an example of a hybrid approach combining data-driven modelling with physics-based feature engineering.

Step 1: Conducting an a priori analysis of causal pathways leading to a set of presumed causal parameters influencing power generation.

Developing a causal machine learning model necessitates exposing it only to parameters with causal relevance. Otherwise, the model might encode spurious associations rather than true causal relationships. Thus, identifying the causal structure of wind turbine power generation via a Directed Acyclic Graph (DAG) related to the input dataset is crucial.

Step 2: Training a set of regularized neural network predictors and model selection guided by sparsity based on causal invariance.

Wind turbine power generation, based on fluid and material conditions, can be represented as the sum of nonlinear functions relying exclusively on subsets of state parameters, reflecting distinct causal paths. Constraining parameters through subsets enhances interpretability and prevents non-additive interactions between inputs from different causal pathways.

We employed a Two-layer LSTM: the first LSTM layer with 128 units, followed by another LSTM layer with 64 units. To our knowledge, this marks the first instance of employing a multi-head neural network to consistently constrain interactions between input parameters in accordance with a causal model.

While we only include input parameters assumed to have a direct causal connection with power generation, the neural network might not infer the correct causal model. Measurement bias and unobserved causal paths make it unlikely for the model to converge to the true causal structure. Hence, quantifying its causal performance becomes crucial. We aim to achieve this through the concept of Invariant Causal Prediction (ICP). ICP postulates that the true causal model's parameters remain invariant under a distributional shift, even if the data's spurious correlations between features change. Our next task is to apply ICP to assess the model's causal coherence.

In addition to that, we have designed our client as flask server which receives parameters from FL server. All the communication are tested, and the results are obtained using local host server with different port numbers, considering the deployment in the data space. Once Dataspace is defined with necessary parameters, we can directly launch it.



2.1.2 Key features and documentation

Our FL service is designed to enable collaborative training of machine learning models by multiple data owners, termed as “agents”, while ensuring the privacy and control of their data.

Key Features:

1. **Data Privacy and Control:** Each agent trains models on their local data without the need to share it. This preserves data privacy and gives full control to data owners over their proprietary information.
2. **Model Weights Sharing:** Instead of sharing sensitive data, agents only exchange the weights of the trained models. This approach maintains data confidentiality while benefiting from collaborative learning.
3. **Efficient and Cost-Effective Training:** By distributing the model training across various agents, our service accelerates the training process and reduces costs. It eliminates the need for centralizing large datasets, thereby avoiding the requirement of supercomputers for processing.
4. **Framework Utilization:** We leverage the [Flower Federated Learning Framework](#), renowned for its efficiency and robustness in federated learning environments.

Our service is ideal for scenarios where data privacy is paramount, and collaborative efforts are needed to build more accurate and comprehensive machine learning models. In this context, we can find the link for the REST API [here](#).

2.1.3 End-to-end FL workflow

The sequence diagram of Figure 2 highlights the end-to-end workflow of ENGIE FL framework:

1. First, all agents must register to the FL server. They declare the communication endpoints and some metadata about their private data, which must be followed by a predefined schema on the FL server.
2. Training the Model: A client, user or an agent initiates the training of a model by sending parameters (which are like instructions or rules) to the central service (FLServiceServer). This central system then creates an identifier for the new model and return it for future references.
3. Federated Learning Process: The model is not trained in one place; instead, the training process is distributed across multiple agents (FLAgent1, FLAgent2, etc.). These agents could be different computers or servers. Each agent has its own private data it uses for training, which means the data stays secure and isn't shared with others.



4. Combining the Knowledge: Once each agent has trained the model with its data, it sends back just the learned weights (the outcomes of training) to the FLServiceServer. This server then combines all these weights to form a complete model. This is like taking lessons learned from each agent and creating a collective knowledge without sharing the data.
5. Making the Model Available: The complete model is stored back in the ModelStore, and a location for the model is created so others can access it.
6. Using the Model: Finally, when someone wants to use this trained model, they can download it using its location URL. After downloading, they can use it with their private data to make predictions or inferences on their own private data.

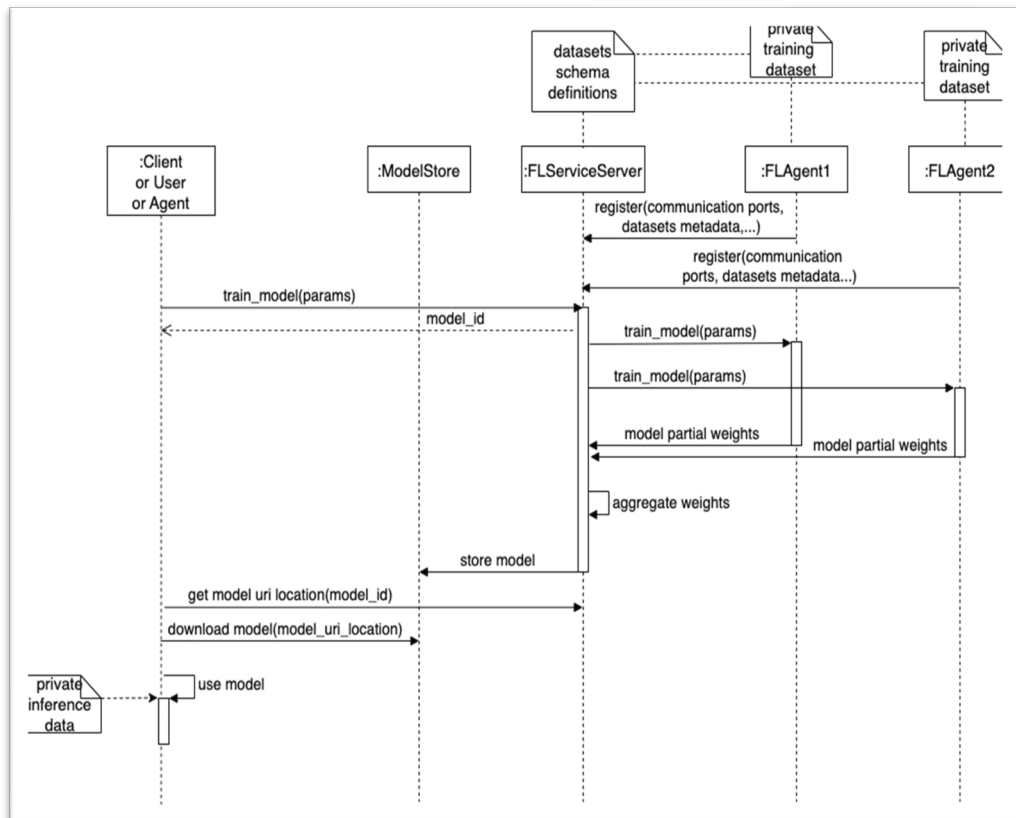


Figure 2: End-to-end workflow of the knowledge-driven FL method



2.1.3 Next steps

For future work (in D6.3), we will improve the accuracy and the performance of our approach. In addition to that, the approach will be integrated to the ENERSHARE Data Space following the diagram from Figure 3.

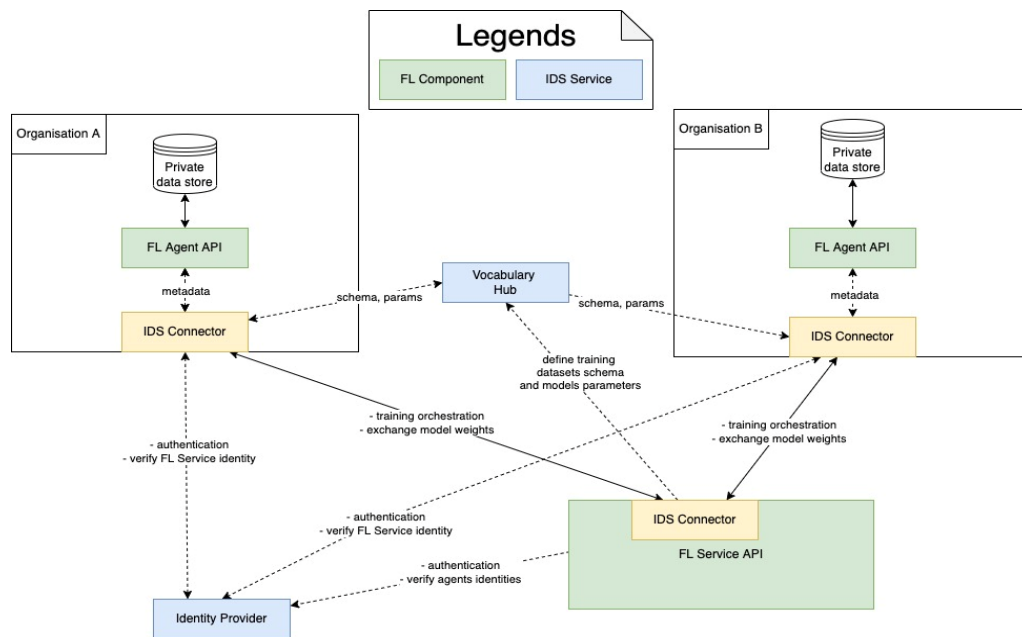


Figure 3: Integration of the knowledge-driven FL method with the Data Space

Components and Interactions:

- **Private Data:** Each participating organization, represented as Organization A and B, maintains a private data. This data must comply with schema defined in the Vocabulary Hub.
- **FL Agent API:** will use the private data in the FL learning process. It processes metadata and communicates with the FL Server through IDS Connector. It will only share metadata and trained model weights.
- **IDS Connector:** We are planning to use TNO Security Gateway (TSG) Connector, providing secure communication channels between different organisational entities in the Dataspace.



- FL Service: orchestrates the training process and manages the exchange of model weights to ensure that the actual data remains within organizational boundaries.
- Vocabulary Hub: Supplies the necessary schemas and parameters to harmonize the definition of training datasets and model parameters across different entities. This ensures that all participants are aligned in terms of data understanding and model configuration.
- Identity Provider: A centralized service that authenticates all entities involved in the federated learning process, from individual FL agents to the FL Service API. This ensures that only verified participants are involved in the data exchange and model training processes.

2.2 Federated learning framework

2.2.1 Development progress

The development of the FL platform and both local predictive and global models are on TRL 5. The FL algorithms were tested on actual data in lab environment.

The first version of the local predictive models for the energy use on the level of the individual household, which use the LSTM technique, was described in D6.1. However, the dataset used to create the first version of the local predictive model was sparse and did not allow to make the predictions more suitable to detailed energy use per day in any given household. The level of predictions was good, but not of high accuracy, because the differences according to, for example, a day per week were not possible to take into consideration (see Figure 4). The local predictive models needed improvement to consider the days of the week, which demonstrate the different patterns of the energy usage. The dataset, received from the Slovenian pilot (Pilot 3), allowed to make the predictions according to the days in a week.

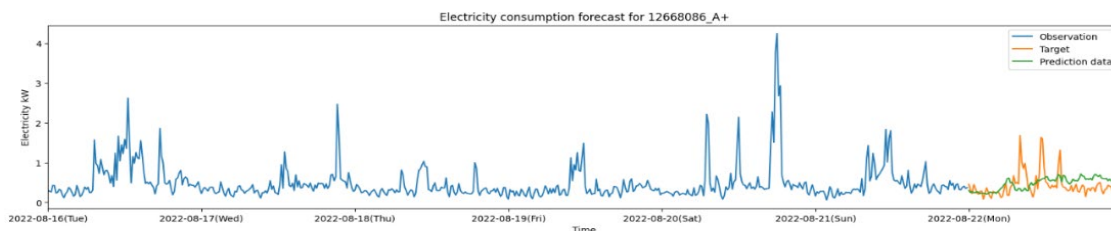


Figure 4: Predictions of the version 1 according to the target values

The dataset from Slovenian pilot comprises the energy usage from the low-density populated area in Slovenia for the period from January 1, 2021, to September 09, 2023. The dataset incorporates time series observations of the energy usage on the level of the individual



household, providing a comprehensive record of energy used for a long period of time. The dataset contains records with the aggregated values for every 15 minutes for the whole period mentioned above. The primary objective is to comprehend the received observations and to tune the local predictive LSTM model to this long-term rich dataset. The previous dataset allowed to run the model on sparser (3 months in year 2023, 4 months in year 2022) and highly aggregated data (1 record per day). This dataset allows to train the local models and validate them based on actual energy consumption from the households. We studied dataset for the inconsistencies and gaps. The conclusion is that it is a well-prepared dataset, which could be fully used for the local predictions.

The representation of three time series from the whole available historical period is shown in Figure 5.

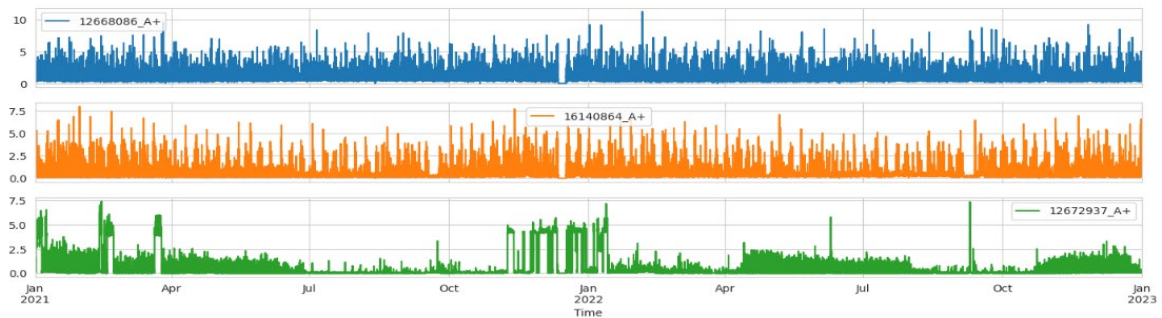


Figure 5: Sample of the dataset from the three households for the period from 01-01-2021 till 01-01-2023

We have run existing model on the provided dataset and understood that model needed to be retrained since the aggregation levels for energy consumption were too different. We needed to adjust the minimum and maximum thresholds for the energy consumption in the model according to the new dataset. Also, we needed to change batch size since the data is denser than the previous dataset. We have tuned evaluation interval from 100 to 200 since the dataset contains two years of time series.

The training of the model is improved upon the local FL client. The interface to load the model is depicted in Figure 6.



```

def fit(self, parameters, config):
    """Train the provided parameters using the locally held dataset.
    -----
    parameters : List[numpy.ndarray]
        The current (global) model parameters.
    config : Dict[str, Scalar]
        Configuration parameters which allow the
        server to influence training on the client. It can be used to
        communicate arbitrary values from the server to the client, for
        example
    Returns:
    -----
    parameters : List[numpy.ndarray]
        The locally updated model parameters.
    num_examples : int
        The number of examples used for training.
    metrics : Dict[str, Scalar]
        A dictionary mapping arbitrary string keys to values of type
        bool, bytes, float, int, or str. It can be used to communicate
        arbitrary values back to the server.
    """
    past_history = 120
    future_target = 24
    STEP = 1

    x_train_multi, y_train_multi = multivariate_data(dataset[:, :1], dataset[:, :1], 0,
                                                    TRAIN_SPLIT, past_history,
                                                    future_target, STEP)

    x_val_multi, y_val_multi = multivariate_data(dataset[:, :1], dataset[:, :1],
                                                TRAIN_SPLIT, None, past_history,
                                                future_target, STEP)

    BATCH_SIZE = 128
    train_data_multi = tf.data.Dataset.from_tensor_slices((x_train_multi, y_train_multi))
    train_data_multi = train_data_multi.cache().shuffle(BUFFER_SIZE).batch(BATCH_SIZE).repeat()

    val_data_multi = tf.data.Dataset.from_tensor_slices((x_val_multi, y_val_multi))
    val_data_multi = val_data_multi.batch(BATCH_SIZE).repeat()

```

Figure 6: The configuration of the client with all its parameters

The configuration and hyper parameters for the local LSTM-based model are shown at Table 4 and Figure 7.

Table 2: Configuration parameters for LSTM model for local predictions

Hyper parameters	Hyper parameters values
Training	Batch size = 32 Buffer size = 2000 Train split = 1500
LSTM	Evaluation interval = 200 Epochs = 25 Patience = 5



Hyper parameters:
Sequential = TRUE
Layers = 32
Optimizer = RMSprop
Loss parameter = MAE

Figure 7: New hyper parameters for LSTM model for local energy consumption predictions

The configurable parameters are extracted out of the training and operational LSTM model. The improved training process is demonstrated in Figure 8.

```

val_data_multi = tf.data.Dataset.from_tensor_slices((x_val_multi, y_val_multi))
val_data_multi = val_data_multi.batch(BATCH_SIZE).repeat()

input_timesteps=x_train_multi.shape[1]
output_timesteps= future_target
num_links= y_train_multi.shape[2]

num_inputs=x_train_multi.shape[2]

# Update local model parameters
self.model.set_weights(parameters)

# Get hyperparameters for this round
batch_size: int = config["batch_size"]
epochs: int = config["local_epochs"]
modelstart = time.time()

early_stopping = EarlyStopping(monitor='val_loss', patience = PATIENCE, restore_best_weights=True)
steps_per_epoch = 5

train_data_multi = tf.data.Dataset.from_tensor_slices((x_train_multi, y_train_multi))
val_data_multi = tf.data.Dataset.from_tensor_slices((x_val_multi, y_val_multi))
val_data_multi = val_data_multi.batch(BATCH_SIZE).repeat()
early_stopping = EarlyStopping(monitor='val_loss', patience = PATIENCE, restore_best_weights=True)

# Create a TensorBoard callback
logs = "logs/" + datetime.now().strftime("%Y%m%d-%H%M%S")

tboard_callback = tf.keras.callbacks.TensorBoard(log_dir = logs,
                                                histogram_freq = 1,
                                                profile_batch = '500,520')

history= self.model.fit(x_train_multi,
                        y_train_multi,
                        steps_per_epoch=steps_per_epoch,
                        validation_steps = 5,
                        #epochs=10,
                        epochs=4,

                        batch_size=32,
                        verbose=1,
                        validation_data=val_data_multi,
                        callbacks=[early_stopping,tboard_callback ],
                        shuffle=False)

```

Figure 8: The training process of the LSTM model according to the configuration parameters



With that additions we have added to our model the recognition of the weekdays and trained separately the models on the basis of the patterns of energy usage from the historical period of two years. We can receive now the improved predictions according to the day of a week, which is depicted in Figure 9.

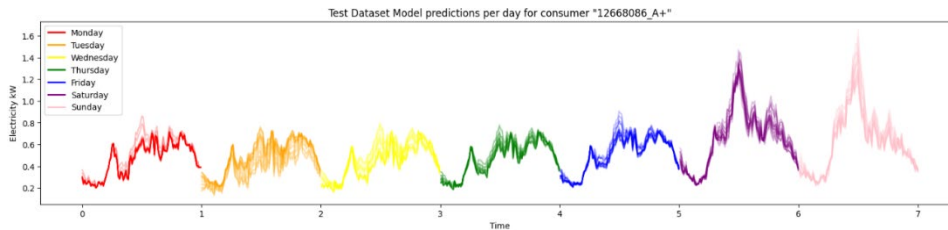


Figure 9: The predictions of the energy usage for the next 24 hours according to the day in a week for a specific household

The new predictive model was Dockerized and run on the FL platform. The new predictions were being taken by the global energy grid simulator model, and the results achieved on overloading of the local transformers (example from the global model) are shown in Figure 10 where part a) demonstrates the logs on work of local predictive clients while they are running, part b) represents the actual intermediate results from the local clients, and part c) demonstrates the result of the global model.

a) Running local model

```

Period Flower: 2022-07-12 14:04:32,469 | connection.py:36 | ChannelConnectivity_READY
DEB10 Flower: 2022-07-12 14:04:32,469 | connection.py:36 | ChannelConnectivity_READY
5/5 [=====] - 67s 6s/step - loss: 0.5951 - mae: 0.6193 - mse: 0.5951 - val_loss: 0.6702 - val_mae: 0.8156 - val_mse: 0.6702
Epoch 2/10
5/5 [=====] - 5s 198ms/step - loss: 0.4566 - mae: 0.5265 - mse: 0.4566 - val_loss: 0.5330 - val_mae: 0.7254 - val_mse: 0.5330
Epoch 3/10
5/5 [=====] - 98s 6s/step - loss: 0.5951 - mae: 0.6193 - mse: 0.5951 - val_loss: 0.6702 - val_mae: 0.8156 - val_mse: 0.6702
Epoch 2/10
5/5 [=====] - 5s 1s/step - loss: 0.4566 - mae: 0.5265 - mse: 0.4566 - val_loss: 0.5330 - val_mae: 0.7254 - val_mse: 0.5330
Epoch 3/10
5/5 [=====] - 40s 10s/step - loss: 0.4576 - mae: 0.5781 - mse: 0.4576 - val_loss: 0.6362 - val_mae: 0.7947 - val_mse: 0.6362
Epoch 4/10
5/5 [=====] - 15s 3s/step - loss: 0.4576 - mae: 0.5781 - mse: 0.4576 - val_loss: 0.6362 - val_mae: 0.7947 - val_mse: 0.6362
Epoch 4/10
5/5 [=====] - 124s 13s/step - loss: 0.5951 - mae: 0.6193 - mse: 0.5951 - val_loss: 0.6702 - val_mae: 0.8156 - val_mse: 0.6702
Epoch 2/10
5/5 [=====] - 18s 4s/step - loss: 1.6332 - mae: 1.1913 - mse: 1.6332 - val_loss: 0.7586 - val_mae: 0.8694 - val_mse: 0.7586
Epoch 5/10
5/5 [=====] - 23s 5s/step - loss: 1.6332 - mae: 1.1913 - mse: 1.6332 - val_loss: 0.7586 - val_mae: 0.8694 - val_mse: 0.7586
Epoch 5/10
5/5 [=====] - 13s 774ms/step - loss: 0.4566 - mae: 0.5265 - mse: 0.4566 - val_loss: 0.5330 - val_mae: 0.7254 - val_mse: 0.5330
Epoch 3/10
5/5 [=====] - 10s 2s/step - loss: 0.4576 - mae: 0.5781 - mse: 0.4576 - val_loss: 0.6362 - val_mae: 0.7947 - val_mse: 0.6362
Epoch 4/10
5/5 [=====] - 4s 879ms/step - loss: 1.6332 - mae: 1.1913 - mse: 1.6332 - val_loss: 0.7586 - val_mae: 0.8694 - val_mse: 0.7586
Epoch 5/10
5/5 [=====] - 2s 467ms/step - loss: 1.2734 - mae: 1.0826 - mse: 1.2734 - val_loss: 0.7520 - val_mae: 0.8649 - val_mse: 0.7520
Epoch 6/10
5/5 [=====] - 156s 5s/step - loss: 0.5951 - mae: 0.6193 - mse: 0.5951 - val_loss: 0.6702 - val_mae: 0.8156 - val_mse: 0.6702
Epoch 2/10
5/5 [=====] - 21s 5s/step - loss: 1.2734 - mae: 1.0826 - mse: 1.2734 - val_loss: 0.7520 - val_mae: 0.8649 - val_mse: 0.7520
Epoch 6/10
5/5 [=====] - 4s 829ms/step - loss: 1.0039 - mae: 0.9228 - mse: 1.0039 - val_loss: 0.8247 - val_mae: 0.9077 - val_mse: 0.8247
Epoch 7/10
5/5 [=====] - 22s 5s/step - loss: 1.2734 - mae: 1.0826 - mse: 1.2734 - val_loss: 0.7520 - val_mae: 0.8649 - val_mse: 0.7520
Epoch 6/10
5/5 [=====] - 3s 686ms/step - loss: 0.4566 - mae: 0.5265 - mse: 0.4566 - val_loss: 0.5330 - val_mae: 0.7254 - val_mse: 0.5330
    
```



ENERSHARE has received funding from [European Union's Horizon Europe Research and Innovation programme](#) under the Grant Agreement No 101069831

b) intermediate results from running local clients

```
-- RESULTS {'loss': 0.3068103790283203, 'mae': 0.4440426528453827, 'val_loss': 0.3511671721935272, 'val_mse': 0.35116714239120483, 'val_mae': 0.5862153768539429}
-- RESULTS {'loss': 0.3068103790283203, 'mae': 0.4440426528453827, 'val_loss': 0.3511671721935272, 'val_mse': 0.35116714239120483, 'val_mae': 0.5862153768539429}
```

c) Results from global running model

```
Sending to DSO Agent...
DSO: Done processing DFs
...Execution Finished.
Writing to Influxdb
Asking DSO Agent to perform calculation
Transformer is overloaded
```

Figure 10: The intermediate and final results from the global model based on the new version of the predictive local models.

Anonymization technique development

In parallel with the improvement of the local predictive model, SoTA analysis was performed for finding the suitable anonymization techniques to use to the merged view from low-density populated area before it can come to DSO. Since the DSO does not have explicit consent from the household owners to receive their measurements, before the predictions go to the DSO, the anonymization of provided data must be guaranteed.

The overview of the existing privacy-enhancing techniques was made. The comparison for the suitability of the different solutions was performed on the base of the publications over the use-cases from energy, smart industry, and healthy living domains. The short list included:

- Differential privacy
- Data masking (as one of the data obfuscation methods)
- Zero-knowledge proofs.

Data masking can provide good results. But it requires previous knowledge on data. However, it is not possible in the case of the energy use from the specific household, since the set of the actual devices differs from house to house.

Zero-knowledge proofs can provide quite a high level of the anonymization and security. However, it does provide quite an overhead on every byte of data. It increases the data being sent to another party around 5 times. Since FL is appreciated by being light-weight technique, which does not require a stable high frequency link between the different parties, the use of such technique would influence the overall performance very negatively.



In respect with these techniques, differential privacy brings good solution without big extra overhead on the performance of the predictions in the distributed fashion. **Differential privacy (DP)** is an approach for providing privacy while sharing information about a group of individuals, by describing the patterns within the group while withholding information about specific individuals. This is done by making arbitrary small changes to individual data that do not change the statistics of interest. Thus, the data cannot be used to infer much about any individual.

There are different methods from differential privacy which can be applied. We have chosen data swapping. By swapping data nobody can trace anymore from which household the predictions are coming. However, we learned from the energy experts that the location of the household influences the predictions. The further a household is from the transformer, the more energy is needed to be supplied. Therefore, we need to make a special variant of the data swapping.

Figure 11 shows that the whole low-density population area is divided into geographical zones: close to the transformer, far, further, etc. Within every zone, the swapping will be made between the household belonging to that zone.

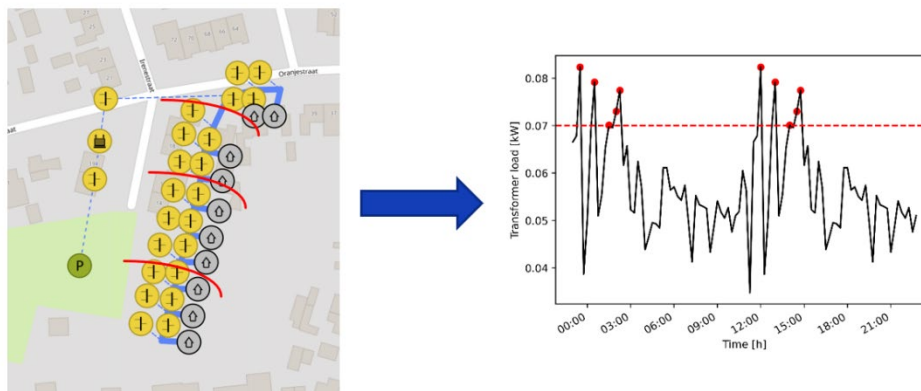


Figure 11: Separation of the low-density population area into the zones within which the data swap can be performed

Applying FL solution for sending only the results of the local predictions instead of the raw data from smart meters in household and swapping data between the households, belonging to the same defined subzone, allows to achieve the required level of privacy for the people living in the households. A global model is able to run and still provide the required predictions on overloading of the transformer on the basis of the anonymized predictions.

The current status for anonymization technique implementation: SotA analysis for the suitable anonymization techniques is performed, the decision is taken on which specific anonymization technique to implement – the differential privacy swapping method was chosen, the implementation details (including introduction of the subzones for swapping data between the households) and needed parameters are defined. The team starts to work on the implementation.

2.2.2 Documentation

Documentation of APIs includes the FL platform, FL local predictive model, FL global model.

FL Platform documentation

The platform is dockerized, therefore the platform is being started from the Docker commands (see Figure 12).

Commands for docker

Build container

```
docker build . -t docker-image-nametag
```

To run Docker container

```
docker run --rm -ti docker-image-nametag sh
```

Run FL framework inside the docker

```
poetry run ./cmd.sh
```

Figure 12: Docker commands to start the FL platform

The Federated Learning plan execution could be submitted to the FL server within the platform. It is called FL strategy. For the predictions of the electrical transformer overload, a special FL strategy is used which is presented in Figure 13.





```
# Create strategy
strategy = fl.server.strategy.FedAvg(
    fraction_fit=0.1, #Fraction of clients used during training. Defaults to 0.1
    fraction_eval=0.1, #Fraction of clients used during validation. Defaults to 0.1
    min_fit_clients=5, #Minimum number of clients used during training
    min_eval_clients=5, # Minimum number of clients used during validation
    min_available_clients=5, #Minimum number of total clients in the system
    eval_fn=get_eval_fn(model), #Optional function used for validation
    on_fit_config_fn=fit_config, #Function used to configure training
    on_evaluate_config_fn=evaluate_config, #Function used to configure validation
    accept_failures=True, #Whether or not accept rounds containing failures. Defaults to True
    initial_parameters=fl.common.weights_to_parameters(model.get_weights()) #Initial global model parameters ..look at it
)

#
# Start Flower server for four rounds of federated learning
log.debug("-- Starting Server ")
#fl.server.start_server("0.0.0.0:50053", config={"num_rounds": 5}, strategy=strategy)
# Start Flower server
fl.server.start_server(
    server_address="0.0.0.0:8080",
    config={"num_rounds": 5}, strategy=strategy,
)
```

Figure 13: FL strategy for the transformer overloading predictions with the configurable values for every FL parameter

To use other local and global models, the users can generate their own strategy plan, and to submit it into the running FL platform.

Local predictive model documentation

The input to the local predictive model is a timeseries dataset which is loaded to the local model to start its operations on available data. Data input format is discussed in deliverable D6.1.

The dataset is loaded from the provided location, and it is being converted to the smaller batches – dataframes (see Figure 14).





```
# Make TensorFlow logs less verbose
os.environ["TF_CPP_MIN_LOG_LEVEL"] = "3"
###=====>new data
dfs = pd.read_excel('TP_Podkraj_0bu_enershare_A+_15min_consumers.xlsx')
#remove first row
dfs = dfs.iloc[1:]

dfs['Time'] = dfs['Time'].astype('datetime64[s]')

#convert to numeric/float
cols = dfs.columns.drop('Time')

dfs[cols] = dfs[cols].apply(pd.to_numeric, errors='coerce')

dfs2=dfs.copy()
dfs2.head()

dfs2.set_index('Time', inplace=True)
dfs2.info()
```

Figure 14: FL parameter loading and converting dataset to framework dataframe

After the dataset is converted to several dataframes, the time is converted to the time format with which FL platform works, the target window size is chosen, and the lookback steps are defined (see Figure 15).

```
def multivariate_data(dataset, target, start_index, end_index, history_size,
                    target_size, step, single_step=False):
    data = []
    labels = []

    start_index = start_index + history_size
    if end_index is None:
        end_index = len(dataset) - target_size

    for i in range(start_index, end_index):
        indices = range(i-history_size, i, step)
        data.append(dataset[indices])

        if single_step:
            labels.append(target[i+target_size])
        else:
            labels.append(target[i:i+target_size])

    return np.array(data), np.array(labels)
```



Figure 15: Converting the data to the time series format, providing the target window and the lookback and the steps

The local client loads the model and initiates the calculations what is demonstrated in Figure 16.

```
# Define Flower client
class CifarClient(fl.client.NumPyClient):
    def __init__(self, model, x_train_multi, y_train_multi, x_val_multi, y_val_multi):
        log.debug("----- ")
        log.debug("-- Initializing Client ")
        self.model = model
        self.x_train_multi, self.y_train_multi = x_train_multi, y_train_multi
        self.x_val_multi, self.y_val_multi = x_val_multi, y_val_multi
        self.yhat= None

    def get_parameters(self):
        """Return the current local model parameters as a list of NumPy ndarrays.
        """
        raise Exception("Not implemented (server-side parameter initialization)")
```

Figure 16: The initialization of the local client with the model and dataframe

The results from local clients is presented in Figure 17.

```
# Return updated model parameters and results
parameters_prime = self.model.get_weights()
num_examples_train = len(self.x_train_multi)

results = {
    "loss": history.history["loss"][0],
    "mae": history.history["mae"][0],
    "val_loss": history.history["val_loss"][0],
    "val_mse": history.history["val_mse"][0],
    "val_mae": history.history["val_mae"][0],
}
log.debug("-- RESULTS {} ".format(results))
return parameters_prime, num_examples_train, results
```

Figure 17: Configuration of the returned results from local clients



Global model – energy grid simulator

In the current implementation, the global model reads the input files (merged local model predictions) and the topology directly from the file system. Upon global model initialization, the federated learning local model forecasts that are previously merged in the script, are read from the file system. Upon simulation initialization (once the simulation to define the status of the electricity transformer is started) the electrical grid topology file is also read from the file system (see Figure 18).

```
def __init__(self):
    self.fl_consumer_loads = self.read_fl_load_forecasts()

def simulate(self):
    print('Initializing grid simulation parameters...')
    self.init_simulation()
    print('Initializing InfluxDB...')
    self.init_influxdb()

def init_simulation(self):
    main_resource_filename = 'Slovenian_Topology.esdl'
    with(open('../ESDL_Files_GridSimulator/Slovenian_Topology.esdl', 'r')) as f:
        main_resource_contents = quote(f.read())
```

Figure 18: Initialization of the simulation

In the final release (deliverable D6.3), a REST API will be developed to retrieve these files using, so that the files do not have to be stored on the file system.

After performing OpenDSS calculations, the global model (the grid simulator) stores the grid state analysis results in InfluxDB. For that purpose, an InfluxDB client is initialized upon simulation run (see Figure 19).

```
def simulate(self):
    print('Initializing grid simulation parameters...')
    self.init_simulation()
    print('Initializing InfluxDB...')
    self.init_influxdb()
```



```
# Defining the Influx DataBase
def init_influxdb(self, host, port, dbName, measurementName):
    self.host = host
    self.port = port
    self.dbName = dbName
    self.measurementName = measurementName
    self.client = InfluxDBClient(host=self.host, port=self.port)
    self.client.create_database(self.dbName) # Create the DB dbName
    self.client.switch_database(self.dbName) # Tell to the client that we are going to work with this DB

    # Create a empty datapoint array
    self.datapoints = []
    self.batch_size = 0
```

Figure 19: Loading and initializing the client for InfluxDB to write the final results from the simulation

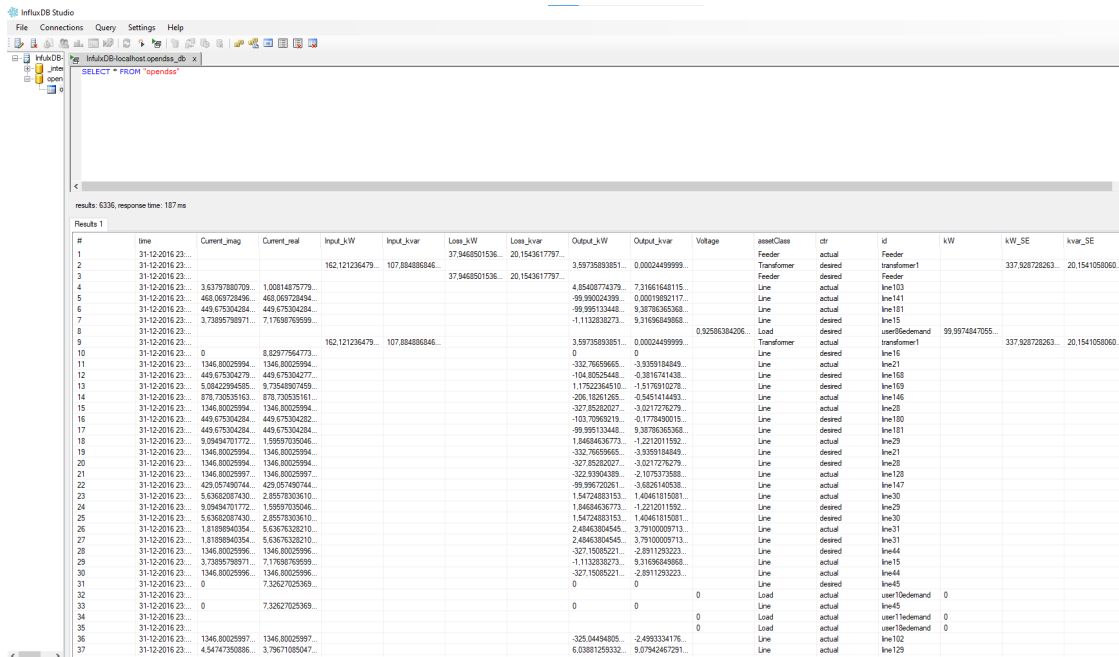
For every simulation step run, InfluxDB client writes the grid state analysis results to InfluxDB. The grid state analysis results are temporarily stored in a dataframe (datapoints) and written to the database by the client (see Figure 20).

```
270         }
271     }
272     dp_lines.append(dp_lines_u)
273
274     # Return the simulation data
275     self.datapoints.append(dp_pq_losses)
276     self.datapoints.append(dp_pq)
277     [self.datapoints.append(d) for d in dp_lines]
278     [self.datapoints.append(d) for d in dp_pv]
279
280     if len(self.datapoints) >= self.batch_size:
281         print('Writing to influxdb...')
282         temp = self.datapoints
283         self.datapoints = []
284         self.client.write_points(temp)
285
286     # Finish the execution
287     print('\n')
288     print('...Execution Finished.')
```

Figure 20: Writing process to InfluxDB

Figure 21 shows a part of the results stored in InfluxDB. For every energy asset in the topology (household, line, transformer, bus), grid state parameters are calculated by OpenDSS and stored in the database, along with the timestamp for which they are calculated.





#	time	Current_mag	Current_level	Input_kW	Input_kvar	Loss_kW	Loss_kvar	Output_kW	Output_kvar	Voltage	assetClass	ctr	id	kW	kW_SE	kvar_SE
1	31-12-2016 23:...										Feeder	actual	Feeder			
2	31-12-2016 23:...			162.121256479...	107.884886846...	37.9468501536...	20.1543617797...	3.59735893851...	0.00024499999...		desired	transformer1		337.928728263...	20.1541058060...	
3	31-12-2016 23:...	3.63797880709...	1.00814875779...					4.85408774378...	7.31661648115...		actual	Line	line102			
4	31-12-2016 23:...	468.069728496...	468.069728494...					99.990024399...	0.00019952117...		actual	Line	line141			
5	31-12-2016 23:...	449.675304284...	449.675304284...					99.995133448...	9.38786365368...		actual	Line	line181			
6	31-12-2016 23:...	3.78987789971...	7.17828789989...					1.1133382373...	9.31666468666...		desired	Line	line15			
7	31-12-2016 23:...									0.92586384206...	desired	userDemand	userDemand	99.9974847055...		
8	31-12-2016 23:...			162.121256479...	107.884886846...			3.59735893851...	0.00024499999...		actual	transformer1		337.928728263...	20.1541058060...	
9	31-12-2016 23:...	0	8.82977564773...					0	0		desired	line16				
10	31-12-2016 23:...	1346.80025994...	1346.80025994...					332.76659665...	-3.9359184849...		actual	Line	line21			
11	31-12-2016 23:...	449.675304279...	449.675304277...					-104.80525448...	-0.3816741438...		desired	Line	line168			
12	31-12-2016 23:...	5.0842298685...	9.7548907849...					1.17822345410...	1.5170301278...		actual	Line	line169			
13	31-12-2016 23:...	878.730535163...	878.730535161...					-206.18281265...	-0.5451414493...		actual	Line	line146			
14	31-12-2016 23:...	1346.80025994...	1346.80025994...					-327.85282027...	-3.0217276279...		actual	Line	line29			
15	31-12-2016 23:...	449.675304284...	449.675304282...					103.70982919...	-0.1776490015...		desired	Line	line100			
16	31-12-2016 23:...	449.675304284...	449.675304284...					-99.995133448...	9.38786365368...		desired	Line	line181			
17	31-12-2016 23:...	9.09494701772...	1.95997039046...					1.8484636773...	-1.2212011592...		actual	Line	line29			
18	31-12-2016 23:...	1346.80025994...	1346.80025994...					332.76659665...	-3.9359184849...		desired	Line	line21			
19	31-12-2016 23:...	1346.80025994...	1346.80025994...					-327.85282027...	-3.0217276279...		desired	Line	line29			
20	31-12-2016 23:...	1346.80025997...	1346.80025997...					322.93904399...	-2.1075373588...		actual	Line	line128			
21	31-12-2016 23:...	429.057490744...	429.057490744...					-99.99520261...	-3.8820146538...		actual	Line	line147			
22	31-12-2016 23:...	5.8362087430...	2.89578303810...					1.8472485193...	1.40461815091...		actual	Line	line30			
23	31-12-2016 23:...	9.09494701772...	1.95997035046...					1.8484636773...	-1.2212011592...		desired	Line	line29			
24	31-12-2016 23:...	5.8362087430...	2.89578303810...					1.5472485193...	1.40461815091...		Line	desired	line30			
25	31-12-2016 23:...	1.8189840354...	5.83676328210...					2.48463804545...	3.79100009713...		actual	Line	line31			
26	31-12-2016 23:...	1.8189840354...	5.83676328210...					1.5472485193...	1.40461815091...		Line	desired	line30			
27	31-12-2016 23:...	1346.80025996...	1346.80025996...					-327.15005221...	-2.8911293223...		Line	desired	line44			
28	31-12-2016 23:...	3.78987789971...	7.17828789989...					1.1133382373...	9.31666468666...		Line	actual	line15			
29	31-12-2016 23:...	1346.80025996...	1346.80025996...					-327.15005221...	-2.8911293223...		Line	actual	line44			
30	31-12-2016 23:...	0	7.2827025369...					0	0		Line	desired	line45			
31	31-12-2016 23:...	0	7.2827025369...					0	0	0	Load	actual	user1demand	0		
32	31-12-2016 23:...							0	0	0	Load	actual	user1demand	0		
33	31-12-2016 23:...							0	0	0	Load	actual	user1demand	0		
34	31-12-2016 23:...	1346.80025997...	1346.80025997...					-325.04494805...	-2.493334176...		Line	actual	line102			
35	31-12-2016 23:...	4.54747350886...	3.79671895047...					6.0388128332...	9.07942467291...		Line	actual	line129			

Figure 21: Screenshot of the table of InfluxDB with the results from the run of the simulation of the electrical grid

2.2.3 Examples of running services on pilot 3 data

The final results demonstrated that the model with the tuned local parameters is predicting the energy consumption with the very small loss: MAE value of the loss is 0,006.

Table 3: The results for the local predictions of 5 households after 5 training runs with the relevant MAE and the last improvement of the model as the next step will be normalization of the input data to decrease the loss (MAE value) to minimum.

Step	loss	MAE	MSE	Execution time
1	0.0066	0.0547	0.0090	2 sec
2	0.0066	0.0546	0.0066	1 sec
3	0.0065	0.0545	0.0065	2 sec
4	0.0074	0.0613	0.0091	2 sec



The last step to develop the local predictive model will be the normalization of the input data to decrease the loss (MAE value) to minimum.

The improved model is capable of providing the local predictions with the very small loss MAE (value 0,006), what means improved accuracy of the model. The global model represented by the grid simulator, developed for Deliverable 6.1 did not need the tuning. It is still able to provide the information whether the transformer will be overloaded or not in the coming 48 hours (see Figure 22).

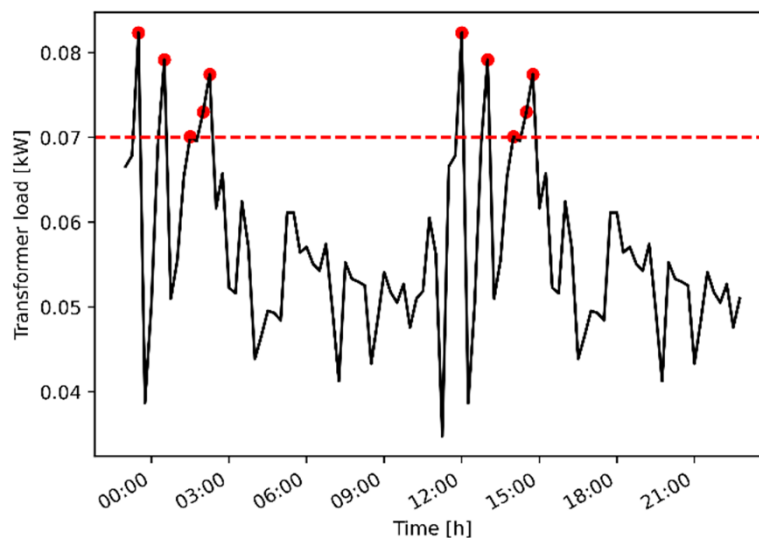


Figure 22: Estimated overloads of the electrical transformers of the global model running on the new version of the LSTM model.

2.2.4 Integration with the Data Space

An integration of the service with the Data Spaces happens by starting the FL platform from the Data Space Connector. The TNO TSG Connector with its renewed protocol will be used as a standardised basis interface to start the FL platform and to facilitate data exchange.

Figure 23 The Data Space connector from the electricity service provider is used to trigger the Federated Learning platform and the elaborated local and global models, and the Data Space connector on DSO side is used as the information sink for the resulting predictions on overloading of the electrical grid transformer (refer to Figure 23). Our integration methodology entails the implementation of the request/response API on the TSG Connector side and an API on FL platform side in such a manner that the FL platform and both services for local predictions and global energy grid simulator model could be started from the TSG connector. TSG



Connector starts the FL platform through own Federated Learning API. After that the local models and the global models are initialized and started. After successful start of the FL framework the platform API sends the Acknowledgement response to the DS connector with the information that everything is started. When the local models are done with their task on prediction at local clients, the results are being sent to merge and to anonymize before the merged view is being sent to the FL global model. Global model receives the merged anonymized predictions and topology of the grid and performs the task of simulation which results in the conclusion whether an electrical transformer will be overloaded. The results are sent to the DSO Data Space connector. Specific data sink is used to keep the results on the chance of the overloading of the electrical transformer. TSG connector is used not only to run the service, but also to ensure its congruence with the prescribed data formats and communication protocols mandated by the Federated Learning platform. The communication between the Connector and FL platform is made by gRPC call, since both solutions support this protocol for the communication.

The workflow is the following (also depicted in Figure 23):

- From the Connector side of the organization 1 the trigger to use the FL platform is given.
- Connector FL controller starts the FL platform.
- Signal to start the local clients is given to the platform by the Connector.
- Platform starts the local clients according to the configuration, provided by Connector controller.
- Signal is given to platform to start global model by Connector.
- FL platform starts global model.
- Connector of the organization 2 is informed that it should wait for the results.
- FL platform provides the results to the Connector of the organization 2.



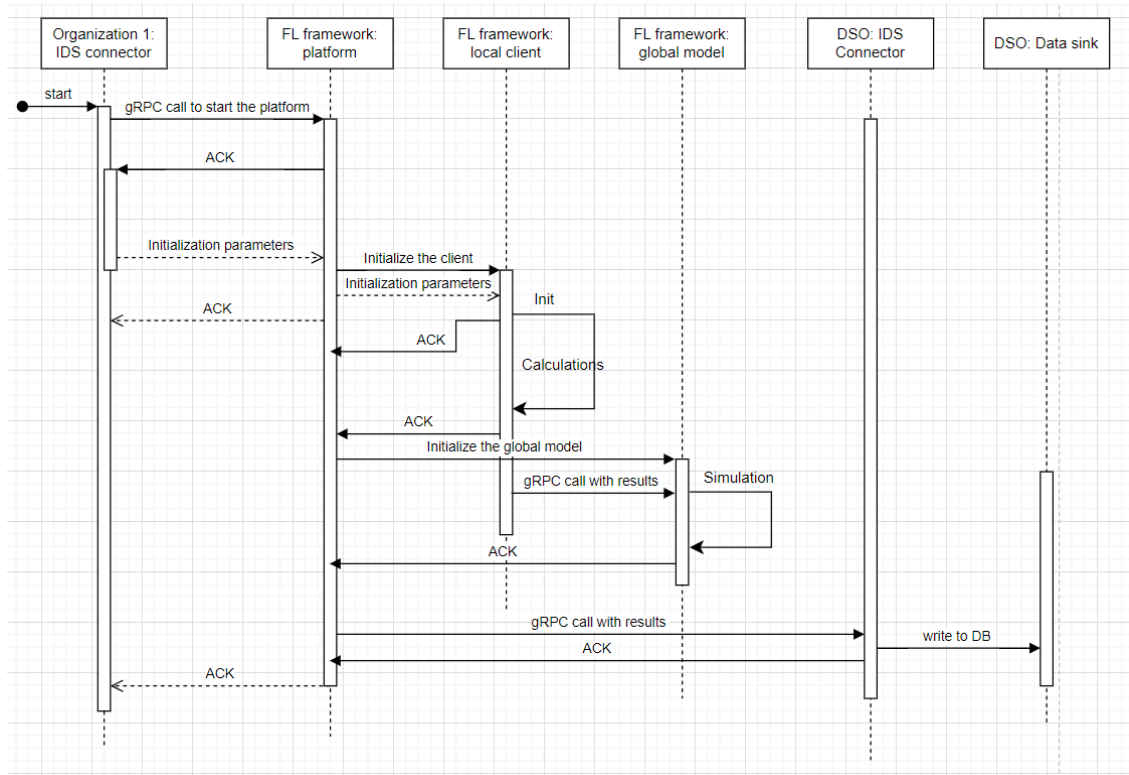


Figure 23: Sequence diagram of the integration of the Federated Learning platform and IDS connectors from different organizations.

2.2.5 Next steps

The next step for the model is to normalise the input to train the LSTM model on such an input to minimise the loss on predictions accuracy. The anonymization technique should be implemented which takes the input from all participating households, merges them into one list and anonymises the aggregated predictions by using Differential Privacy technique on partial swapping of data records. The user-friendly API to the predictive local models and a global model will be added. An integration of the Federated Learning platform with the IDS connector will be implemented as described in the subsection 2.2.4.

2.3 Multivariate energy time series forecasting

2.3.1 Development progress

The proposed methodology merges time-series data from multiple data owners through vertical federated learning. Unlike horizontal federated learning, where agents observe



identical features, enabling them to estimate a local model and subsequently share the coefficients of these features to construct the final model, vertical FL involves data owners observing distinct features. This implies that features from other agents remain unknown, preventing the assignment of coefficients. This disparity poses numerous challenges, requiring the exchange of large quantities of information.

Previously, our privacy-preserving multivariate timeseries forecasting approach was implemented and tested through a series of Python scripts. These scripts handled three core functionalities: encrypting data, fitting models, and computing forecasts. The ENERSHARE project intends to implement an API, detailed in Deliverable D6.1, outlining the core functionalities, input/output formats, and specifics. It will be coupled with a RESTful API that enables communication and data exchange (but for the sake of simplicity, since the operation of the FL service is tightly dependent on communication capabilities, this pair will be referred to as the API).

The development of this API was carried out during the last months, and expected technical challenges inherent to a complex system such as vertical FL were encountered. These hurdles encompass the design, processing capabilities, and memory requirements across the three distinct stages of forecasting: data encryption, model estimation, and forecast generation.

The primary goal of the API is to enable a data owner's contribution to a collaborative effort, emphasizing the iterative nature that demands API instances to interact with others. This collaborative aspect needs to be integrated into the solution's endpoints. Incorporating such interaction is crucial within the proposed methodology. However, the encrypted data exchange between participants presents communication capacities and infrastructure challenges despite maintaining data security through encryption.

An additional step in the process involves preparing data owners to participate in and coordinating configurations before the data encryption task. This phase entails identifying participants and setting up communication channels. Initially, the implementation overlooked Data Spaces, leading to the realization that a central node or server was necessary to facilitate this discovery phase. Participants can enlist with this server, allowing it to configure the initial settings essential for the service to operate. Consequently, a dedicated API was developed to serve this purpose (again, associated with a RESTful API). This service is streamlined, merely storing basic participant metadata and configuration data. It conducts initial data processing to determine client-side operations and manages tasks such as signalling participants and initiating the subsequent steps.

So far, an initial version of the implementations of both APIs (**FL Client API** and **Central FL Server API**) has enabled the execution of functional tests and validation of the service's expected



outcomes. A simple example, simulating a collaboration between two peers, will be presented in the subsequent subsections to illustrate the operation of the service in its current stage. The initial stage of the process was already presented in D9.3 and was recovered here to give a full demonstration of the flow of communication. Finally, a summary of the data used for the functional tests and a demonstration of the results obtained so far is presented.

Regarding the service's TRL, it has undergone substantial development leading to this initial version of the APIs, through which it was possible to perform tests on relevant scenarios (using real data) and simulate a real collaboration scenario where the multiple agents' data is kept private and only shared in its encrypted form. As mentioned previously, the tests performed have the goal of comparing the similarity between the output of local simulations (in the form of a Python script) and the output of the API as a criterion for validating the implementation. While this solution will surely be subject to future iterations, with other necessary developments that are listed in this section and are currently under way, the current state of the service should mark a transition from its previously reported TRL 4 to TRL 5.

2.3.1.1 Initialization

Before initiating the three main processes, the server API must recognize the instances of the client API linked to participants willing to engage in the forecasting service. Each of the peers should register with the server, making themselves visible. This registration enables the distribution of all subsequent tasks. Figure 24 illustrates such registration through log messages from a simple script.

```
INFO | api_server.routes.register_client:22 - Received registration data: {'client_port': '5001', 'api_version': '0.0.1'}  
- "POST /client/register HTTP/1.1" 201 Created  
INFO | api_server.routes.register_client:22 - Received registration data: {'client_port': '5002', 'api_version': '0.0.1'}  
- "POST /client/register HTTP/1.1" 201 Created
```

Figure 24: Log messages of the participants' registration

After this step, all peers are connected to the server and gain mutual awareness. Subsequently, a simple script on the server initiates the first task: collaborative data encryption. Figure 25 shows the log messages generated during the execution of the script.



```
INFO | __main__:<module>:15 - Checking for available clients
INFO | __main__:<module>:23 - Available clients: ['http://127.0.0.1:5001', 'http://127.0.0.1:5002']
INFO | __main__:<module>:24 - Signaling start of encryption
INFO | __main__:<module>:47 - Get available timestamps from each client
INFO | __main__:<module>:58 - Gathered timestamp information from all available peers
INFO | __main__:<module>:60 - Send encryption metadata
INFO | __main__:<module>:75 - Encryption metadata received by all peers
INFO | __main__:<module>:86 - Start encryption cycle
```

Figure 25: Log messages of the steps taken to perform collaborative data encryption

The server sends a GET request to check which registered peers are available. Upon receiving the request, both clients confirm their readiness. Next, the server proceeds to gather the initial data for encryption, specifically the timestamps that each client's dataset has available, to check for common timestamps. As data encryption happens sequentially, the server is also responsible for establishing a peer order and sharing it. This is fundamental to the encryption task, and it will underly all stages of the forecasting process. This order is shared with all peers, marking the completion of the initialization segment.

2.3.1.2 Encryption

The server sends a POST request to the first client to kickstart the encryption method. In the log messages for the clients, the first client (see Figure 26) is the first in the chain, which is confirmed by the log message stating that it is retrieving anonymized data from the other peer. The second participant is computing anonymized data and sending it to the former, as shown in the log messages (see Figure 27). After the encryption step is performed, the first client in the chain forwards partially encrypted data to the second peer in the chain, which performs the same operation. In this chain comprising two peers, the second peer notifies the first of the



encryption cycle and shares the resulting encrypted data. This concludes the encryption process.

```
- "GET /is-alive HTTP/1.1" 200 OK
INFO | api_client.routes.encryption:start_encryption:50 - Running encryption start with initial metadata method [POST]
- "POST /encryption/start HTTP/1.1" 200 OK
INFO | api_client.routes.encryption:get_available_indexes:73 - Running encryption available timestamps method [GET]
INFO | api_client.routes.encryption:get_available_indexes:95 - Loading datasets
INFO | api_client.routes.encryption:get_available_indexes:100 - Saving information about available timestamps
- "GET /encryption/available-timestamps HTTP/1.1" 200 OK
INFO | api_client.routes.encryption:post_encryption_metadata:122 - Running encryption metadata method [POST]
INFO | api_client.routes.encryption:post_encryption_metadata:133 - Received encryption metadata
- "POST /encryption/metadata HTTP/1.1" 200 OK
INFO | api_client.routes.encryption:post_encryption_cycle:149 - Running encryption cycle endpoint [POST]
INFO | api_client.routes.encryption:post_encryption_cycle:180 - Performing local anonymization of data
INFO | api_client.routes.encryption:post_encryption_cycle:193 - Retrieving anonymized data from other peers
INFO | api_client.routes.encryption:post_encryption_cycle:201 - Start encryption
INFO | api_client.routes.encryption:post_encryption_cycle:207 - Performing encryption step on all anonymized data
INFO | api_client.routes.encryption:post_encryption_cycle:214 - Sending partially encrypted data to next peer
- "POST /encryption/cycle HTTP/1.1" 201 Created
INFO | api_client.routes.encryption:close_encryption_cycle:320 - Running encryption cycle finish method [POST]
INFO | api_client.routes.encryption:close_encryption_cycle:328 - Closing encryption cycle. Settings and data stored.
- "POST /encryption/close HTTP/1.1" 201 Created
```

Figure 26: Log messages of the first peer in the chain interacting with the server API and with the other peer

```
- "GET /is-alive HTTP/1.1" 200 OK
INFO | api_client.routes.encryption:start_encryption:50 - Running encryption start with initial metadata method [POST]
- "POST /encryption/start HTTP/1.1" 200 OK
INFO | api_client.routes.encryption:get_available_indexes:73 - Running encryption available timestamps method [GET]
INFO | api_client.routes.encryption:get_available_indexes:95 - Loading datasets
INFO | api_client.routes.encryption:get_available_indexes:100 - Saving information about available timestamps
- "GET /encryption/available-timestamps HTTP/1.1" 200 OK
INFO | api_client.routes.encryption:post_encryption_metadata:122 - Running encryption metadata method [POST]
INFO | api_client.routes.encryption:post_encryption_metadata:133 - Received encryption metadata
- "POST /encryption/metadata HTTP/1.1" 200 OK
INFO | api_client.routes.encryption:get_anonymized_data:271 - Running encryption anonymized data retrieval method [GET]
- "GET /encryption/anonymized-data HTTP/1.1" 200 OK
INFO | api_client.routes.encryption:post_encryption_cycle:149 - Running encryption cycle endpoint [POST]
INFO | api_client.routes.encryption:post_encryption_cycle:229 - Performing encryption step on all anonymized data
INFO | api_client.routes.encryption:post_encryption_cycle:236 - Last encryption step reached. Closing cycle and sending data to all peers.
- "POST /encryption/cycle HTTP/1.1" 201 Created
```

Figure 27: The second peer in the chain interacting with the server API and receiving request from the first peer in the chain. In the end, closes encryption cycle

2.3.1.3 Model estimation

With all the users' data encrypted, model estimation is now possible. As in the encryption phase, an initialization moment is needed for the execution of this process. Another simple



script, run from the server side, implements the needed steps. Figure 28 depicts the logs from that script.

```
INFO | __main__:<module>:15 - Checking for available clients
INFO | __main__:<module>:24 - Available clients: ['http://127.0.0.1:5001', 'http://127.0.0.1:5002']
INFO | __main__:<module>:25 - Signaling start of model fitting
INFO | __main__:<module>:34 - Start model fitting cycle
```

Figure 28: Log messages of the beginning of the model estimation task

The server, again, checks which clients are available for this task. After receiving the confirmation from both peers, a simple POST request results in each participant internally preparing to start the model estimation. Finally, the last log message refers to the moment when the first peer in the encryption order receives a POST request to start performing the model estimation. The start of this iterative process is illustrated in the log messages on the client that initiates it (see Figure 29).

```
- "GET /is-alive HTTP/1.1" 200 OK
INFO | api_client.routes.fit:start_encryption:39 - Model estimation task setting endpoint [POST]
- "POST /fit/start HTTP/1.1" 200 OK
INFO | api_client.routes.fit:post_fit_run:53 - Iterative model estimation run endpoint [POST]
INFO | api_client.helpers.fit_helpers:fit_model_run_thread:211 - Getting anonymized data from peers [GET]
- "POST /fit/run HTTP/1.1" 201 Created
INFO | api_client.helpers.fit_helpers:fit_model_run_thread:228 - Sending anonymized data to peers [POST]
INFO | api_client.helpers.fit_helpers:fit_model_run_thread:240 - Sending signal for peers to start model estimation [POST]
INFO | api_client.helpers.fit_helpers:fit_model_run_thread:257 - Global model estimation, iteration 0
INFO | api_client.helpers.fit_helpers:fit_model_local:99 - Local model estimation
INFO | api_client.routes.fit:get_iteration_coefficients:127 - Model estimation iteration coefficients request endpoint [GET]
- "GET /fit/iteration/0 HTTP/1.1" 200 OK
```

Figure 29: First model estimation log messages from the first peer in the chain

As the task begins, an initial data exchange occurs. The first peer computes a less processed version of the encrypted data and sends a request to other peers. Finally, it shares suitable data with all participants (not fully illustrated in this example, as there are only two peers). This sharing is a one-time process, eliminating the need for repeated data sharing. The first peer also uses a POST request causing other participants to start the process on their side as well and perform the first local model estimation. Once completed, each peer requests the local coefficients from other peers, enabling the global model iteration. This is done sequentially (first peer requests coefficients, then the second peer does the same requests, and so forth) to ensure stability in the process. In fact, each time a peer has access to all the (encrypted) coefficients, they execute the global iteration and, if the stopping criteria is not met, they proceed to the next local estimation. Synchronous execution by all peers might lead to mixed



iterations and vulnerability in the exchanged data's cohesion due to differences in processing times or computational performances across systems.

The log messages in Figure 30 are produced when the participant has just finished requesting the coefficients from its peers. It immediately performs the global model update, and it is possible to see which iteration is currently being performed and which was the stopping criteria. It is possible to see that none of the conditions for halting estimation has been reached, so the process goes back to local model estimation and coefficients requests.

```
INFO | api_client.helpers.fit_helpers:fit_model_iteration_update:315 - All local model estimations received. Performing verifications
INFO | api_client.helpers.fit_helpers:fit_model_iteration_update:384 - Global iteration: 12. Max iterations: 500.
INFO | api_client.helpers.fit_helpers:fit_model_iteration_update:385 - Stop criteria: 0.4025463728007693. Threshold: 1e-05.
INFO | api_client.helpers.fit_helpers:fit_model_iteration_update:386 - Performing local model estimation for iteration 13.
INFO | api_client.helpers.fit_helpers:fit_model_local:99 - Local model estimation
INFO | api_client.routes.fit:get_iteration_coefficients:127 - Model estimation iteration coefficients request endpoint [GET]
- "GET /fit/iteration/12 HTTP/1.1" 200 OK
INFO | api_client.routes.fit:post_iteration_coefficients:112 - Model estimation iteration continuity endpoint [POST]
- "POST /fit/iteration/13 HTTP/1.1" 202 Accepted
```

Figure 30: A model estimation process still hasn't reached any stopping conditions and carries on

In this illustrative example (Figure 31), the model estimation has concluded as it reached the maximum number of iterations. This participant will therefore save the last computed coefficients and send a POST request to the next participant in the chain. This will make them finalize the estimation on their side as well and save their last computed coefficients. This signal to stop the estimation is propagated through the chain and the model estimation process is now terminated.

```
INFO | api_client.helpers.fit_helpers:fit_model_iteration_update:359 - All local model estimations received. Performing verifications
INFO | api_client.helpers.fit_helpers:fit_model_iteration_update:409 - Global iteration: 952. Max iterations: 1000.
INFO | api_client.helpers.fit_helpers:fit_model_iteration_update:410 - Stop criteria: 9.991435111618522e-07. Threshold: 1e-06.
INFO | api_client.helpers.fit_helpers:fit_model_iteration_update:411 - One of stopping conditions reached. Model estimation finalized.
INFO | api_client.helpers.fit_helpers:fit_model_iteration_update:414 - Saving model coefficients.
INFO | api_client.helpers.fit_helpers:fit_model_iteration_update:420 - Sending signal to stop model estimation to next peer.
```

Figure 31: A model estimation process finishes by reaching the maximum number of iterations

2.3.1.4 Forecast generation

The final task is to use the coefficients that resulted from the model estimation process and apply them to generate forecasts. This again implies a collaborative process.

A script on the server side is executed to initialize this process. After checking for peer availability, the server shares which timestamps will be the target of the forecast computation. The shared prediction metadata is the verification of which timestamps are available by all peers to generate forecasts for. Finally, the server uses a POST request to start the process. In



this case, since this is not an iterative process, all peers can start the task simultaneously. Log messages can be found in Figure 32.

```
INFO | __main__:<module>:15 - Checking for available clients
INFO | __main__:<module>:23 - Available clients: ['http://127.0.0.1:5001', 'http://127.0.0.1:5002']
INFO | __main__:<module>:24 - Signaling start of prediction computation, and posting timestamps for forecast generation
INFO | __main__:<module>:52 - Get available timestamps from each client
INFO | __main__:<module>:63 - Gathered timestamp information from all available peers
INFO | __main__:<module>:65 - Send predict metadata
INFO | __main__:<module>:75 - Predict metadata received by all peers
INFO | __main__:<module>:77 - Start forecast computation
INFO | __main__:<module>:84 - Predict task initiated successfully
```

Figure 32: Log messages of the beginning of the forecasting generation task

Figure 33 shows the logs from one of the peers, depicting the responses to the server's requests. Two simple messages confirm that this peer has computed its partial forecasts. It proceeded to request from the other participant its partial forecasts for the target variables. Combining the two parts will yield the final predictions for their target variable. A received GET request shown in the logs reveals that the other participant executing the same process and retrieving this peer's portion of the forecasts. The service has now completed its final goal.

```
INFO | api_client.routes.predict:start_encryption:48 - Running encryption start with initial metadata method [POST]
- "POST /predict/start HTTP/1.1" 200 OK
INFO | api_client.routes.predict:get_predict_metadata:72 - Running predict available timestamps endpoint [GET]
- "GET /predict/available-timestamps HTTP/1.1" 200 OK
INFO | api_client.routes.predict:post_predict_metadata:120 - Running predict available timestamps endpoint [POST]
- "POST /predict/available-timestamps HTTP/1.1" 201 Created
- "POST /predict/run HTTP/1.1" 200 OK
INFO | api_client.routes.predict:start_prediction:149 - Running endpoint to start forecast generation [POST]
INFO | api_client.routes.predict:get_results:174 - Running endpoint to get partial forecast results [GET]
- "GET /predict/result HTTP/1.1" 200 OK
INFO | api_client.helpers.predict_helpers:run_compute_forecasts:136 - Finished prediction generation. Saved predictions in CSV format
```

Figure 33: Log messages for the forecasting process on the client side

2.3.1.5 Tests and illustrative results

To perform the first validation tests on the current version of the APIs, a dataset from real (and publicly available [here](#)) power generation wind parks in Norway was used. This dataset contains the target variables for which forecasts were produced. The features used in these tests were engineered from Numerical Weather Predictions (NWP) extracted from the European Centre for Medium-Range Weather Forecasts (ECMWF) Web API service. Both datasets have hourly resolution. The results presented in Figure 34 use data for model fitting from 2020, and forecasts were generated for the first month of 2021. In total, five wind parks were selected for the experiment.



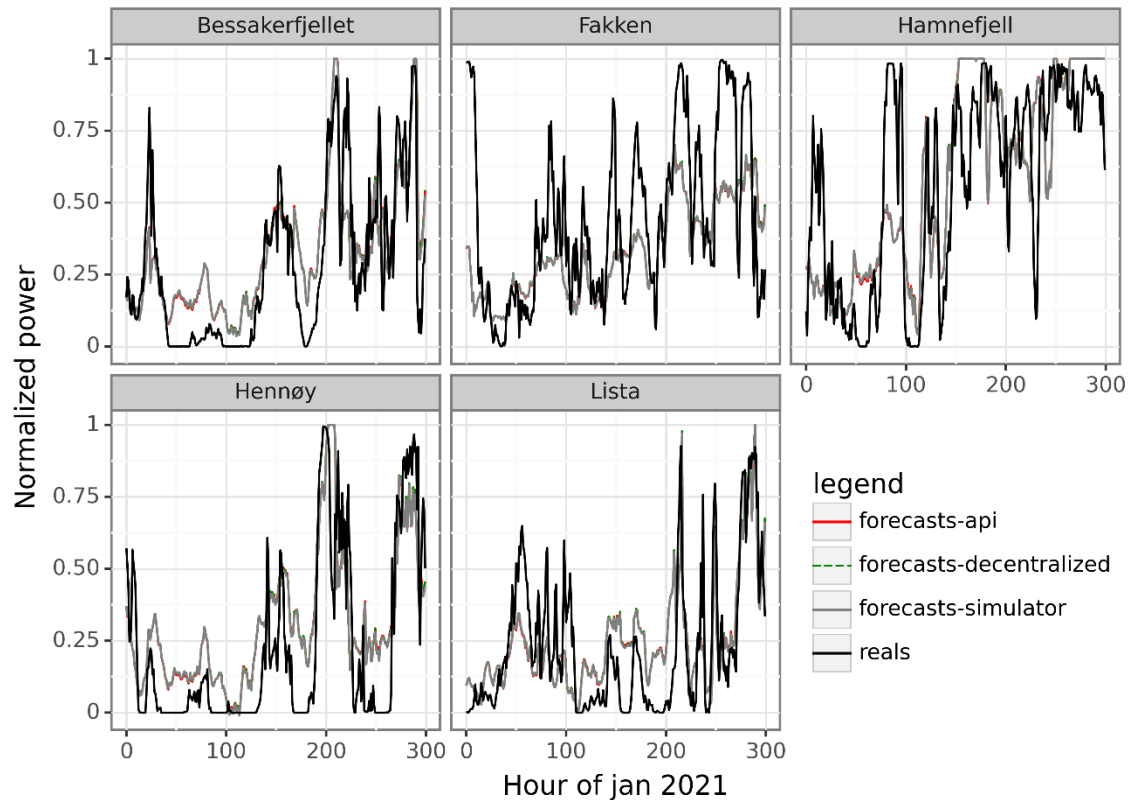


Figure 34: Example of forecasts for 5 wind parks using three different approaches for the FL approach

Three forecast simulations of three distinct approaches are represented (only a few days of January are shown for easier visualisation). For a brief description, using their labels as reference:

- *"forecasts-decentralized"* depicts the scenario where the collaborative model is trained in a decentralized manner without data encryption — peers share necessary information without privacy considerations.
- *"forecasts-simulator"* showcases the output derived from our Python script simulator. It also assumes a decentralized path but employs a data anonymization through encryption. The communications for encryption and model training are simulated with a simple sequential "for" cycle.
- *"forecasts-api"* represents the output of the APIs, the final step of the validation, in which peer communication is fully integrated through the latest developments and using the process workflows illustrated previously.



These experiments allowed to confirm 1) if the data encryption impacts the performance of the model, and 2) perform the same assessment for the integration of peer communication through a RESTful API.

All three experiments, based on the same datasets, resulted in very similar forecasts (in fact, Figure 34 shows three virtually overlapping curves). This validates the implementation of the API as conforming to expected results.

2.3.2 Documentation

Both APIs currently reside in an INESC TEC GitLab repository. They were developed using the FastAPI web framework in Python. When the APIs are instantiated, an automatic, interactive documentation that is OpenAPI specification-compliant in Swagger format is made available through a dedicated endpoint. The most recent version of this documentation can be accessed using [this link](#) (client API only).

It is important to note that the endpoints that make up the APIs so far were developed mainly for interaction between API instances and are not to be invoked by users. Methods enabling interaction with the service (specifically, for ingesting the data used in the process and to edit some configuration parameters) are yet to be implemented.



2.3.3 Integration with the Data Space

Figure 35 depicts the two options that are being considered to integrate the FL forecasting service into the ENERSHARE Data Space environment.

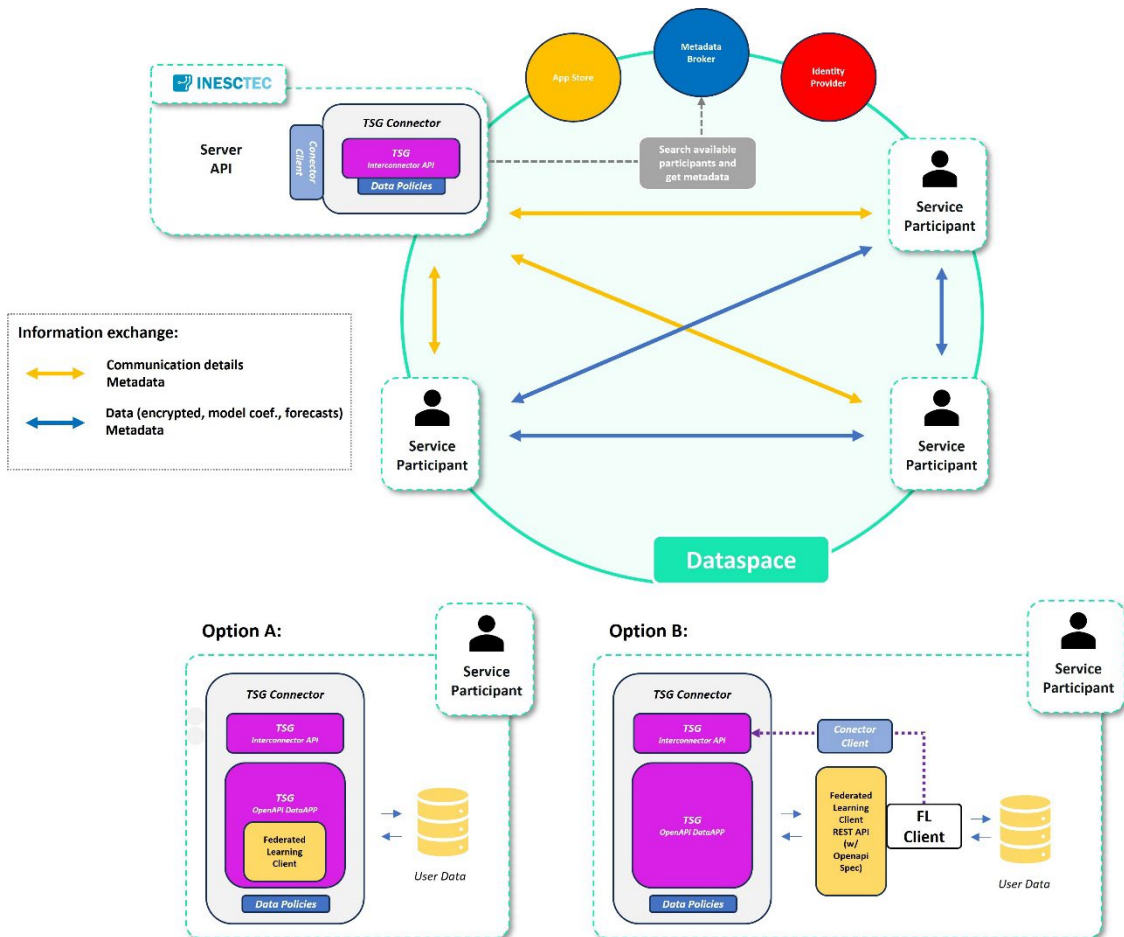


Figure 35: Plan for integration of Multivariate energy time series Federated Learning forecasting service with Data Spaces (server integration and options for client integration).

First, it is important to remember the two types of APIs that are being developed. These are:

1. **Central FL Server API:** This component is responsible for discovering active FL agents (Service Participant in Figure 35) and retrieving their meta-data, including the FL client version, availability, and communication endpoints. It features a Python API with a set of scheduled tasks that initiate the federated learning process (previously outlined in

deliverable D6.1.). The Central FL Server API calls all active agents (with valid Client API versions) to participate in the FL forecasting run.

2. **FL Client API:** Installed locally by each FL agent/participant, this component includes a RESTful API with essential functionalities for peer-to-peer data exchange and the execution of FL forecasting algorithms. It is important to note that this service does not require additional data beyond what is provided by each peer.

The primary demonstration flow is currently being tested, exclusively using conventional HTTP M2M communication. This approach was considered to initially focus on identifying potential bottlenecks in the methodology and refining the essential APIs that will be the key enablers of the FL forecasting process. In parallel, the Data Space integration is also being tested, with an additional set of APIs being prepared in accordance with the official TSG Documentation (<https://tno-tsg.gitlab.io/>) provided by TNO. As depicted in Figure 35, two integration options are currently under examination and testing for exposing both the **Central FL Server API** and **FL Client API** into the ENERSHARE Data Space, namely:

- **Option A:** The **FL Client API** is distributed as a **FL Data APP** (following the recent example of TSG Federated Learning Data APP <https://tno-tsg.gitlab.io/docs/data-apps/existing/#federated-learning>), which can be downloaded from the APP Store and instantiated at the TSG connector.
- **Option B:** The official **TSG OpenAPI Data APP** (<https://tno-tsg.gitlab.io/docs/data-apps/existing/#openapi-data-app>) can be used to expose both client and central server RESTful APIs to the Data Space environment (based on the Openapi specifications of each service)

Regardless of the chosen implementation flavour (Option A or B), several Data Space components will significantly contribute to the demonstration flow of this service.

1. **Connector.** The TNO Security Gateway (TSG) connector considered as reference dataspace communication point for each service, within the dataspace environment. Will be used for both central FL Server and the Client FL API.
2. **Identity Provider:** Identity provision (identification) for each FL client which will be exchanging data during the FL process.
3. **Metadata broker:** Here to be used by each FL client to register their self-descriptions, containing details of their service (API version, availability, data sharing policies, etc.) and by the Central FL Server to find which FL clients are currently active in the DS environment and can be called for the FL forecasting process.
4. **APP Store:** Service that provides a secure registry for the FL Data APP (Client API).



2.3.4 Next steps

While a basic version of the APIs exists and functions independently (yet to be integrated into the Data Spaces framework), several updates and improvements are listed and can be applied to increase the quality and robustness of the solution. These improvements cover various aspects, from refining the methodology to boosting software performance. The forthcoming developments include:

- **Data ingestion in the API:** so far, ad hoc data loading and ingestion has been used to test the API with the available dataset. A data parser module is yet to be implemented, which will allow users to load their own data onto the service.
- **Communication stability:** since most of the processes rely heavily on communication and operate through an iteration logic, it is important to ensure that the system is safeguarded against occasional network issues. Simple strategies like retrying failed requests or implementing strategies for when a peer is no longer responsive are fundamental.
- **Optimization of data sharing over the network:** especially during the encryption phase, the data being exchanged can amount to a great volume. Since the payload size should be limited to accommodate bandwidth limitations and to avoid long waiting times for each individual request, strategies such as streaming of data can help break down the data being sent into smaller segments.
- **Model estimation convergence analysis:** the proposed methodology implies verifying, in each iteration, if there was any malicious noise injection, that is yet to be implemented.
- **Distribution and deployment:** Package the service to be distributed and easily deployed as a Docker container.
- **Documentation:** improve the API's documentation and examples for further clarification.
- **Data Space integration:** some adjustments are expected when this important step is finally possible. Namely, adjusting the server instance as a standalone entity according to the features available in the Data Space and some of the initial tasks that are now required. For example, registering with the server is expected to not be necessary within the Data Space framework, since the API can be downloaded as a Data App and the Metadata Broker will be automatically tracking which users have installed it.



2.4 Federated transfer learning flexibility potential assessment

2.4.1 Development progress

In the first phase of the development in deliverable D6.1, a simple graphical user interface (GUI) was created for the purpose of Federated Transfer Learning (TL) Framework. In the current phase we have initiated the development of a comprehensive infrastructure designed to support a Federated TL Framework, that will have the option to be accessed through the developed GUI or to run automatised federated transfer learning retraining pipelines without the need for the user input. This innovative approach is set to enable us interaction with transformer substations by integrating them into a cohesive DT, a digital replica of a physical transformer substations. This DT framework will enable us to remotely manage and control each substation. Central to this development is the integration of [Kubernetes](#), a powerful system for automating deployment, scaling, and management of containerised applications. On top of Kubernetes will run a distributed ML training framework [Ray](#), which facilitates efficient handling of large-scale data processing and machine learning tasks.

Further enhancing our system's capabilities is the implementation of an advanced orchestrator, [ZenML](#) or [Flyte](#). This orchestrator is offering the agility to trigger on-demand retraining of machine learning models for each substation within the digital twin. This way we can retrain each model from a central station with new locally stored data, improving the efficiency of the flexibility assessment and energy forecasting models. This framework represents a stride towards realizing a more interconnected and intelligent energy management ecosystem for federated TL.

An example of the concept can be seen in Figure 36. When an orchestrator initialises the retraining of the model, autoscaler ensures there is enough resources, and a pretrained model is downloaded from the data spaces. This model is then used in the TL process, where each transformer station trains their new model through the TL. This concept is easily scaled to multiple transformer substations, mostly because of Kubernetes and Ray. This framework enables us to run multiple other services for each substation, with Federated TL being one of them.



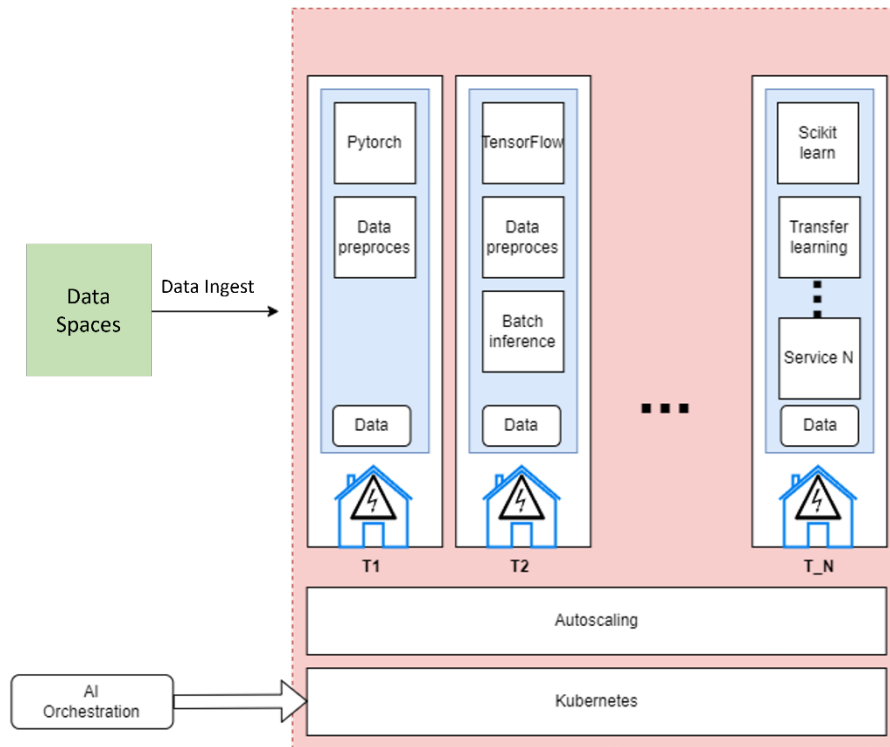


Figure 36: Example of a digital twin model for federated transfer learning

With the first version of the infrastructure built for Federated Transfer Learning framework and testing the service on pilot’s data, the TRL of the service increased from 4 to 5.

2.4.2 Integration examples of the running service on pilot’s data

The initial development of the infrastructure to support Federated TL framework has been developed with the utilisation of Kubernetes, Ray, and ZenML software. An example of a pipeline built can be seen in Figure 37. This pipeline periodically re-runs and re-trains the model based on new stored data on the local machine.

The service will be integrated as a data sink for pretrained machine learning models stored in Data Spaces. The presented pipeline was tested on the pilot’s data, to assess flexibility potential through forecasting of power consumption. Based on the forecasted consumption, we can assess whether the consumption will be in specified range, or it might be under or overutilized. In Figure 38, the comparison between the predicted values and the ground truth produced by the pipeline is depicted.

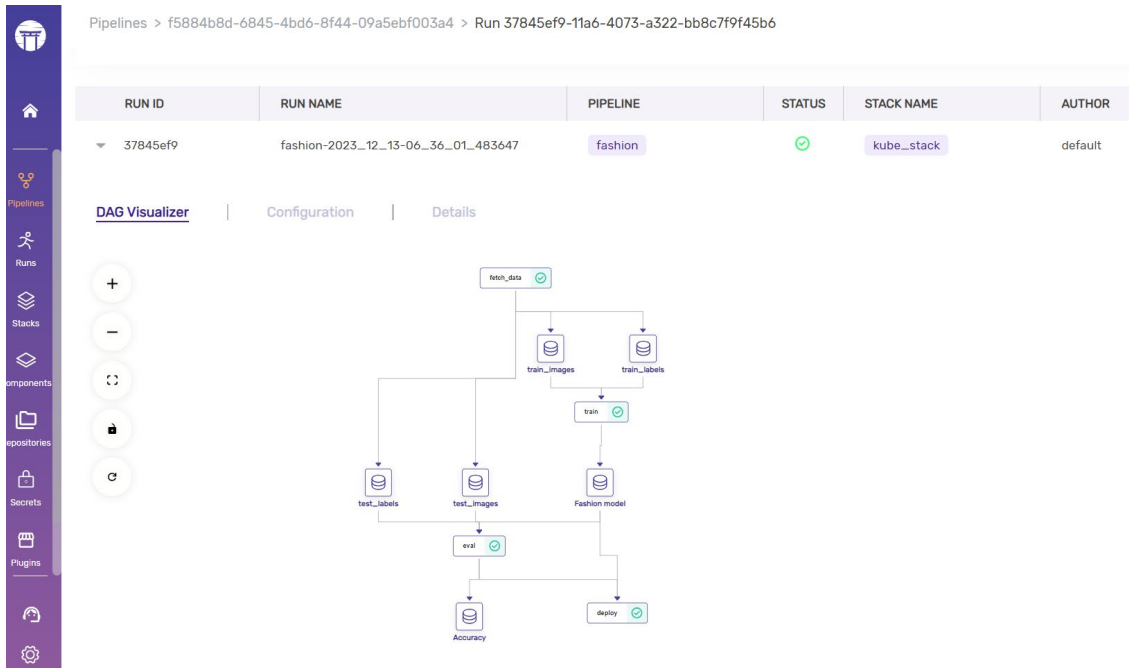


Figure 37: An example of a pipeline in the digital twin of Federated TL framework visualized with ZenML

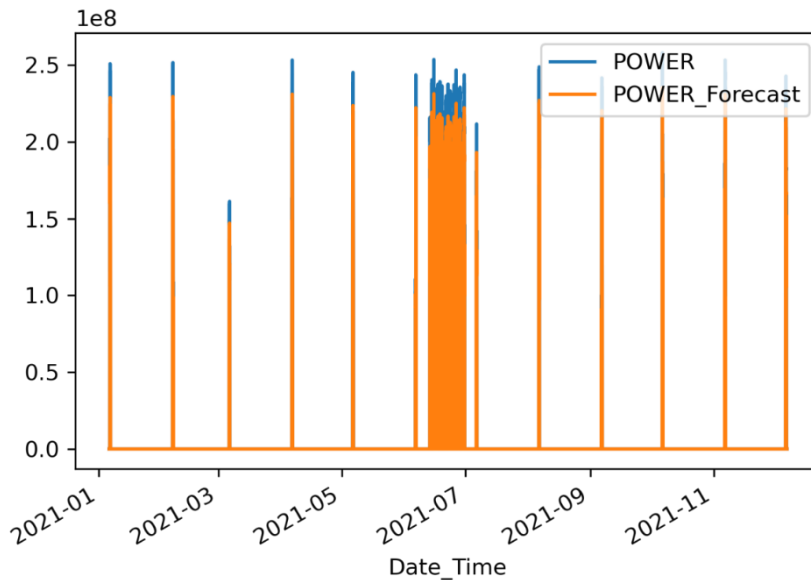
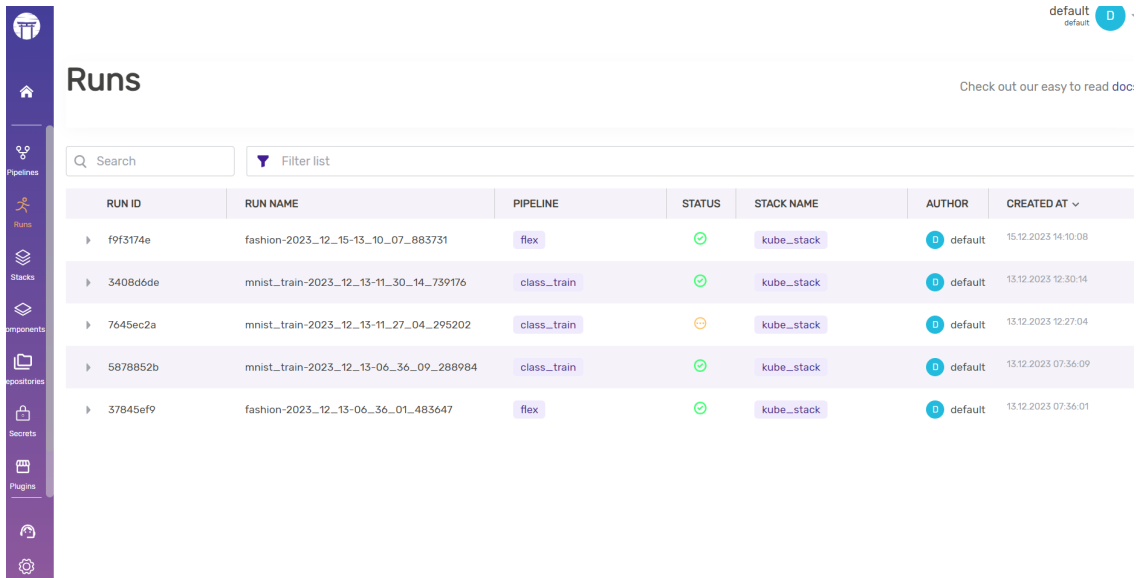


Figure 38: Example of the forecasting performance for flexibility assessment service



ENERSHARE has received funding from [European Union's Horizon Europe Research and Innovation programme](#) under the Grant Agreement No 101069831

The automatic pipeline execution can be seen in Figure 39, where we are testing the retraining of two separate pipelines and evaluate their performance.



The screenshot shows the 'Runs' page in the Enershare interface. It features a sidebar with navigation icons for Pipelines, Runs, Stacks, Components, Repositories, Secrets, and Plugins. The main content area displays a table of pipeline runs with columns for Run ID, Run Name, Pipeline, Status, Stack Name, Author, and Created At. The table contains five rows of data, with most runs in a 'Success' state (green circle) and one in a 'Warning' state (orange circle).

RUN ID	RUN NAME	PIPELINE	STATUS	STACK NAME	AUTHOR	CREATED AT
f9f3174e	fashion-2023_12_15-13_10_07_883731	flex	Success	kube_stack	default	15.12.2023 14:10:08
3408d6de	mnist_train-2023_12_13-11_30_14_759176	class_train	Success	kube_stack	default	13.12.2023 12:30:14
7645ec2a	mnist_train-2023_12_13-11_27_04_295202	class_train	Warning	kube_stack	default	13.12.2023 12:27:04
5878852b	mnist_train-2023_12_13-06_36_09_288984	class_train	Success	kube_stack	default	13.12.2023 07:36:09
37845ef9	fashion-2023_12_13-06_36_01_483647	flex	Success	kube_stack	default	13.12.2023 07:36:01

Figure 39: An example of execution of automated pipelines for federated transfer learning

The pipelines can be easily setup with python scripts, as shown in the example figure below:





```
from zenml.integrations.constants import SELDON, SKLEARN
from zenml.pipelines import pipeline
from zenml.integrations.seldon.services import SeldonDeploymentConfig
from zenml.integrations.seldon.steps import (
    SeldonDeployerStepConfig,
    seldon_model_deployer_step,
)

@pipeline(required_integrations=[SELDON, SKLEARN])
def continuous_deployment_pipeline(
    importer,
    trainer,
    evaluator,
    deployment_trigger,
    model_deployer,
):
    # Link all the steps artifacts together
    x_train, y_train, x_test, y_test = importer()
    model = trainer(x_train=x_train, y_train=y_train)
    accuracy = evaluator(x_test=x_test, y_test=y_test, model=model)
    deployment_decision = deployment_trigger(accuracy=accuracy)
    model_deployer(deployment_decision, model)

# Initialize a continuous deployment pipeline run
deployment = continuous_deployment_pipeline(
    ...,
    model_deployer = seldon_model_deployer_step(
        config=SeldonDeployerStepConfig(
            service_config=SeldonDeploymentConfig(
                model_name="model",
                replicas=1,
                implementation="FLEX_SERVER",
                secret_name="flex-init-container-secret",
            ),
            timeout=120,
        ))
)

deployment.run()
```

Figure 40: An example of built pipeline for ZenML



2.4.3 Integration with the Data Space

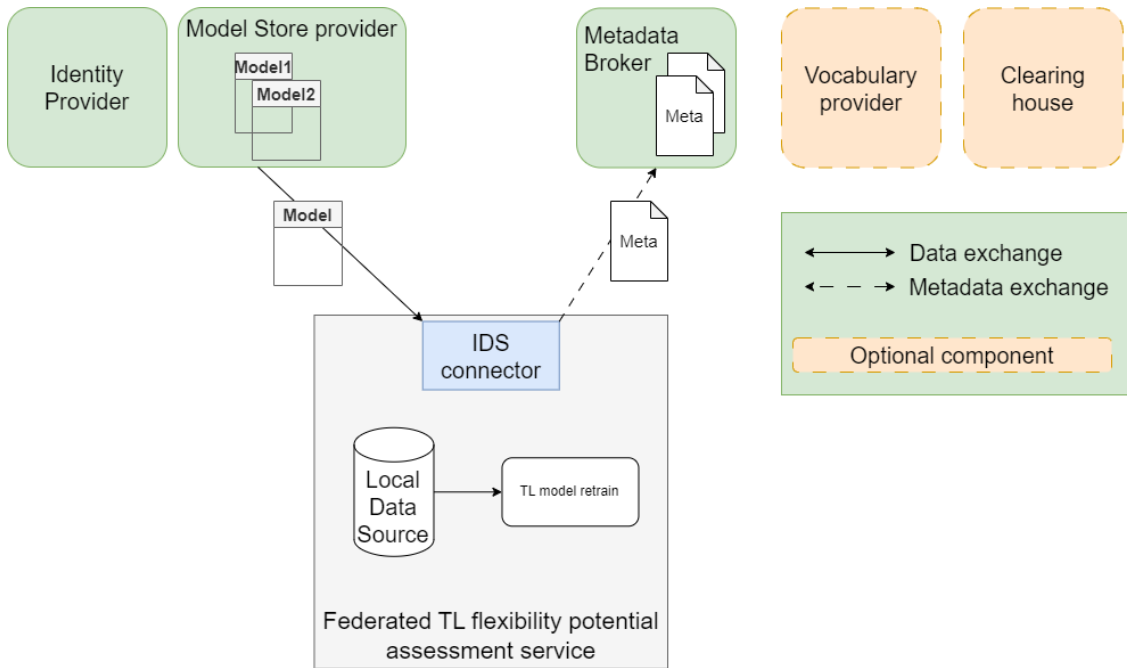


Figure 41 Data spaces integration diagram

Currently the conceptual design of integration with dataspace was created and is in the process of implementation. In the data space, the pretrained model that will be offered as assets will be ingested by each substation through the IDS connector and the model retraining will commence on the local data. Data space will then be utilised to offer additional offline trained models, that can replace or complement existing ones, with the purpose of utilising them for Federated TL for flexibility assessment service.

2.4.4 Next steps

For the final release, we intend to focus on the classification problem type, where we will design a model that can detect whether substation network has electric vehicles, heat pumps, or photovoltaics present and produce report. With it, we will be able to assess if this substation has the flexibility potential to shift certain loads in case of grid overutilisation. Additionally, we will also focus on the integration of the infrastructure framework with the GUI developed in the first phase of the project.



2.5 Comparison between federated learning methods

Table 4 presents a comparison between the different ENERSHARE FL frameworks in terms of characteristics and properties. This table shows that online learning is not yet considered in any of the FL approaches and that INESC TEC approach covers a different type of statistical learning models by augmenting linear models with the additive models framework to capture non-linearities, but keeping interpretability at the same time for the end-user. Furthermore, INESC TEC approach only uses encryption of weights/data and not anonymization. A valuable feature common to all approaches is vertical federated learning, which enables the use of variables (or features) distributed by multiple owners and that is very important in the Data Space context.

Table 4: Comparison (in terms of features) between the different ENERSHARE FL services

	ENGIE	TNO	INESC TEC	COMS
Horizontal data (data owners observe same features)	N	Y	Y	N
Vertical data (data owners observe different features)	Y	Y	Y	Y
Local data encryption	N	N	Y	N
Local weights encryption	Y	N	Y	N
Local weights anonymization	Y	Y	N	Y
Transfer Learning (only model weights are shared)	Y	N	N	Y
Linear model (e.g., additive model)	Y	N	Y	Y
Non-linear models (e.g., ANN)	Y	Y	N	Y
Online learning	N	N	N	N



2.6 Collaboration with Eclipse privacy model working group

As mentioned above, data privacy is a constraint for willingness to share data and therefore FL offers a solution to minimise risks related to data sharing. It is therefore important to enhance data privacy through the promulgation of practices for privacy engineering. Since FL also implies managing privacy locally, the use of privacy models can offer a common guide that will evaluate the minimum level of engagement on privacy.

Taking this into consideration, it is proposed to join synergies with the Eclipse group specialized in “Models for privacy engineering” by two specific activities led by Trialog:

1. Organize a webinar to present Models for Privacy interest group to ENERSHARE. It is expected that it will expose the importance of privacy models and promote an active engagement from ENERSHARE.
2. Coordinate a presentation about ENERSHARE's FL approach to the Models for Privacy group.

It will open a profound discussion on how privacy models can moderate privacy practices within federated learning services.

The Models for Privacy group promotes the use, creation and sharing of privacy models as a common language for descriptive capabilities, and as an information sharing tool for prescriptive capabilities. By this, ENERSHARE will be part of an active group to mutually share insights on models for privacy and federated learning.



3 Data-driven Energy Services

This Chapter outlines the updated versions of the data-driven energy services provided by ENERSHARE, targeting different stakeholders, namely:

- **Local communities:** data-based evaluation of the economic feasibility of sharing resources/assets business models; price estimation of local energy communities.
- **TSOs/DSOs:** improvement of net-load forecasting with crowdsensing; monitoring the growth of flexible resources, estimation, and forecasting of electrical grid status.
- **Multi-energy utilities:** vector energy services (electricity, heat, and gas) by leveraging behind-the-meter/local communities' data.
- **RES value chain:** Data-driven power drive O&M algorithms; benchmarking of wind turbines or wind power plant behaviours.
- **End-user/consumer:** proxy models for studying energy efficiency actions; aggregation of flexibility for market purposes.

These services will be provided by energy stakeholders, who may collaborate with non-energy stakeholders such as facility operators, municipalities, social services, and original equipment manufacturers. Citizens can choose to share their personal data in exchange for new services or compensation. It is essential to note that ENERSHARE ensures data remains at the generation point, employing federated learning algorithms in some services. Data owners retain full control over their data, emphasizing trust and sovereignty as per WP4 guidelines. Integration with the Data Space (explained in a dedicated subsection) is a key aspect in these services.

Table 5 presents a summary of the current development status and outlines forthcoming contributions for the data-driven energy services.

Table 5: Overview of the different data-driven energy services releases in WP6

Data-driven energy service	Features in deliverable D6.1 (Alpha version)	New features in D6.2 (Beta version)	Future features for D6.3 (Final version)
Energy community sizing with assets sharing (INESC TEC)	Optimization model formulated and tested	- Current formulation improvements to broaden its utility - Formulating, modeling, and testing business models - REST API	- Integrate several price mechanisms within the business models - Data Space integration - Frontend
Flexibility modelling of thermoelectric	Main optimization modelling and testing	- Validation of output flexibility measurement - Improvement of the service: parameters,	- Fine-tuning the REST API for integration with the data space



Data-driven energy service	Features in deliverable D6.1 (Alpha version)	New features in D6.2 (Beta version)	Future features for D6.3 (Final version)
water heaters (INESC TEC)		variables cleaning, converter for load diagrams - REST API	- Converted into a standalone library for offline testing and analysis
Community market pool to estimate energy price of internal transactions (INESC TEC)	Main optimization modelling and testing	- Formulation improvements to broaden its utility - Formulating, modelling, and testing business models - REST API	- Integration with the Data Space - Inclusion of shared resources (e.g., batteries)
Failure detection for wind turbines (TECNALIA, HINE, ENGIE)	- Development of physical model (PM generator) - Development of gearbox failure detection and explanation models - Development of the DT for the hydraulic pitch system	- Data pre-process and features extraction - API docs - DS integration details	- Failure hybrid model - Failure classifier based on explainable ML models - Refined the tool's API - Data Space integration
Substation Load forecasting tool (NESTER)	- Forecasting model formulated and tested - Feature engineering process tested - Meta-Learning model trained	- Implementing the Pattern Sequence Forecasting (PSF) model - API documentation - DS integration plan	- Inclusion of PSF models with different clustering algorithms - Refined the tool's API - Data Space integration
Energy usage prediction (COMS)	Initial model based on electricity consumption	- Enhancing the model to heat forecasting - REST API - Dataspace integration plan	- Evaluating the heat models on different houses and proposing changes - Data Space integration
Service local load forecasts and estimation of electrical grid status (TNO)	- Energy grid simulation tool - Local predictive models - FL distributed platform	- Visual API and MapEditor congruent with open-source geographical location tools - API docs	- Visualisation API for the results - IDS Connector integration
Flexibility Analytics and Register (FAR) service (ED)	Development of main openAPIs documentation, initial/basic analytics based on static information max up and down regulation.	- Integration with COMS and TNO service - User database modification - API docs updates - User Interface - Consent management	- Development of locational flexibility analytics - Development of a visualisation dashboard for analytics retrieved



Data-driven energy service	Features in deliverable D6.1 (Alpha version)	New features in D6.2 (Beta version)	Future features for D6.3 (Final version)
		- 1st cycle of Data Space integration	- Full cycle, e.g., actual compilation of demo data of analytics testing - Integration with COP service
Aggregation of flexibility from end users (FORTUM)	- Initial version of the disaggregation algorithm and of the whole pipeline - Designed to be compliant with the old rules	- Fairness and performance improvements - Potential Data Space integration plans - Adaptation to the new SVK rules	- Frequency source integration - Prequalification-like end-to-end test - Data Space integration
Data-driven management of surplus renewable energy generation in distribution systems (RWTH)	Not included in D6.1	- Development of the two stages ML approach - API development - Data Space integration plan	- Finetune ML models with pilot data - Data Space integration plan

3.1 Energy community sizing with assets sharing

3.1.1 Development progress

New developments were implemented since deliverable D6.1, namely:

- The transformation of all installations to a Point of Delivery (PoD) type of installation (previously, only individual installations were in this format). This allows for greater flexibility regarding the operation of shared resources, that can now participate in the energy community market as an individual installation, instead of automatically allocate their energy to the individual installations.
- The development of business models, currently being tested: e.g., PoDs with shared assets can allocate their energy directly to their owners (who also own individual installations) vs PoDs with shared assets participate in the local energy market and only after the ownership of this assets is considered to share the benefits obtained vs mixed solutions between the previous two business models. The different business models result in different ways to distribute the benefits *a posteriori* between the members (i.e., after the computation of the sizing), which can also constitute part of the business model.



Currently, the tool has an active private GitLab repository [here](#). In the future, the tool will be distributed as an open-source Python library in a [public GitHub repository](#).

A comprehensive REST API framework has been developed, seamlessly bridging the codebase with the data space connector. While the REST API functionality of the service is operational and functions seamlessly with actual data, the codebase has not yet attained the status of a completely self-contained package integrated with REST API. While a standalone tool has not been realized, the existing REST API and the advanced stage of the current solution, taking into account the successfully implemented, tested, and approved shared battery assets and various business models, allow the tool to be classified at TRL6 for this project phase.

3.1.2 Documentation updates

A RESTful API was developed in Python using the FAST API web framework. A repository has been created on GitHub, which can be accessed in this [link](#). The documentation can be accessed [here](#).

3.1.3 Integration with the Data Space

The services being developed by INESC TEC within the scope of Task 6.2. (energy community sizing, REC pricing and EWH flexibility) share a similar (on a high level) integration with ENERSHARE Data Space components. This integration is illustrated below, in Figure 42.

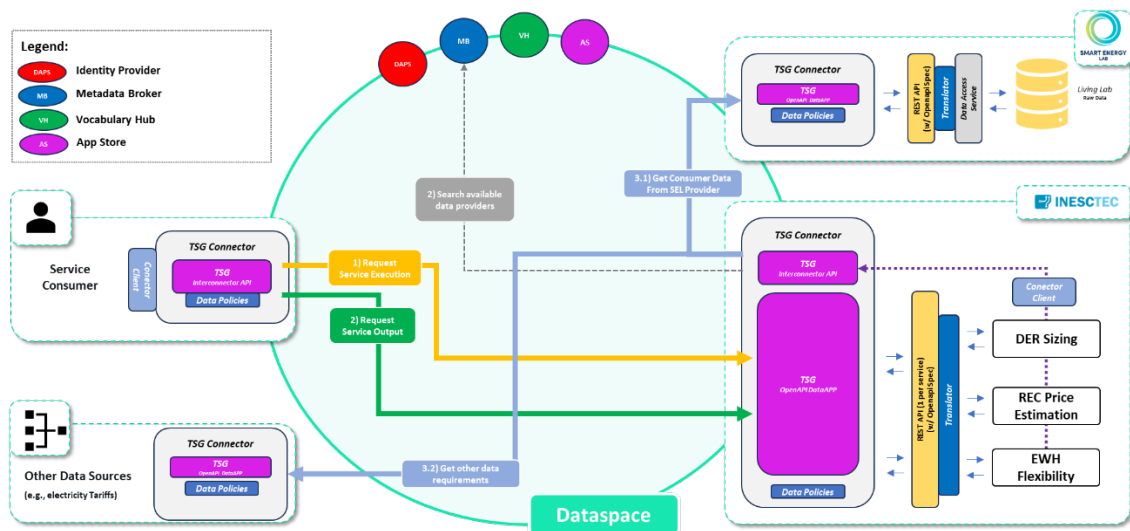


Figure 42: Energy community sizing, REC pricing and EWH flexibility services integration with the ENERSHARE Data Space (high level overview)



At a high level, the demonstration flow starts with a Service Consumer requesting an execution from an INESC TEC service. Subsequently, the service undertakes the task of identifying available data providers within the data space capable of supplying the necessary data to meet the execution's data requirements. Several data space components play an important role in this flow, namely:

1. **Connector**. The TNO Security Gateway (TSG) connector is here considered as reference dataspace communication point for each service, within the dataspace environment. Two main forms of interaction are considered:
 - a. Interaction using the Inter Connector API (<https://tno-tsg.gitlab.io/docs/core-container/api/#inter-connector-api>)
 - b. Interaction using the TSG OpenAPI Data APP (<https://tno-tsg.gitlab.io/docs/data-apps/existing/#openapi-data-app>)
2. **Identity Provider**
 - a. Provides identities as well as the Dynamic Attribute Tokens for continuously verifying the claims attached to the identities.
3. **Metadata broker**
 - a. Provides the metadata (aka Self Descriptions) of connected connectors to provide discoverability within the dataspace. This will allow INESC TEC services to find which service providers are available and the respective data to be shared (and in which conditions, depending on the predefined policies by the Data Providers).
4. **Vocabulary hub**
 - a. Contains the vocabulary of all the metadata used within IDS. A “translator” module will be added to each service, capable of adapting service specific vocabulary with the DS vocabulary.

Two key developments are presently underway to facilitate the integration of INESC TEC services, with the previously mentioned DS components.

1. **Python Client**, which facilitates the communication with **TSG Connector Interconnector API** to discover and retrieve published data artifacts (e.g., electricity tariffs or historical data files)
2. **Services REST API** and respective **OpenAPI Specifications** which will facilitate the call of the developed services (DER Sizing, REC Price Estimation and EWH Flexibility) via a dataspace environment. The **TSG OpenAPI Data APP** will be used to safely expose these backend REST API services to the Dataspace



To complement this high-level introduction, we have also prepared (for the services mentioned above) multiple sequence diagrams illustrating service-specific interactions with the data space.

Figure 43 depicts the interaction of the **REC Sizing** service, with the Data Space. The remaining services (**REC LEM Pricing and EWH Flexibility**) are detailed in sections 3.2 and 3.3, respectively.



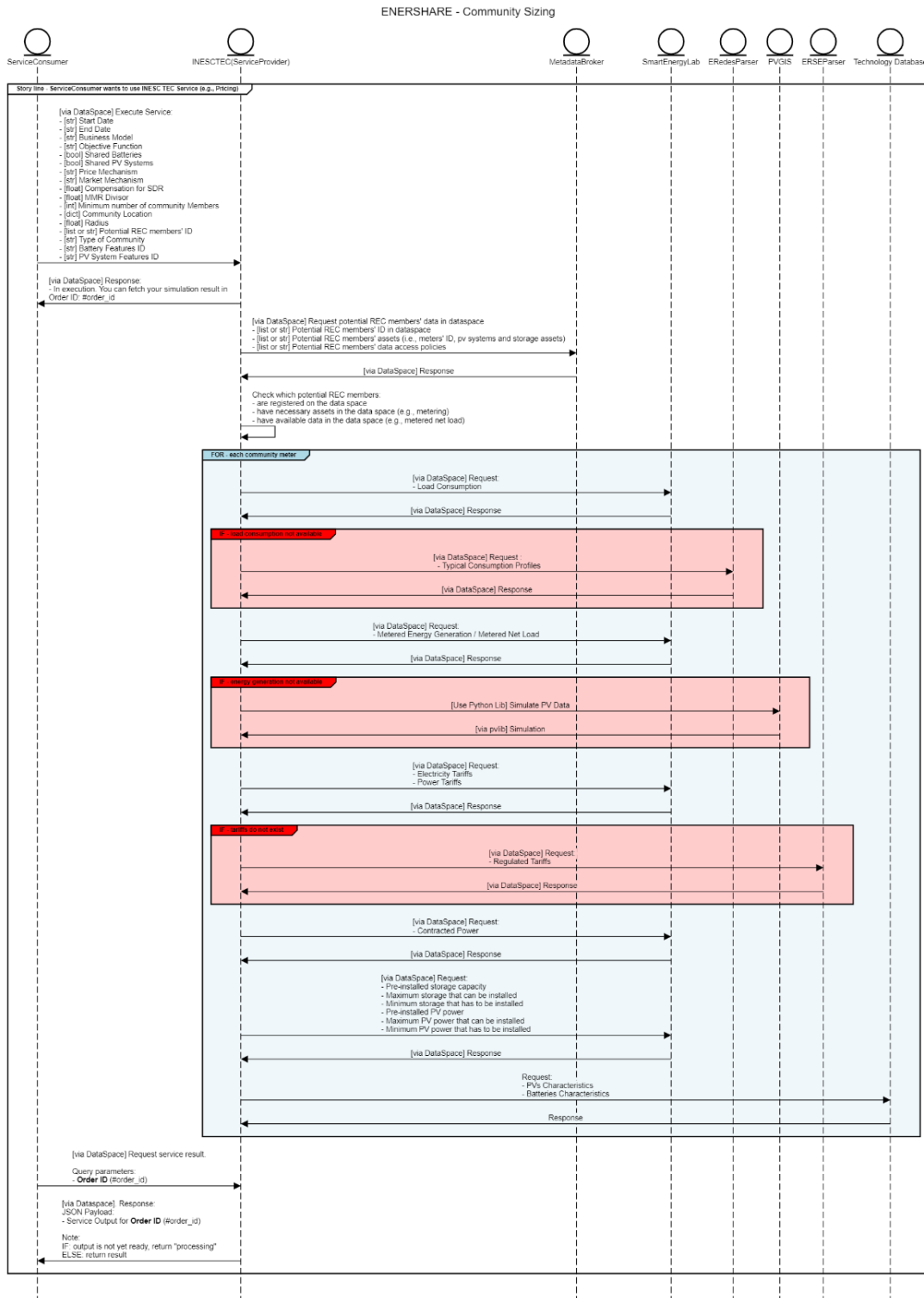


Figure 43: Sizing Service: interaction with Data Space



ENERSHARE has received funding from European Union's Horizon Europe Research and Innovation programme under the Grant Agreement No 101069831

A service request is done by the Service Consumer, that must send a minimum amount of information about its preferences regarding the community sizing service. After that, and considering the information sent by the Service Consumer, the Service Provider will request the Meta Data Broker to check some general points regarding the community, i.e., to verify if there is enough information to compute the sizing service. Then, several requests are made to the SEL to provide installations' characteristics and, when some of these characteristics are unavailable, the service provider uses other sources. Also, a technology database that provides the characteristics of the several possible technologies that can be used in the community will be called to provide information regarding these features. Figure 44, Figure 45 and Figure 46 provide further information about the parameters used by the service.

Description	API	Search on Dataspace	Search on external space
Business Model	x		
Objective Function	x		
Shared Batteries?	x		
Shared PV?	x		
Price Mechanism	x		
Market Mechanism	x		
Compensation for SDR	x		
Mid-Market Rate Divisor	x		
Minimum number of REC members	x		
Location of the REC	x		
Radius	x		
ID of each REC member	x		
Type of community	x		
Battery Features ID	x		
PV System Features ID	x		
start date	x		
end date	x		
ID of each installation		x	
Location of each installation		x	
ID of each REC member	x	x	
Member's ownership of each installation		x	
Power limit for each installation		x	
Buying Tariffs of each installation for each time step		x	x
Selling Tariffs of each installation for each time step		x	x
Contracted Power of each installation		x	x
Contracted Power Tariff of each installation		x	
Grid Access Tariffs of each installation for each time step		x	x
Voltage Level of each installation		x	
Load consumption of each installation for each time step		x	
Energy production of each installation for each time step		x	
Regulated Tariffs			x
Storage Technology specifications			
Minimum state-of-charge			x
Maximum state-of-charge			x
Maximum Charging Power			x
Maximum Discharging Power			x
Charge efficiency			x
Discharge efficiency			x
Lifespan			x
Investment costs (per capacity)			x
PV Technology specifications			
Efficiency			x
Yearly Degradation			x
Generation Profile			x
Lifespan			x
Investment costs (per capacity)			x
Pre-installed storage for each installation		x	
Maximum storage that can be installed for each installation		x	
Minimum storage that has to be installed for each installation		x	
Pre-installed PV for each installation		x	
Maximum PV that can be installed for each installation		x	
Minimum PV that has to be installed for each installation		x	

Figure 44: Parameters interaction with Data Space



Description	Units	Possible format	Further explanation and Observations
Business Model	-	string	
Objective Function	-	string	
Shared Batteries?	-	bool	
Shared PV?	-	bool	
Price Mechanism	-	string	
Market Mechanism	-	string	
Compensation for SDR	-	string	
Mid-Market Rate Divisor	-	string	
Minimum number of REC members	-	integer	
Radius	km	float	
ID of each REC member	-	string	
Type of community	-	string	
Battery Features ID	-	string	
PV System Features ID	-	string	
start date	-	string	
end date	-	string	
ID of each installation	-	string	Delivering Point ID
Location of the REC			
Location of each installation	-	array with two strings or floats	Should provide the coordinates of the installation - [lat, long] - so that we can access the solar production characteristics in PVGIS.
ID of each REC member	-	list of strings	Name/Identification of each participant in the REC
Member's ownership of each installation	% installation / member	float	It must answer the question: how much does each member own each installation?
Power limit for each installation	kW	float	Maximum power that the installation can receive from the grid in each time step.
Buying Tariffs of each installation for each time step	€/kWh	float	Discriminated per each time step.
Selling Tariffs of each installation for each time step	€/kWh	float	Discriminated per each time step.
Contracted Power of each installation	kW	float	-
Contracted Power Tariff of each installation	€/kW.day	float	-
Grid Access Tariffs of each installation for each time step	€/kWh	float	Discriminated per each time step.
Voltage Level of each installation	-	string	-
Load consumption of each installation for each time step	kWh	float	Discriminated per each time step.
Energy production of each installation for each time step	kWh	float	Discriminated per each time step.
Regulated Tariffs			



Description	Units	Possible format	Further explanation and Observations
Storage Technology specifications			
Minimum state-of-charge	%	float	-
Maximum state-of-charge	%	float	-
Maximum Charging Power	kW	float	-
Maximum Discharging Power	kW	float	-
Charge efficiency	%	float	-
Discharge efficiency	%	float	-
Lifespan	Years	float	Expected Duration of the battery.
Investment costs (per capacity)	€/kWh	float	It is the cost, in euros per kWh installed, of a specific storage technology.
PV Technology specifications			
Efficiency	%	float	Probably will not be used (but keep it for now)
Yearly Degradation	%	float	Probably will not be used (but keep it for now)
Generation Profile	kWh/kW _{ins} talled	float	Retrieved from PVGIS
Lifespan	Years	float	Expected Duration of the PV.
Investment costs (per capacity)	€/kW	float	It is the cost, in euros per kW installed, of a specific PV technology.
Pre-installed storage for each installation	kWh	float	How much storage capacity does an installation already has installed (previous to running the problem)?
Maximum storage that can be installed for each install	kWh	float	-
Minimum storage that has to be installed for each inst	kWh	float	-
Pre-installed PV for each installation	kW	float	How much PV capacity does an installation already has installed (previous to running the problem)?
Maximum PV that can be installed for each installator	kW	float	-
Minimum PV that has to be installed for each installati	kW	float	-

Figure 45: Units and Formats of parameters



Description	Default	Optional	Possible Source
Business Model	"all market"	TRUE	Service Consumer
Objective Function	"arbitrage"	TRUE	Service Consumer
Shared Batteries?	No	TRUE	Service Consumer
Shared PV?	No	TRUE	Service Consumer
Price Mechanism	crossing value	TRUE	Service Consumer
Market Mechanism	pool	TRUE	Service Consumer
Compensation for SDR	0	TRUE	Service Consumer
Mid-Market Rate Divisor	2	TRUE	Service Consumer
Minimum number of REC members	2	TRUE	Service Consumer
Radius	4	TRUE	Service Consumer
ID of each REC member	-	FALSE	Service Consumer
Type of community	CEC	TRUE	Service Consumer
Battery Features ID	-	TRUE	Service Consumer
PV System Features ID	-	TRUE	Service Consumer
start date	-	FALSE	Service Consumer
end date	-	FALSE	Service Consumer
ID of each installation	-	FALSE	Service Consumer
Location of the REC	-	TRUE	Service Consumer
Location of each installation	-	TRUE	Smart Energy Lab
ID of each REC member	-	FALSE	
Member's ownership of each installation	-	FALSE	Meta Data Broker
Power limit for each installation	10^12	TRUE	Smart Energy Lab
Buying Tariffs of each installation for each time step	-	FALSE	Smart Energy Lab or ERSE
Selling Tariffs of each installation for each time step	-	FALSE	Smart Energy Lab or ERSE
Contracted Power of each installation	-	FALSE	Smart Energy Lab
Contracted Power Tariff of each installation	-	FALSE	Smart Energy Lab or ERSE
Grid Access Tariffs of each installation for each time step	-	FALSE	Smart Energy Lab or ERSE
Voltage Level of each installation	LV	TRUE	Smart Energy Lab
Load consumption of each installation for each time step	-	FALSE	Smart Energy Lab or E-REDES
Energy production of each installation for each time step	-	TRUE	Smart Energy Lab or PVGIS
Regulated Tariffs			Smart Energy Lab or ERSE
Storage Technology specifications			
Minimum state-of-charge	-	TRUE for Smart Energy Lab	
Maximum state-of-charge	-	TRUE for Smart Energy Lab	
Maximum Charging Power	-	TRUE for Smart Energy Lab	
Maximum Discharging Power	-	TRUE for Smart Energy Lab	
Charge efficiency	-	TRUE for Smart Energy Lab	NREL data, available at https://data.openei.org/submissions/5865
Discharge efficiency	-	TRUE for Smart Energy Lab	
Lifespan	20	TRUE for Smart Energy Lab	
Investment costs (per capacity)	-	TRUE for Smart Energy Lab	
PV Technology specifications			
Efficiency	-	TRUE	
Yearly Degradation	-	TRUE	
Generation Profile	-	TRUE for Smart Energy Lab	Smart Energy Lab or PVGIS
Lifespan	20	TRUE for Smart Energy Lab	Smart Energy Lab or NREL data
Investment costs (per capacity)	-	TRUE for Smart Energy Lab	
Pre-installed storage for each installation	0	TRUE	Smart Energy Lab
Maximum storage that can be installed for each installation	100	TRUE	Smart Energy Lab
Minimum storage that has to be installed for each installation	0	TRUE	Smart Energy Lab
Pre-installed PV for each installation	0	TRUE	Smart Energy Lab
Maximum PV that can be installed for each installation	100	TRUE	Smart Energy Lab
Minimum PV that has to be installed for each installation	0	TRUE	Smart Energy Lab

Figure 46: Default values and sources of parameters



ENERSHARE has received funding from European Union's Horizon Europe Research and Innovation programme under the Grant Agreement No 101069831

Figure 47 presents the outputs of the API.

Description	Units	Possible format
Total cost of REC	€	float
Individual members' cost	€	float
Individual members' savings (when compared with the individual self-consumption case)	€	float
Batteries initial energy content	kWh	float
Batteries energy content at each time step	kWh	float
Batteries charged and discharged energy at each time step	kWh	float
Net consumption behind-the-meter of each installation for each time step	kWh	float
PV generation of each installation for each time step	kWh	float
Imported and exported energy from the retailer of each installation for each time step	kWh	float
PV power that should be installed at each installation	kWh	float
Battery capacity that should be installed at each installation	kWh	float
Contracted power (maximum metered power flow)	kW	float
Total Installed PV power at each installation	kW	float
Total Installed Battery capacity at each installation	kWh	float
Allocated generation diagram from shared PV installation to each prosumer installation at each timestep	kWh	float
Allocated generation diagram from shared battery installation to each prosumer installation at each timestep	kWh	float
Self-consumption energy for each installation for each timestep	kWh	float
Energy bought to other members of the REC for each timestep	kWh	float
Energy sold to other members of the REC for each timestep	kWh	float
Pool price of the community or price of the bilateral transaction (dependent on the business model) for each timestep	€/kWh	float
Financial Indicator 1	-	float
Financial Indicator 2	-	float
Financial Indicator 3	-	float
Financial Indicator 4	-	float
Financial Indicator 5	-	float

Figure 47: Sizing API outputs

3.1.4 Next steps

The following next steps are already being planned:

1. Incorporate several price mechanisms within the business models.
2. Keep developing business models.
3. Develop the external sources to the Data Space as well as their interaction with the service (e.g., ERSE, technologies database, PVGIS access).



4. Development of a basic frontend for the service.

3.2 Flexibility modelling of thermoelectric water heaters

3.2.1 Development progress

Since deliverable D6.1, the tool has resulted in some modifications, namely improvements and reviews of some pre-defined parameters, cleaning of variables, etc. Furthermore, a converter capable of converting an Electric Water Heater (EWH) load diagram to a binary usage diagram was developed from scratch and incorporated. Finally, by pairing the optimization tool with the converter, the first tests were carried out with real data, and it was even possible to obtain some results.

The current state of this service reflects a significant increase in its development, positioning it in the crucial transition stage between Technology Readiness Level 5 and Level 6. At TRL 5, apart from the new functionalities, the service has undergone thorough validation in a relevant environment with real data scenarios and results validation, affirming its capability to perform in circumstances representative of its intended use (EWH real data with domestic surveys responses for capturing hot water usage periods). Additionally, it also includes a robust REST API skeleton with an advanced description of the connection phase, and the main service that incorporates a fully functional codebase that interacts seamlessly with real data.

This API skeleton serves as a pivotal milestone, demonstrating the integration and functionality of the tool's core components with Data Space and virtually establishes the final connections between the codebase and the connector. While the service's REST API functionality is operational, it remains unimpaired with the Data Space and real data. Moreover, the codebase is yet to achieve the status of a fully standalone package (with REST API integration), placing it precisely at the boundary of TRL 6.

3.2.1.1 Built-in load to binary usage converter

The main objective of this service is the ability to extract periods of flexibility and simulate energy savings associated with a given EWH load diagram. In this sense, this service must be able to provide this information using only as inputs the device's charging diagram, some technical information about the device in question, and the user-defined comfort temperature for using hot water. During the process of creating the tool, as detailed in D6.1, it was assumed that a binary hot water usage calendar might be available. However, to make the tool more robust, a converter is now included so that the service can receive the raw diagrams coming from a given load measurement sensor/tool.



The converter (illustrated in Figure 48) that was developed requires as inputs the load diagram in question, and three additional parameters: EWH operating power; maximum permissible temperature of stored water (defaulted at 80°C); user-defined comfort temperature when using hot water (defaulted at 40°C). With this set of information, the converter can detect the periods in which the EWH came into operation, and estimate periods of hot water usage, associated with this *ON/OFF* activation of the EWH. For this purpose, it is assumed that the EWH comes into operation whenever the temperature of the stored water drops below a certain threshold. When the activation period is long enough, it is assumed that hot water has been used, the duration of which is estimated based on the activation duration. In this way, together with a simplified version of the formula for mixing hot water from the EWH with cold water from the network (see D6.1), it is estimated which energy equivalent is possibly used, together with the reheating time (to restore the energy losses), total the total activation period.

$$Heating\ Period_{Total} = Heating\ Period_{Usage=1} + Heating\ Period_{Usage=0}$$

$$Heating\ Period_{Usage=1} = \frac{Heating\ Period_{Total}}{\left(1 + \frac{EWH^{maxTemp} \cdot Flow^{EWH} \cdot c}{3600 \cdot W^{in}}\right)}$$

where,

$$c - \text{water specific heat capacity } (4.184\ J \cdot kg^{-1}K^{-1})$$

$$Flow^{EWH} = \frac{Flow^{total} \cdot (Temp^{Comf.} - Temp^{Inlet})}{EWH^{maxTemp} - Temp^{Inlet}}$$

It is important to note that, as mentioned, even if hot water is not actually used, it is normal for the EWH to come into operation when the temperature, through convection losses, drops beyond the established limit. To try to overcome the signalling of these false positives, the converter determines the average duration between automatic connection intervals, and it is only marked as a period in use, if the usage interval is less than 90% of the average *OFF* periods, or if the estimated period of usage is greater than 2 minutes.

$$\overline{OffPeriod} = \frac{\sum_{i=1}^n OffPeriod_i}{n}$$

$$\begin{cases} \text{If } OffPeriod_i < 0.9 \cdot \overline{OffPeriod} \text{ , } OnPeriod_j^{Usage} = 1 \\ \text{If } OnPeriod_j^{Length} > 2min. \text{ , } OnPeriod_j^{Usage} = 1 \\ \text{Else , } OnPeriod_j^{Usage} = 0 \end{cases}$$



Finally, the converter exports a binary map of hot water usage, which is the main requirement of the previously developed tool.

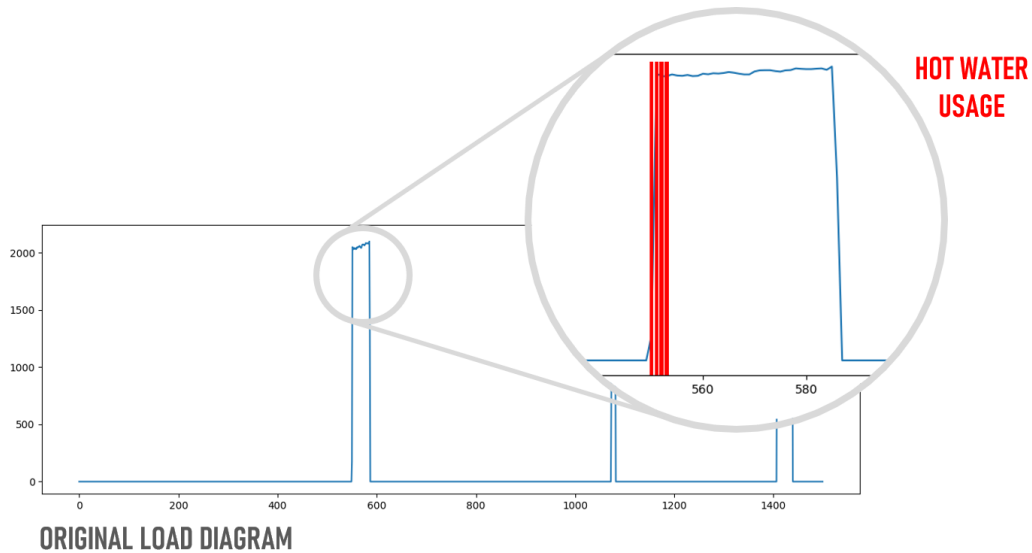


Figure 48: EWH Load to Usage converter visualisation example

Even though, with the provision of a binary usage diagram from the user, where certainty of the binary usage was higher, with the inclusion of this tool, any type of additional user-sided work becomes unnecessary, making it possible to use only the parameters and the diagrams from the measurement system.

3.2.1.2 Parameters and converter refinement using real data

To refine the assignment of parameters and improve some functionalities of both the main tool and the converter, a real dataset is prompted into the service, completely paired with survey results, which contain real binary usage values. Through this example dataset, it was possible to improve some parameters and overcome execution flaws that would only exist for the optimization of longer periods. Furthermore, it also allowed to define which of the solvers is most suitable for the problem in question.

3.2.1.3 Service outputs

In addition to improvements to the service's core, the outputs section was also reorganized, now including a new set of visual features. The output graph structure has been completely revamped, now including direct comparison between the original load diagram and the optimized load diagram. Furthermore, it is also able to directly visualize the periods of hot water usage, heating (*ON* status), and available flexibility. Along with this, a new structure of



numerical results is also implemented, namely periods of flexibility, energy savings, and inclusion of monetary savings, if price data is available.

3.2.2 Documentation

The REST API was developed in Python using the FastAPI framework, that provides interational documentation for visualizing its structure in Swagger format, yielding with the OpenAPI Specification. The current service status and current REST API structure of this service can be previewed through INESC TEC CPES' GitHub platform, using the following links:

[INESCTEC GitHub: EWH Flexibility REST API](#)

[INESCTEC GitHub: EWH Flexibility REST API Documentation \(Swagger\)](#)

The software code of the service is available in open source, in the following repository:

[INESCTEC GitHub: EWH Flexibility Library](#)

3.2.3 Integration with the Data Space

A preview of the workflow for the service interaction with the REST API and other dataspace participants can be found in the sequence diagram presented in Figure 49. For an overall and more detailed overview, please check section 3.1.3.

The first step is made between both service consumer and provider, linking a set of input parameters (see section 3.2.2), that, after validation return, response message of the order status. The user must be registered in the Data Space and should have the minimum required data for running the service (see Table 6). All the non-optional data must be made available and visible to the service provider. After validation the service gathers all necessary information (optional specification, although recommended, can be automatically filled by the service).

Ultimately, the service generates a response while fulfilling the output, and the REST API provides another endpoint, dedicated for retrieving the results (both linked to the users' order ID). If the service was unable to process the order, be for data unavailability or for ongoing processing, it returns informative messages with details about the status.



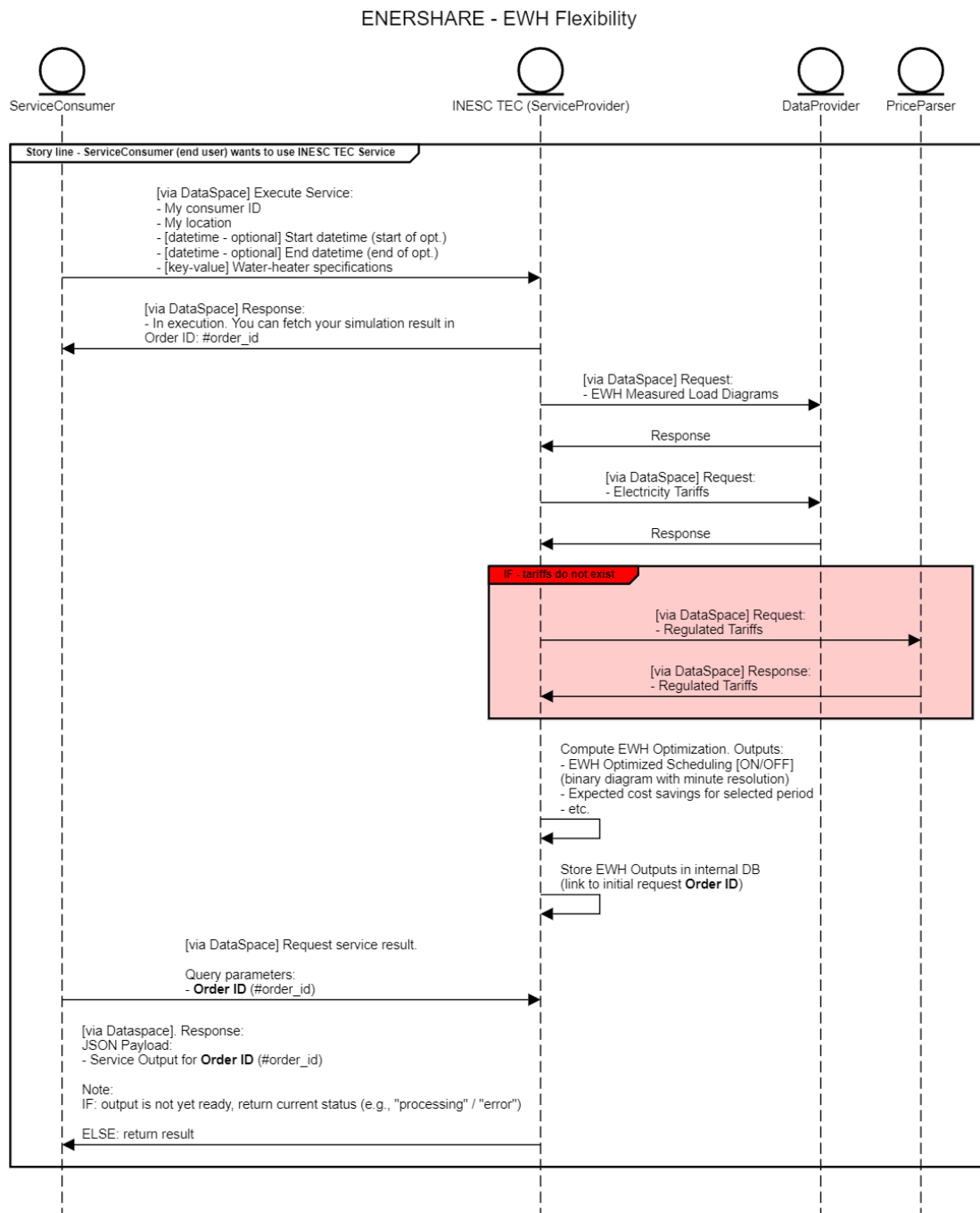


Figure 49: Sequence diagram for the interaction with the EWH Flexibility REST API

Although detailed information is available in the API documentation, both the request and return data summaries are presented in Table 6 and Table 7, for quick reference.



Table 6: EWH Flexibility Data Space request detail

Main description	Variable	Units	Type	Optional	Source
Member identification	User: A string that unequivocally identifies the user for requesting data	-	str	FALSE	Smart Energy Lab
Time period for service application	datetime_start A string that reflects the start date and time (minute resolution)	-	str	FALSE	
	datetime_end A string that reflects the end date and time (minute resolution)				
EWH Specifications	ewh_capacity EWH capacity	l	str	TRUE	Smart Energy Lab or Locally Stored Information
	ewh_power EWH heating power	W	int	TRUE	
	ewh_max_temp EWH maximum allowed water temperature	°C	int	TRUE	
	user_comf_temp Hot-Water Usage Comfort Temperature (minimum user-defined temperature)	°C	int	TRUE	
	tariff Tariff selection between simple (1) or dual (2)	-	int	TRUE	
	price_simple Simple pricing value per kWh	€	float	TRUE	
	price_dual_day Dual day pricing value per kWh	€	float	TRUE	
	price_dual_night Dual night pricing value per kWh	€	float	TRUE	
	tariff_simple Fixed daily simple tariff pricing	€	float	TRUE	
	tariff_dual Fixed daily dual tariff pricing	€	float	TRUE	
	EWH Load Diagram	timestamp A string that reflects the observation timestamp (minute resolution)	-	str	
Load: EWH minute-based consumption		W	float		



Table 7: EWH Flexibility Data Space output detail

Main description	Variable	Units	Type
Member identification	User: A string that unequivocally identifies the user for requesting data	-	str
Time period for service application	datetime_start A string that reflects the start date and time (minute resolution)	-	str
	datetime_end A string that reflects the end date and time (minute resolution)		
Service output	original_energy Total accumulated energy for the real measurement profile	kWh	str
	optimized_energy Total accumulated energy in kWh for the optimized measurement profile	kWh	int
	original_price Total cost for the real measurement profile	€	int
	optimized_price Total cost for the optimized measurement profile	€	int
	avg_daily_energy Average daily energy Consumption for the optimized scenario	kWh	int
	total_flexibility Flexibility availability for the optimized period	min	float
	perc_flexibility Percentage of the total simulated period that can provide flexibility	%	float
	avg_daily_flexibility Average flexibility availability for the optimized scenario	min	float
	savings_cost Pricing savings between real and optimized scenarios	€	float
	savings_energy Energy savings between real and optimized scenarios	kWh	float
	original_usage_profile Estimated hot-water usage profile based on EWH's real load diagram	-	array[{str, bool}]
	optimized_calendar: Optimized EWH calendarization, including timestamp and binary flag for usage	-	array[{str, bool}]



3.2.4 Next steps

The next steps involve defining and fine-tuning the REST API for the connections with the data space. Adjustments may be necessary in the formulation of the service's internal functions, or data structuring, depending on the form of integration with the data connector.

The service should also be converted into a standalone library for offline testing and analysis. It should also be made available into a public *GitHub* repository with description, and basic guides and functioning instructions.

Regarding TRL advancement, the forthcoming steps involve transforming the service into a standalone tool and showcasing the tool's prototype in the relevant environment (via *Data Space*), marking a critical step towards its final design. This progress underscores the tangible progress made and the impending readiness for broader demonstrations and potential deployment.

3.3 Community market pool to estimate energy price of internal transactions

3.3.1 Development progress

A Python library has been developed that implements all possible pricing calculation methods introduced in deliverable D6.1. Namely, the library offers:

- “Vanilla” implementations of the mid-market rate (MMR), supply and demand ratio (SDR), the compensated SDR (SDRC) and the market pool equilibrium, where the service user can provide buy and sell offers, composed of metered net consumptions and respective opportunity costs.
- The iterative approach implementation for price computation, where the buy and sell offers are constructed after a two-stage mixed-integer linear problem (MILP) is solved. The MILP algorithm minimizes individual and renewable energy communities’ (REC) operation costs and is also provided as an independent function of the library that can be customized with several options including timeframe (pre- or post-delivery) and the market paradigm (bilateral contracts or local energy market (LEM) pool).
- The possibility to consider the “shadow” prices of the market equilibrium constraint of the MILP as the prices for the LEM.

This important development makes the tool readily available to be used by any developer.



During the process of creating the library, several functional tests were developed to specifically test each function (main and auxiliary such as data parsing functions). The tool has also been tested with real data, non-specific to the pilot, with some of the main results obtained being shared on deliverable D9.3.

No major bugs were found since D6.1 but some updates to the MILP formulation were made to better reflect the Portuguese legislation for REC management, namely:

- A new constraint was introduced to enforce that all surplus energy is shared amongst the members of the community.
- The optimization is now centred around “meters” and not “members”, which ceases to allow pre-allocation of energy between meters of a same member and the distortion of a real pool market structure.

A REST API framework was also implemented that effectively establishes the final connections between the codebase and the connector to the Data Space. Although the REST API functionality of the service is operational and works seamlessly with real data, the codebase has not yet attained the status of a fully standalone package with REST API integration. This positions it at TRL 6, considering the possibility to already install and use the library.

3.3.2 Documentation

The library has a GitLab repository that is under active development and which can be accessed, with explicit authorization, in this [link](#). A screen capture of the repository can be found in Figure 50. The README file provided in the repository lists all available functions accompanied by an extended explanation of the methodology and specificities of each one. In the future the library will migrate to [this](#) GitHub repository.

An install and test guide are also provided in the repository, as seen in Figure 51. A series of functional tests were prepared to assert the correct operation of all functions, as described in D9.3.



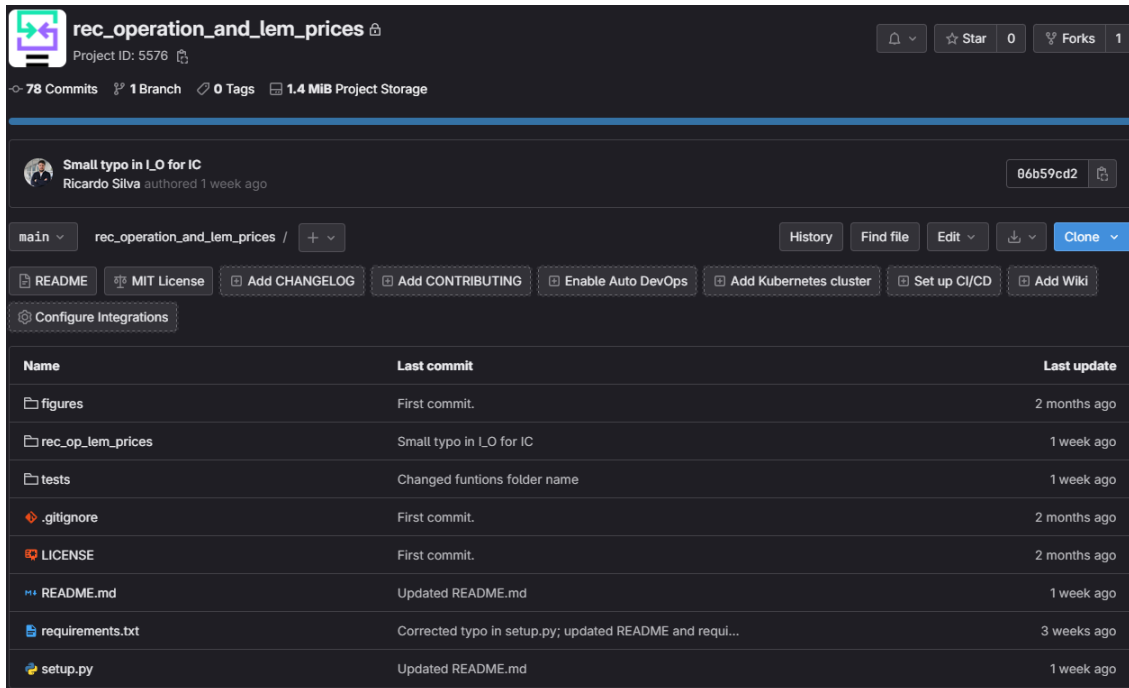


Figure 50: GitLab repository where the Python library developed under the use case for internal REC transactions price estimation is stored

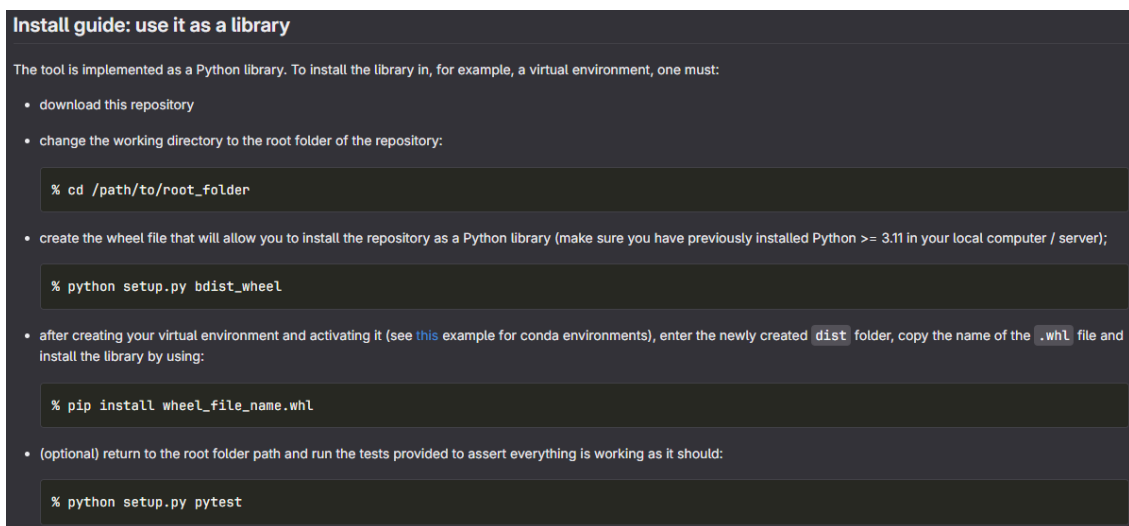


Figure 51: Install and test guide for the library

A second repository has been created on GitHub to accommodate the REST API, which can be accessed in this [link](#). The API was developed in Python using the [FastAPI](#) web framework, which automatically generates interactive documentation either in [Swagger](#) or in [ReDoc](#) formats, both



ENERSHARE has received funding from [European Union's Horizon Europe Research and Innovation programme](#) under the Grant Agreement No 101069831

compliant with the OpenAPI Specification. A screen capture of the Swagger UI can be seen in Figure 52.

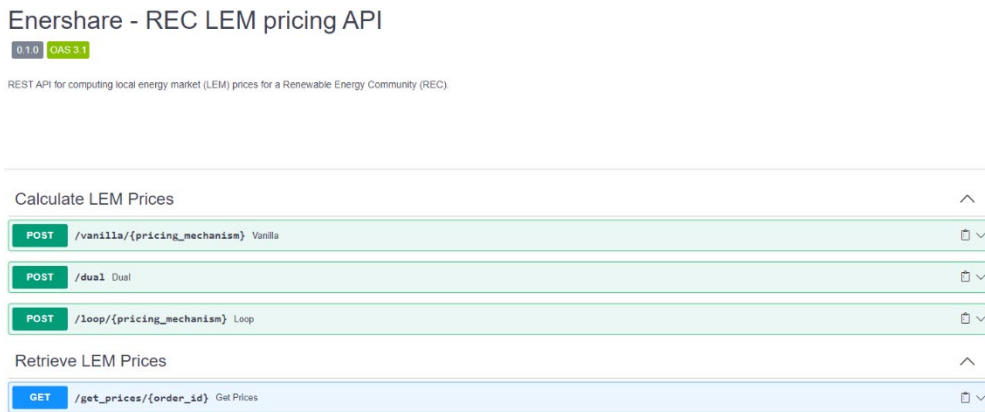


Figure 52: Interactive API documentation provided by Swagger UI

Without needing to instantiate the REST API itself, the documentation can directly be consulted [here](#).

The API provides three main endpoints for requesting the calculation of LEM prices for a REC, one for each of the pricing calculation methods present in section 3.3.1. A fourth endpoint is provided to retrieve the results of a previous request, characterized by an order identification string that is provided when the user makes the initial request.

3.3.3 Integration with the Data Space

The workflow for interacting with the REST API can be found in the sequence diagram of Figure 53. For an overall and more detailed overview, please check section 3.1.3.



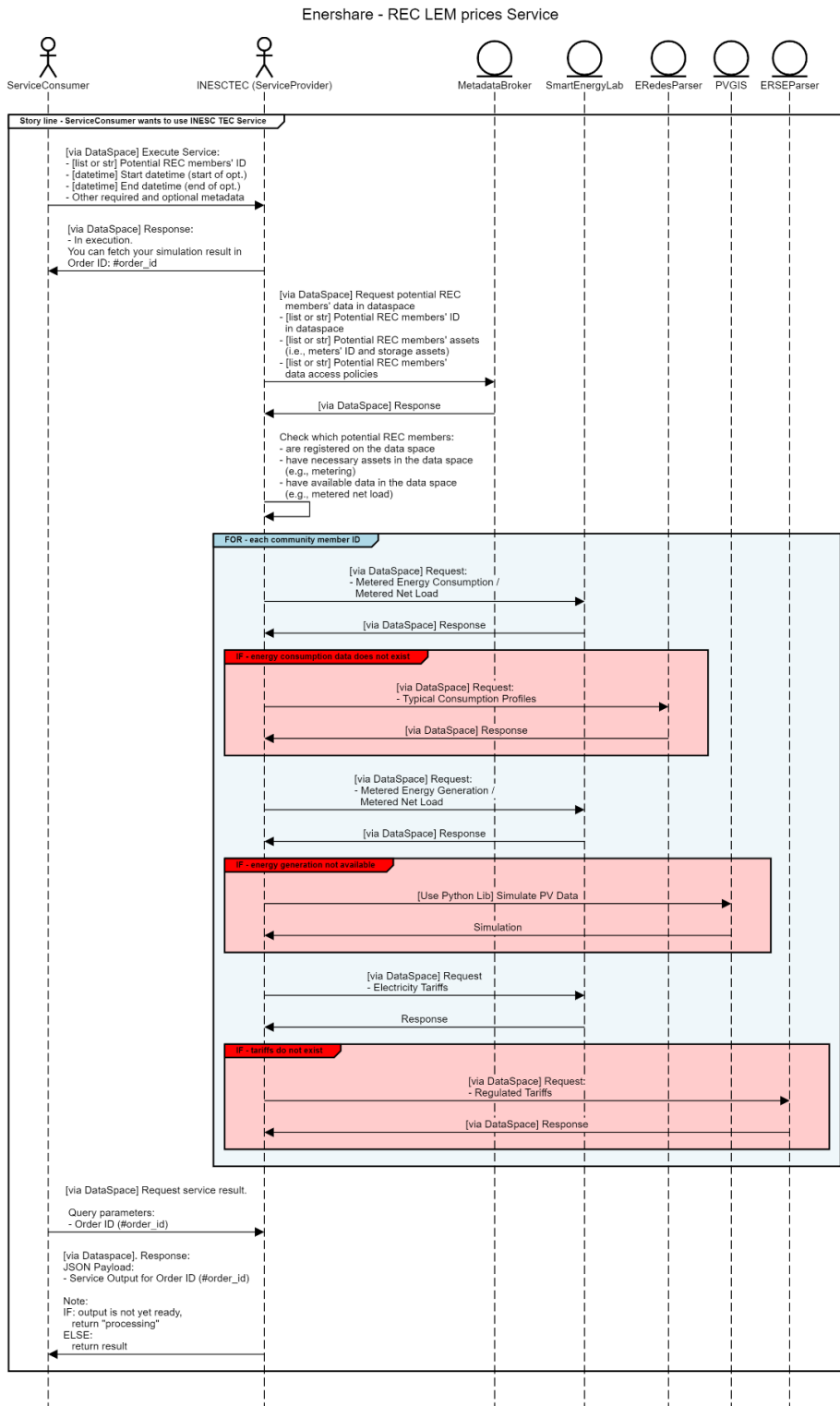


Figure 53: Sequence diagram for the interaction with the REC LEM prices API

A first request is made by the service consumer to the service provider, comprised of a small number of parameters that can be consulted in the [API specification](#), to which, after validation, a response is returned indicating that the request will be processed, along with an order identification string. The request is made for a series of meter ID that make up a real or virtual REC, which must have been previously registered in the Data Space, along with their assets (such as PV panels or storage systems) and other metering information (see Table 8). Such information can be provided upon registration, directly by the potential REC members, by REC managers or by ESCOs that will use the service. All information regarding all member ID and meter ID must be visible to the service provider.

After validating that all registration data is available, the service provider will try to obtain time- and meter-dependent data, namely consumption, generation, and contracted energy prices. Although required to achieve optimal results, adapted to the reality of the members of a potential or existing REC, the unavailability of this data can be circumvented by using publicly available databases with typical profiles and tariffs. A description of this data and the alternative databases used can also be found in Table 8.

After assessing all necessary data, the service provider will generate a response linked to the order identification provided earlier to the service consumer. Another endpoint is offered in the REST API for retrieving the results. If the service was unable to process the data, either because of an impeditive lack of data or because not sufficient time has passed for the computations to finish, the service consumer will receive an informative message. Non-existent order identifications will also be met with an error message.



Table 8: Information required by the REC LEM pricing service

Description	Units	Possible format	Observations and usage	Optional (Default)	Source in Data Space
Member identification: A string that unequivocally identifies a given potential member of the REC.	-	str	For all potential members. For all functions.	FALSE	Smart Energy Lab (SEL)
Meter identification; A string that unequivocally identifies a given meter.	-	str	For all meters.	FALSE	
Contracted power for each meter	kVA	float	For MILP solving.	FALSE	
Meter ownership: Ownership percentages of potential REC members over the meters that will be included in the REC (these include households, UPACs or shared batteries). Typically, one member owns 100% of a meter (e.g. a household), but in certain business models, multiple members can share ownership of assets behind a meter (e.g. shared batteries with a dedicated meter).	%	bool		FALSE	SEL OR local file with information regarding all meters in the pilot
REC type of grid connection: This structure will allow the service to assess which meters are separated by the public grid and which are not. All transactions established under a LEM between meters separated by the public grid are obliged to pay (on the buyer's side) the respective grid tariff for self-consumption.	-	bool	For all pairs of meters. For MILP solving under a bilateral market structure.	TRUE (FALSE)	
Storage asset identification: A string that unequivocally identifies a given battery energy storage system (BESS). Behind each meter, more than one BESS can be defined.	-	str	For all BESS assets. For MILP solving.	FALSE, if the asset exists	
Storage asset nominal capacity	kWh	float			
Storage asset charge efficiency: BESS charge efficiency. If only roundtrip efficiency is available, can be considered as the root square of that value.	%	float			
Storage asset discharge efficiency: BESS discharge efficiency. If only roundtrip efficiency is available, can be considered as the root square of that value.	%	float			
Storage asset initial energy content: (Expected) available energy content of the BESS at the beginning of the optimization horizon.	kWh	float		TRUE (minimum)	
Storage asset maximum charge/discharge power	kW	float		FALSE, if the asset exists	
Storage asset maximum admissible SoC	%	float			
Storage asset minimum admissible SoC	%	float			
Consumption metered: Behind-the-meter (BTM) metered load for the whole post-delivery horizon to be considered. Can be the metered total consumption or net load intake at the meter.	kWh	datetime + float	For whole horizon, with fixed step (15' or 60') for each meter. Limited flexibility for missing data. For all functions.	FALSE	SEL OR E-Redes (Portuguese DSO) typical consumption profiles
Generation metered: BTM metered generation for the whole post-delivery horizon to be considered. Can be the metered total generation or net load outtake at the meter.					SEL OR Photovoltaic Geographical Information System (PVGIS)
Buy opportunity costs: Opportunity cost for buying energy from the retailer, for the whole period to be considered. Can be, for example, the contracted energy buying tariffs with the retailer.	€/kWh				SEL OR ERSE (Portuguese regulatory entity for electrical services) regulated prices
Sell opportunity costs: Opportunity cost for selling energy to the retailer, for the whole period to be considered. Can be, for example, the contracted energy selling tariffs with the retailer.					Potential REC member, REC manager, ESCOs OR SPOT market (OMIE) daily average price minus 20%
Grid tariffs for self-consumption: (Under Portuguese legislation) Tariffs for self-consumed energy from the grid, in vigor for the optimization period.					Must be available for the period considered in the request. Usually defined for periods of the day. Used for MILP solving.



ENERSHARE has received funding from [European Union's Horizon Europe Research and Innovation programme](#) under the Grant Agreement No 101069831

3.3.4 Next steps

The next steps involve the evolution of the current solution to accommodate additional features and its operationalization as a standalone micro-service. In concrete terms, the next steps will consist of:

- 1) The library will need to be updated to accommodate additional features, namely:
 - a. The inclusion of shared resources (batteries and PVs) which are presently absent.
 - b. The possibility to define the divisor for the MMR pricing mechanism.
- 2) In addition, an orchestrator script will have to be implemented between the API and the library that is able to obtain all the necessary data from the data space and other data sources to process an incoming request.
- 3) The possibility to run the API in a Docker container will be made available.

3.4 Data-driven failure detection algorithms for wind turbine components

3.4.1 Development progress

In next subsections, the current development status of the failure detection software of each component is described. Although there have been improvement and progress with respect to the previous version (D6.1 – TRL5), there has not been TRL increase. The increment to TRL7 is expected to be achieved in the final release (D6.3).

3.4.1.1 Anomaly detection of gearbox

Data from a wind farm managed by Engie Green has been received to address the detection of gearbox anomalies. This wind farm is composed by six wind turbines and contains data of the period from January 1, 2018, to November 20, 2023. The dataset incorporates SCADA (Supervisory Control and Data Acquisition) observations from these six wind turbines, providing a comprehensive record of operational parameters. We have also received the maintenance report, which highlighted anomalies in the wind farm and maintenance interventions.

The primary objective is to comprehend the received observations and deduce patterns that describe the normal functioning of the gearbox. To achieve this, we are utilizing the Python pandas and matplotlib libraries.

Furthermore, we are actively collaborating with ENGIE GREEN experts to precisely define and comprehend gearbox anomalies, incorporating their domain expertise into the development of



the anomaly detection model. This holistic approach ensures that our models not only leverage advanced algorithms but also benefit from the insights of industry experts, contributing to a more refined and effective anomaly detection system for the wind turbine's gearbox.

In the current phase of our research, a comprehensive statistical analysis of SCADA data has been performed. The gearbox oil temperature has been identified as the target variable for our investigations. Rigorous tests were conducted to understand relationships between the gearbox oil temperature and other pertinent variables, including active power, nacelle temperature, and rotor speed. According to these abnormal operating conditions. Moreover, we have extracted new features from the existing dataset, notably the median temperature of gearbox oil across the entire fleet. Additionally, we have calculated the absolute difference between the temperature of the analysed wind turbine and its respective fleet's median.

The selected variables will be employed in constructing an anomaly detection model based on Bayesian network principles. A Bayesian network is a probabilistic graphical model used for modelling intricate systems and reasoning within an uncertain context. In contrast to standard failure detection models, primarily based on deep learning, Bayesian networks provide the capability to represent the entire system through a clear and interpretable graph. They also facilitate the more effective inclusion of gearbox knowledge experts during the model construction phase, fostering a synergistic relationship between the power of machine learning algorithms and expert knowledge. At this stage, an initial model has been constructed and is currently undergoing validation using testing data in conjunction with expert knowledge.

For each selected feature, we also generated lagged variables to consider the temporal dimension in the analysis. These selected variables facilitate the detection of abnormal gearbox temperature when training the machine learning model.

3.4.1.2 Anomaly detection of electric generator

Relating to the permanent magnet synchronous generator (PMSG) anomaly detection, data from one turbine with rated power of 800kW has been received. This dataset spans the period from January 2018 to December 2022, and incorporates Supervisory Control and Data Acquisition (SCADA). The sample time of the parameters is 10 minutes.

This dataset has been used to calibrate the physics-based model, according to the methodology explained in deliverable D6.1. It means a progress in the software, obtaining so the Normality hybrid model, although it does not mean a progress in TRL terms (TRL5). This TRL progress is expected to be achieved in the last period of this task. The physics-based model, based in a Simulink model and representing the behaviour of the PMSG in normal condition, has had to be adapted to fit the characteristics of the generator whose data has been received.



As a previous step to calibration of this physics-based model with real data, a **data pre-process** activity has been carried out. From the 64 parameters received, a selection of the required features (13) for the PMSG failure detection service has been done:

- Wind far: parc_code
- ID of the turbine: mac_code
- Date: date
- Temperature in the nacelle: temperature_Data.NacelleTemp
- Stator winding temperature 1: temperature_Data.StatorTemp_1
- Stator winding temperature 2: temperature_Data.StatorTemp_2
- Active power (average): wec_std.Power.avg
- Active power (maximum): wec_std.Power.max
- Active power (minimum): wec_std.Power.min
- Reactive power (average): wec_std.ReactivePower.avg
- Reactive power (maximum): wec_std.ReactivePower.max
- Reactive power (minimum): wec_std.ReactivePower.min
- Wind speed: wec_std.WindSpeed.avg

Missing values has been identified and handled in each case. Besides, normality periods have been clustered in power ranges, to address the calibration in a more optimal way.

Once the dataset has been properly managed, it has been used for calibrating the physics-based model, obtaining in this way the **Normality Hybrid Model**.

Relating to the required failure labelled data, only one event has been labelled in this dataset. This corresponds to the failure of a bearing. This is a kind of failure that was not expected to address, because the focus of the study was the detection of electrical failures. Besides, required sample time of this kind of failures was established in 1Hz. To face this setback, TECNALIA has begun a collaboration with a university (UPV-EHU), which is building a test bench with a scale generator to be able to generate data on electrical failures and thus develop a classifier. The construction of this test bench is out of the scope of ENERSHARE project, but it will be used as data source. The Failure Hybrid Model is expected to be available in the final release (D6.3).

3.4.1.3 Anomaly detection of hydraulic pitch system

In this case, it has not been possible to obtain real operating data from a hydraulic pitch system. However, batches of synthetic data have begun to be generated with an already existing physics model already validated in test bench.



3.4.2 Documentation

A REST API is implemented to give access to the above algorithms. The endpoints of the REST Api will be the following:

- **Root endpoint:**
<https://detection.enershare.urban.tecnalia.dev/api-pilot1/v1/ui/>
- **Generator Anomalies:** /generator/anomalies
 - Description: Retrieve the health status of the generator of a wind turbine, given some specific wind turbine input data.
 - HTTP Method: POST
 - Input data: The input will be a mixture of generator, wind turbine and weather data.
 - Output data: The output will be the health status of the generator: normal operation if everything is going ok, or the failure in case there is an anomaly.
- **Gearbox Anomalies:** /gearbox/anomalies
 - Description: Retrieve a global failure indicator of the gearbox of a wind turbine.
 - HTTP Method: POST
 - Input data: The input will be a mixture of gearbox, wind turbine and weather data.
 - Output data: The output will be a global failure indicator of the gearbox and the probability of failure for each of the input parameters.
- **Hydraulic pitch system Anomalies:** /hydraulic/anomalies
 - Description: Retrieve the health status of the hydraulic pitch system of a wind turbine.
 - HTTP Method: POST
 - Input data: The input will be a mixture of hydraulic pitch system and wind turbine data.
 - Output data: The output will be the health status of the hydraulic pitch system: normal operation if everything is going ok, or the failure in case there is an anomaly.

All the input and output JSON can be transformed into a json-ld align with ENERSHARE ontology using the transformation service developed in WP3. This service is explained in deliverable D3.2.

3.4.3 Integration with the Data Space

Figure 54 depicts the schematic of the Data Space integration for this service. The integration of the wind turbine failure detection services in the ENERSHARE dataspace, will be done using



IDS connectors. We will use the TNO TSG connector for secure data exchange and to facilitate the interoperability between our services and the data sources within the Data space. Each of the three failure detection models will be already trained and will have their own REST API to access their functionalities. The Data App will encapsulate the REST calls and will be responsible for preparing and parsing the input data and redirecting the entering calls in a suitable way. Also, to wrap the output data and provide it in JSON format according to the common ENERSHARE Data Models.

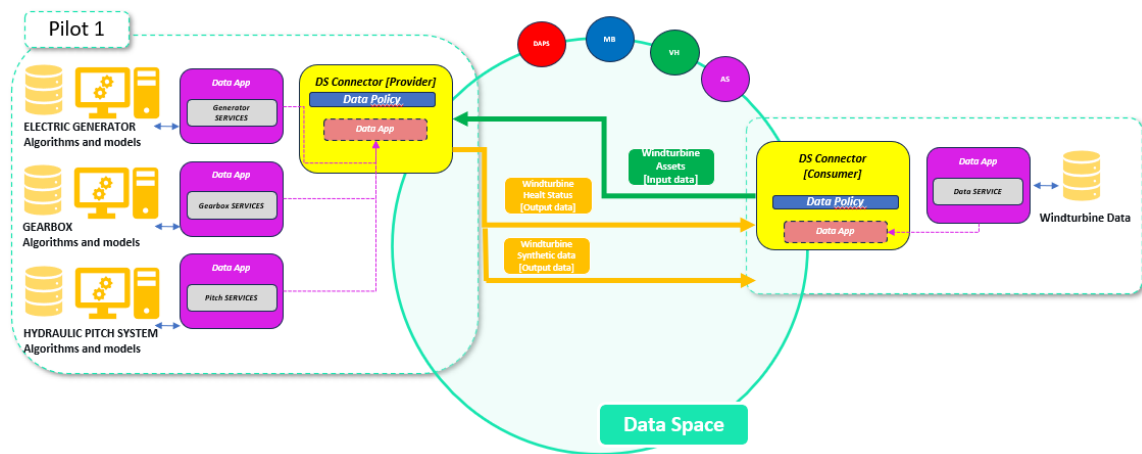


Figure 54: Schematic of the Data Space integration

3.4.4 Next steps

Next activities are the following ones, expecting to reach TRL7:

- Anomaly detection of gearbox:
 - o Development of failure classifier based on ML and explainable AI models.
- Anomaly detection of electric generator:
 - o Development of Failure Hybrid Model.
 - o Development of failure classifier based on ML and explainable AI models, using as data source a small-scale test bench.
- Anomaly detection of hydraulic pitch system
 - o Development of failure classifier based on ML and explainable AI models, using as data source synthetic data.
- Integration of services in Data Space

In addition, Fraunhofer will contribute to the further development of services by exploring options to provide additional data sources and applying a federated learning approach to anomaly detection services as part of the final release.



3.5 Substation Load forecasting tool

3.5.1 Development progress

The main developments made on AI Forecasting – the substation net-load day-ahead forecasting tool – since the delivery of D6.1 are twofold:

- The inclusion of the Pattern Sequence Forecasting (PSF) model into the ensemble stack.
- The development of an API to interact with the tool.

PSF Implementation

The PSF, first introduced by (Martinez Alvarez, et al., 2010) combines Unsupervised Learning techniques with pattern matching for batch generation of daily load profiles. On a first stage, training data is aggregated in daily time-series which are used as input to the clustering algorithms, which group these load profiles. Therefore, this member of the ensemble uses as input a different data set structure, with different features, than the other base learners. To train and predict with PSF, the net-load time series must be pivoted, splitting time and date from the datetime index and using the date as the new index, the time as the columns and populating the cells with the corresponding load values. This way, the daily load profiles (the rows of the new Data Frame) can be clustered together. The new data structure (and the previous one for comparison) is shown in Figure 55.

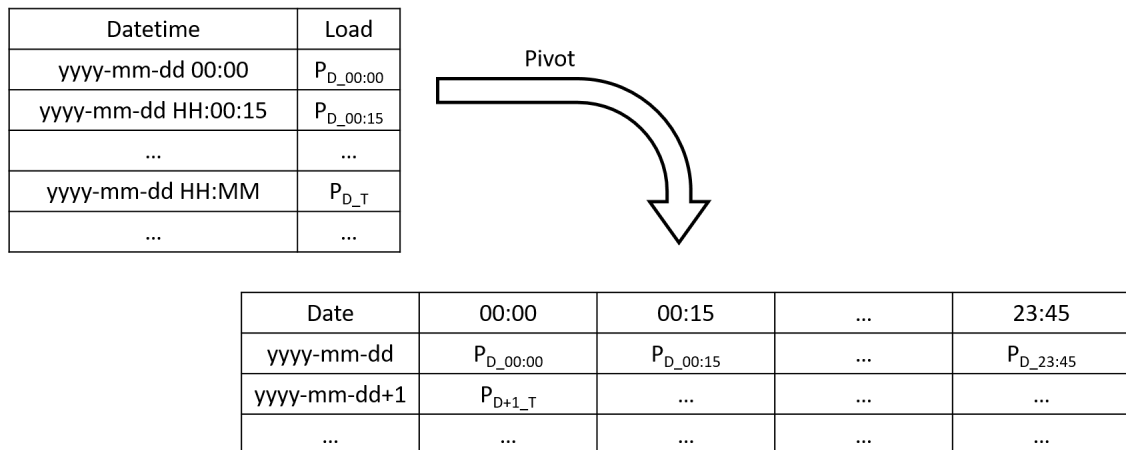


Figure 55: Pivoting of net-load time series into daily observations and quarter-hourly features

The daily profiles are then clustered using the K-Means clustering algorithm. Then, in a next stage, the trained clustering models are used to assign labels to every single load profile observed in the test data and label sequences with a given length are matched in the training



data. For each matched sequence found in the historical data, the subsequent daily profiles are combined (e.g. by averaging) and used to predict the following day. This two-step approach is highlighted in Figure 56 and Figure 57.

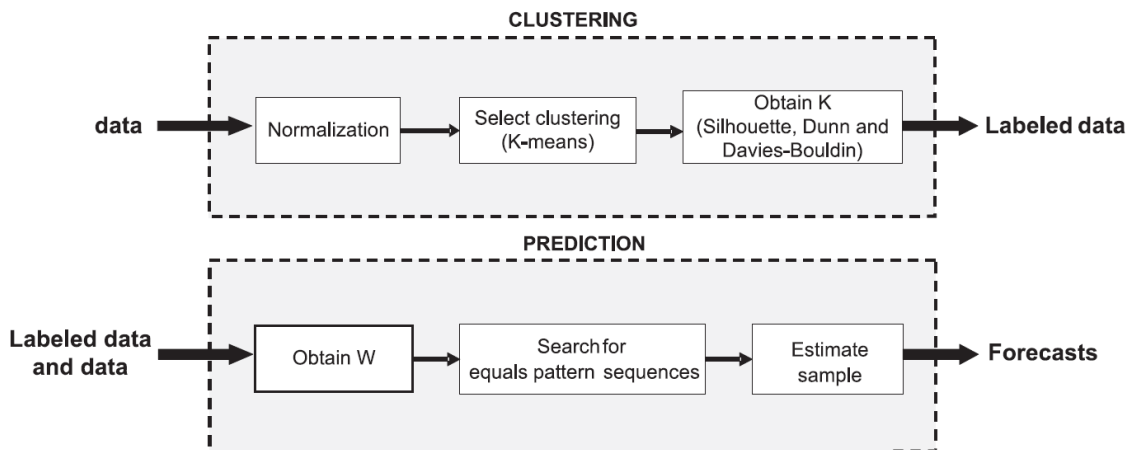


Figure 56: Clustering and prediction processes in the PSF algorithm (Martínez-Alvarez et al., 2010)

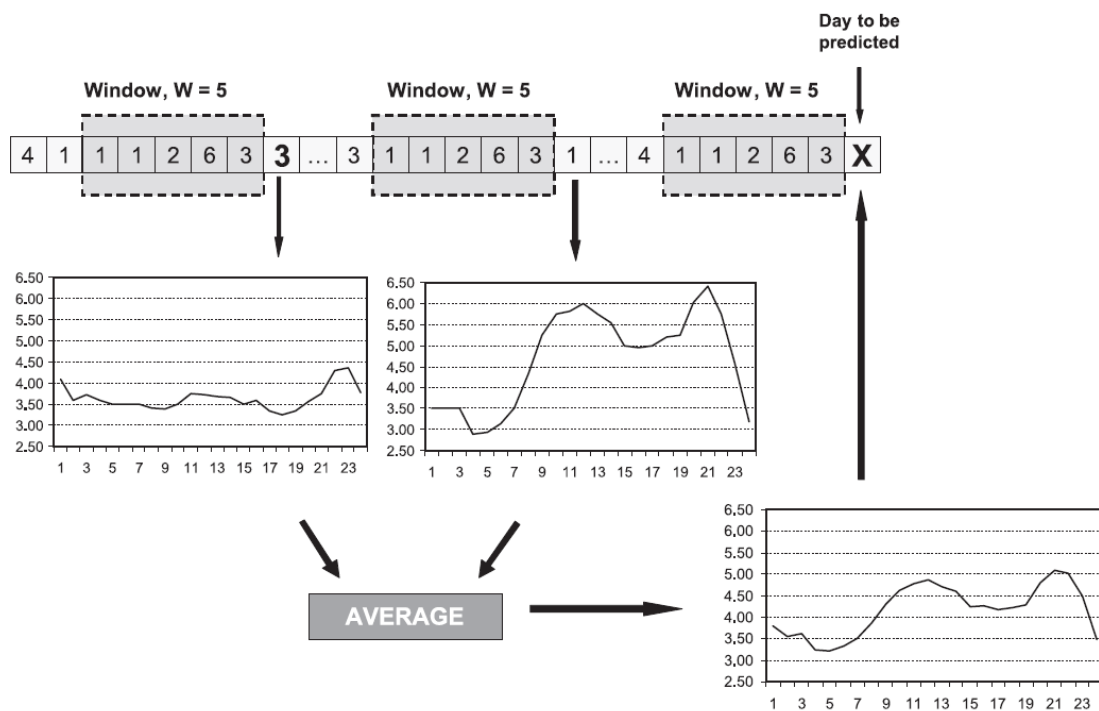


Figure 57: Pattern Sequence matching and generation of forecasts (Martínez-Alvarez et al., 2010)



The generated forecasts are then stacked together with the forecasts coming from the other base learners and used as features for the meta learner. As the input data for the PSF differs from the remaining base learners, the stacking ensemble is implemented manually, i.e., it is not done via the sklearn's `StackingRegressor()` class. The code snippet below shows how the stacking ensemble is now implemented. The `Psf()` class is a self-developed implementation of the PSF algorithm, imported from a `psf.py` file.

```
svr = SVR()
dt = DecisionTreeRegressor()
lasso = Lasso()
xgb = XGBRegressor()
psf = Psf()

final_estimator = XGBRegressor()

svr.fit(xtrain, ytrain)
dt.fit(xtrain, ytrain)
lasso.fit(xtrain, ytrain)
xgb.fit(xtrain, ytrain)
psf.fit_clustering(psf_xtrain)

svr_pred = svr.predict(xtrain)
dt_pred = dt.predict(xtrain)
lasso_pred = lasso.predict(xtrain)
xgb_pred = xgb.predict(xtrain)
psf.predict(psf_xtrain)

base_learner_predictions = np.column_stack((svr_pred, dt_pred,
lasso_pred, xgb_pred, psf.predictions.flatten()))

final_estimator.fit(base_learner_predictions, ytrain)

y_pred = final_estimator.predict(xtest)
```

API Developments

To integrate the AI Forecasting tool with the Data Space and, in general, make it available to the outside world, an API has been developed. The API contains three endpoints: a POST endpoint in which training data is sent to train the model which gets stored in memory; a POST endpoint in which historical data is sent to generate the day-ahead forecasts, which are then



the response of a successful POST request; and a GET endpoint, used to get information on the model, such as its training status and when it was last trained.

3.5.2 Documentation

The tool has a RESTful API for training the model with custom data and obtaining day-ahead forecasts of said model.

Endpoints:

- 1. Root Endpoint ("/"):**
 - Description: This API Endpoint provides info on the available endpoints
 - HTTP Method: GET
- 2. Train Model Endpoint ("/train"):**
 - Description: This API Endpoint receives historical training data in the form of a JSON file.
 - HTTP Method: POST
- 3. Obtain Forecast Endpoint ("/forecast"):**
 - Description: This API Endpoint receives historical data used to forecast the day-ahead load profile, in the form of a JSON file.
 - HTTP Method: POST
- 4. Get Model Endpoint ("/model-info"):**
 - Description: This endpoint provides the training status of the model and the timestamp when it was last trained.
 - HTTP Method: GET

The documentation is published on <https://github.com/RDNester/ai-forecasting-api-docs>.

3.5.3 Integration with the data Space

The schema of the integration of the Substation Load forecasting tool in ENERSHARE Data Space is shown in Figure 58. For the data exchange, the TNO TSG Connectors will be used as a standardised interface for secure data exchange.



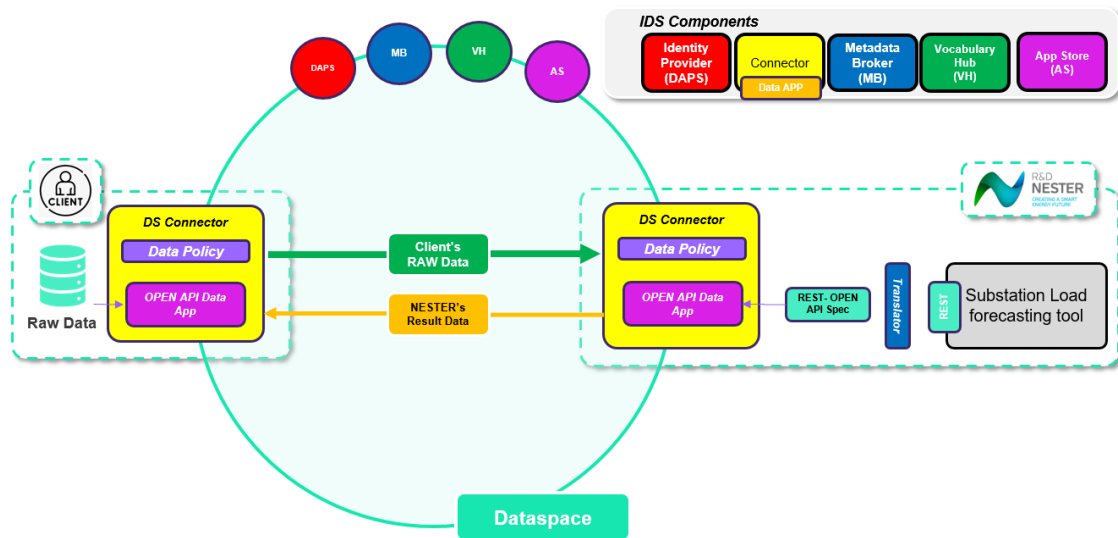


Figure 58: Integration plan for the substation load forecasting tool in the Data Space

These DS connectors will be integrated with the rest of ENERSHARE Common DS components (Identity Provider, Metadata Broker, Vocabulary Hub...). For that purpose, NESTER will configure the NESTER's connector by establishing the identification, authentication, and authorization mechanisms, identify the commonly known, standardised terms to describe data and ensure that the data formats and communication protocols are aligned with the Client's Connector. To ensure the reliability of transmission the data with the required levels of privacy and security standards, the testing process will involve the Client sending RAW data through the DS using their TSG Connector. The Substation Load forecasting tool will then access this data through NESTER's TSG Connector to train the model. Subsequently, the Client will receive the results data containing the generated day-ahead forecast.

3.5.4 Next steps

Currently, the AI Forecasting tool's ensemble of base learners includes an implementation of the PSF algorithm, which uses k-means in its clustering step. To improve the forecasting accuracy, other clustering methods can be tested or different implementations of the PSF algorithm can be included in the stack. Furthermore, an API has been developed with some basic function, but other utilities can be added to make it more complete. Therefore, the next steps in development of the AI Forecasting tool will occur on these three separate fronts:

- Inclusion of PSF models with different clustering algorithms, such as Self Organizing Map (SOM) or Gaussian Mixture, in the base learners' ensemble stack.



- Refinement of the tool's API, namely with the addition of an authentication layer and ability to train and store multiple models.
- Full integration with the ENERSHARE Data Space, as described in the previous section.

3.6 Energy Usage Prediction Service

3.6.1 Development progress

The energy usage prediction service has been expanded to include heat meter forecasts (depicted in Figure 59). Since the KPV utility only records the energy consumption of the houses on an annual basis, we had to change the algorithm to disaggregate the data from the 15-minute measurements of the substations.

The new district heating model was created based on the data received from KPV. The flow chart for creating the model and using the model to predict the heat demand of households by district heating is shown in Figure 59. Based on the data obtained, we found that district heating has high losses of about 40%. The losses vary depending on the time of year. The most important influencing factors are the heating demand and the outside temperature. These losses must be deducted from the heat substation measurement. The prophet model was used to create the model with an additional regressor for the outside temperature.

This model can now predict the district heating consumption. To obtain the predicted energy demand of the households, we had to convert the data according to the annual consumption and the actual demand of the households, which comes from the planning tool. The main disadvantage of such a model is that it can also predict values below 0. We therefore must determine for each house whether it uses district heating or not for a particular day. The most important factors influencing the district heat use are:

- The self-generation of heat energy from appliance operation.
- The self-generation from the persons living in the house.
- The efficiency of the building envelope.



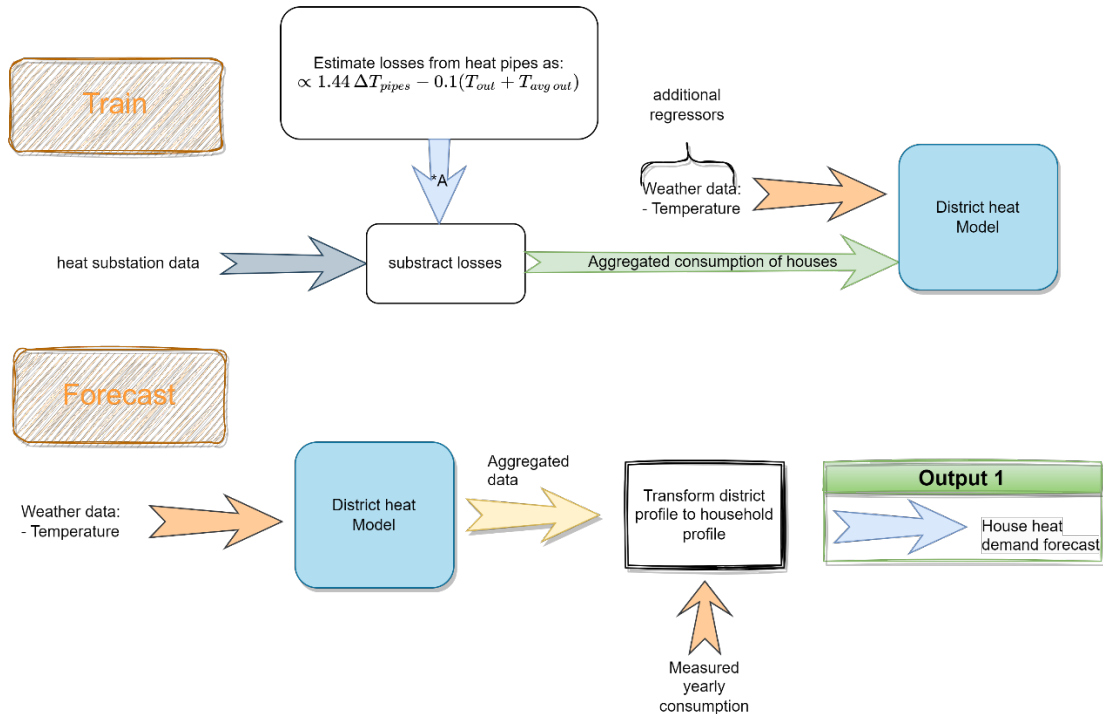


Figure 59: Program flow for training and forecasting data-driven user profiles

A Comparison of the model heat usage forecasting to the measured values (with subtracted losses) is shown in Figure 60. Good matching is observed between forecasted and measured values. In the case of daily profile estimations, Figure 61 shows morning and evening peaks in energy supplied from district heating, which is person behaviour-related features.

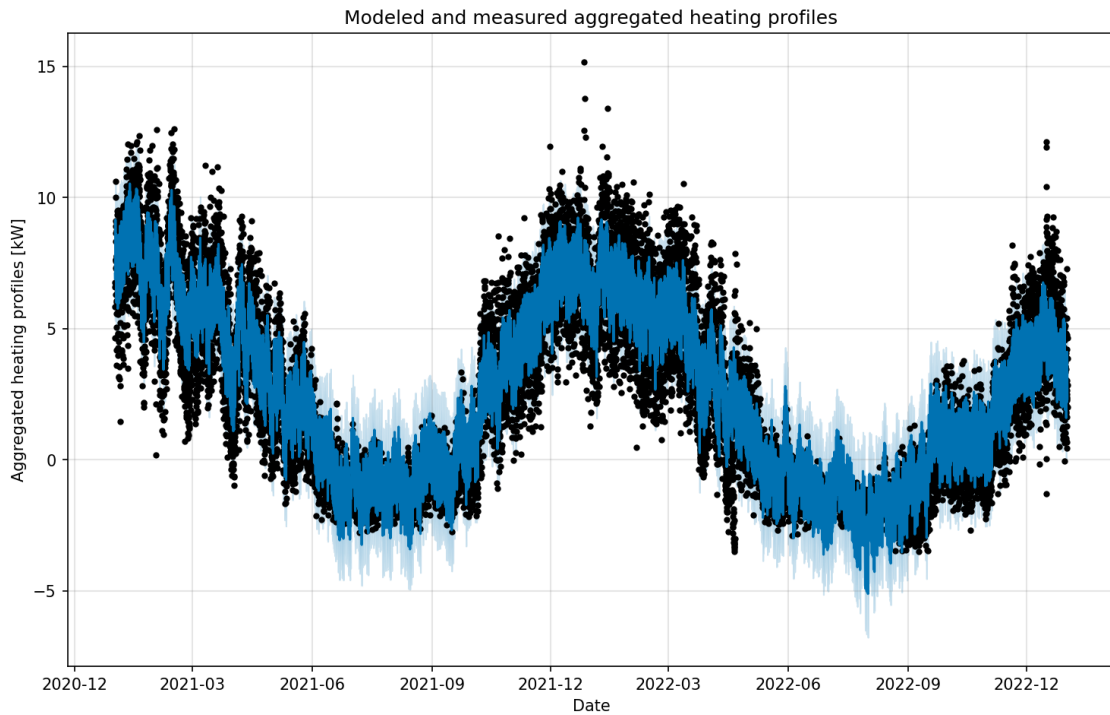


Figure 60: forecasted (blue curve) and measured (black dots) district heating profile

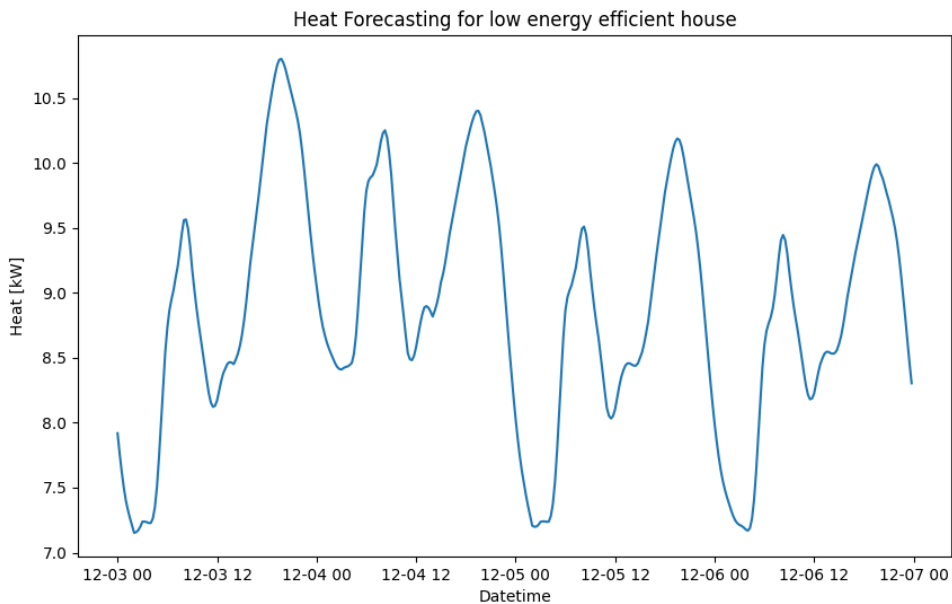


Figure 61: Daily forecasted heat profile of the house.



The TRL of the tool has increased from 4 to 5, as all models were created using the real values and the RESTful API has been completed. Integration into the overall system is now possible.

3.6.2 Documentation updates

The tool provides a RESTful API for accessing and interacting with forecasting data.

Endpoints:

5. Root Endpoint ("/"):

- Description: Provides information about the API and the date of the last change.
- HTTP Method: GET

6. List Models Endpoint ("/List"):

- Description: Lists available models for household profiles (smart meter data with disaggregation).
- HTTP Method: GET

7. List Heat Models Endpoint ("/List_heat"):

- Description: Lists available models for district heat household profiles.
- HTTP Method: GET

8. Forecasting Endpoint ("/forecasting"):

- Description: Generates forecasted user profiles (smart-meter profile) based on provided parameters such as longitude (number), latitude (number), pvgis (boolean) model (string) and time format (string).
- HTTP Method: POST

9. Historical Profiles Endpoint ("/historical"):

- Description: Generates user profiles (smart-meter profile) based on provided parameters such as longitude (number), latitude (number), model (string), date string(\$date) and PVGIS (boolean).
- HTTP Method: POST

10. Forecasting Heat Endpoint ("/forecasting_heat"):

- Description: Generates forecasted district heat household profiles based on provided parameters such as longitude (number), latitude (number), pvgis (boolean) model (string) and time format (string).
- HTTP Method: POST

11. Historical Heat Profiles Endpoint ("/historical_heat"):



- Description: Generates district heat household profiles based on provided parameters such as longitude (number), latitude (number), model (string), date string(\$date) and PVGIS (boolean).
- HTTP Method: POST API description/documentation updates: appendix document or link to a public repository

The documentation is published on <http://consensus.eu:20986/docs/>

3.6.3 Plan to integrate service into Data Space

As part of the integration plan for incorporating the Energy Usage Prediction Service into the Data Space, we will utilise the TNO TSG Connector as a standardised interface for secure data exchange. This connector will facilitate the interoperability between our service (which acts as a data source) and the data sink within the Data Space, Figure 62. Our integration approach will involve configuring the TSG Connector to ensure it aligns with the data formats and communication protocols required by the Energy Usage Prediction Service. We will establish authentication and authorization mechanisms, leveraging the Identity Provider to manage access controls. Adequate testing will be conducted to guarantee that the connector reliably transmits data while adhering to the strict privacy and security standards mandated in the Data Space.



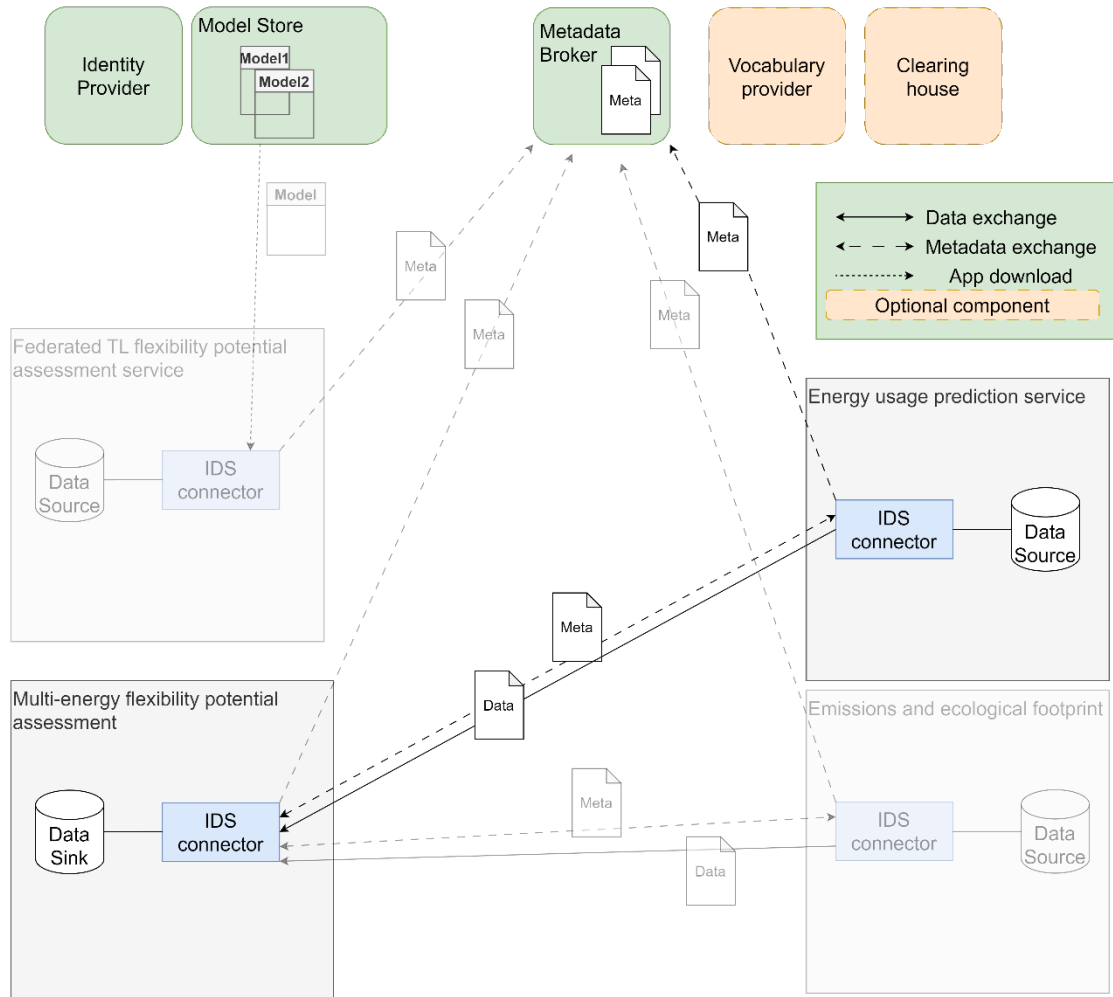


Figure 62: IDS Integration schema and possible interactions inside the pilot.

3.6.4 Next steps

The current models were based on the data sets received from ELCE and KPV. However, especially for the district heating model, for which only the aggregated profile is known, we will better adapt the heat usage profiles to the house's energy efficiency. Due to the lack of data, only linear scaling is applied for different types of houses according to its efficiency. Therefore, the next steps in the energy usage prediction service are related to additional model verification on different assumptions and data obtained to create better models. The final stage also involves integrating the service into the data space.



3.7 Service local load forecasts and estimation of electrical grid status

3.7.1 Development progress

The estimation of the electrical grid status is moved to TRL6, since it was tested and successfully run on ractualdata being recieved from the sensors on heat pumps , and the features of the services currently being presented to Pilot 3 DSO . The next step is to validate it by Pilot 3 DSO experts.

The grid simulator was extended to two tools with user-friendly graphical interface.

The first tool is the [ESDL MapEditor](#), which has been extended to allow one to create an ESDL energy system description by just dragging and dropping energy system components on a map. It can be used now as a front-end for:

- Heat network simulations
- Load flow simulations
- Gas network simulations
- Energy transition scenario simulations

At Figure 63 one can find a screenshot of the ESDL MapEditor on the topology of the grid for 34 houses in low-density populated geographical area to give an impression of its user interface and possibilities. It is compatible with Open Street, Google Streets and Google Satellite maps to define the needed area according to the actual geographical area information from the DSO. It allows to define all assets existing in the grid according to ESDL standard ([Energy System structure - ESDL \(gitbook.io\)](#)).



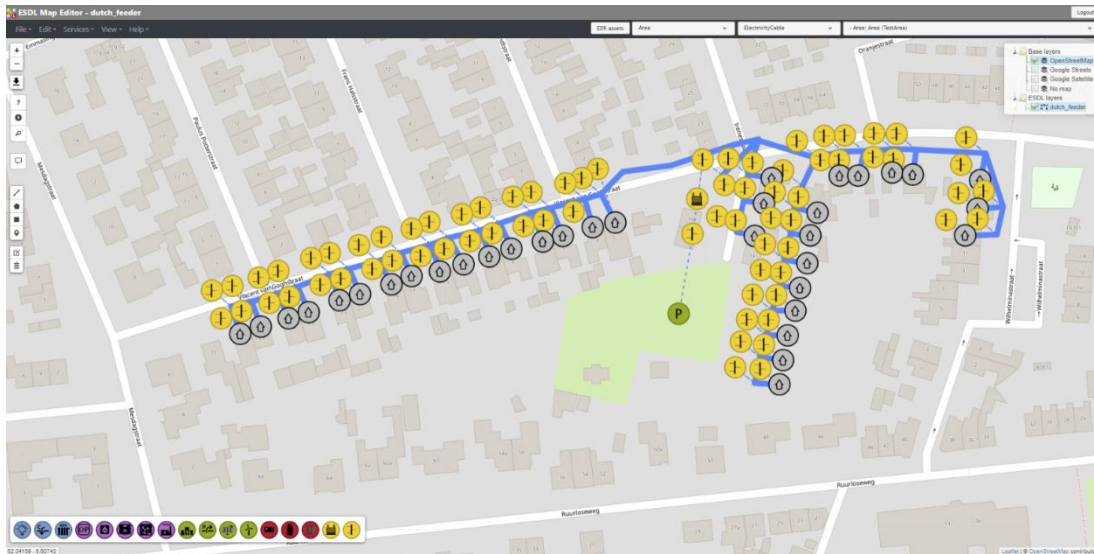


Figure 63: A graphical interface of the energy grid MapEditor with the example of the topology from the low-density populated geographical area

By clicking on an asset, the tool allows you to edit the properties of any energy asset. Right click on an asset, gives you functionality like connecting assets, setting the energy carrier of the individual ports, attaching profiles to a port, and setting marginal costs.

Given the topology and properties of the assets, the simulation of the grid can be run by the Energy System simulator.

The second tool is the Energy System Simulator, which simulates network balancing and the effects thereof, in an interconnected hybrid energy system (hybrid – since it can include wind mills and other energy sources) over a period of time. It takes as inputs the energy system defined in ESDL and the list of the observations/predictions from the households. It is able to run the simulation and define whether the specific transformer is overloaded. Additionally, it is able to calculate an optimal schedule of flexible producers and the effect of this schedule in terms of emissions, costs, load on the network, etc. The user chooses the optimization parameters from the menu at interface before starting the simulation, if needed.

At the heart of the tool are:

- An algorithm to determine the order of energy delivery operations from energy production till the household,
- A tree-based solver that calculates the load on transformers and end-points for households based on the demand-supply solution determined above.



The input predictions on the energy consumption for every involved household could be given directly to the tool by the human user as the csv file, or could be provided by the local predictive Federated Learning agents (see the section 2.2 for the prediction models for local clients). The simulation tool outputs the prediction on the transformer overloading for the next 24 hours during the course of the simulation into an Influx open-source database.

3.7.2 Documentation

The documentation of the Map Editor tool for uploading the topology and define the links between the household, transformers and energy repositories is now available for public access in the following link: [ESDL MapEditor - ESDL \(gitbook.io\)](#).

The documentation to the Energy Grid simulator compatible with ESDL standard is available in the following link: [Energy System Simulator \(ESSIM\) - ESDL \(gitbook.io\)](#).

3.7.3 Integration with the Data Space

Our integration strategy with the Data Spaces relies on two components from Data Spaces: DS Connector and DS registry. DS registry contains the description of the service and its functionality. It allows user to find the service and to understand what service does and which outcome will be received. DS-Connector component is used to actually run the service. The TNO TSG Connector with its renewed protocol is used as a standardised basis interface to start the FL platform and to facilitate data exchange. This connector is pivotal in fostering interoperability between our Federated Learning service, functioning as a data source, and the data sink for the resulting predictions on overloading situated within the Data Space (refer to Figure 64).



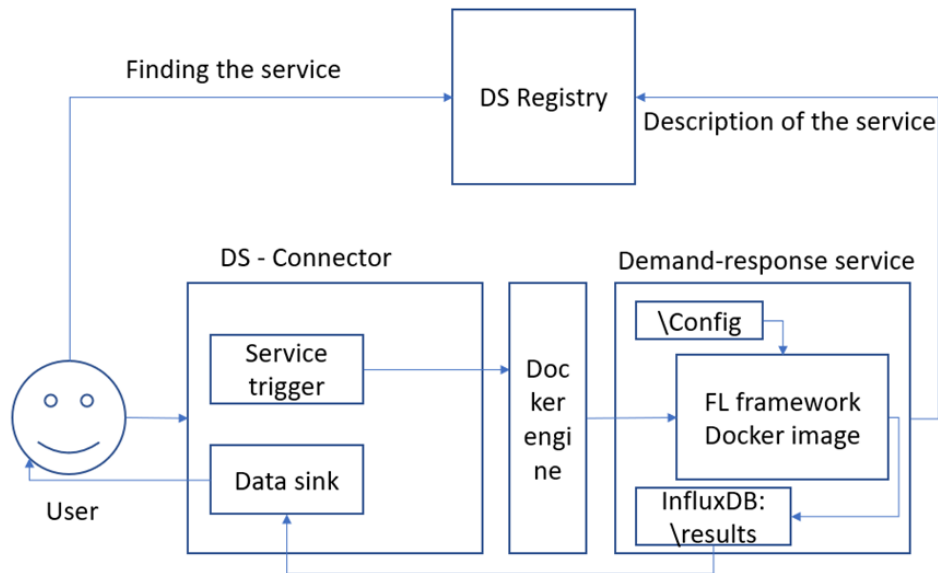


Figure 64: Architecture of the integration of the Federated Learning platform and IDS connectors from different organizations

It starts with the user (we expect an expert from DSO) who has decided that she/he wants to use the local load forecast service, found at DS Registry. Inside the service is based on the predictions on energy usage from the households, defined by user and given as the configuration parameter to receive back the predictions on the electrical transformer in the specific defined area.

Our integration methodology entails that the user uses IDS Connector which is able to start a Dockerized FL framework by sending the Docker commands to the Docker engine to start the specific service Docker image. TSG Connector starts the FL plug-in, which initiates the FL platform and both services for local predictions and global energy grid simulator model. The solution described in Section 2.2 represents the FL framework. The results are written done to the database and are sent to the TSG Connector from which the request for service came. TSG connector is used not only to run the service, but also to ensure its congruence with the prescribed data formats and communication protocols mandated by the organization provisioning data and using IDS Connector for data exchange.

3.7.4 Next steps

Implementation of the integration architecture described in the section above of the service local load forecasts and estimation of the electrical grid status with the TSG connector. For now the results are written to the InfluxDB, and they are not provided back to user through service



graphical API. Implementation of such visualisation of the results on the transformer overloading graphically to user is foreseen for the next deliverable.

3.8 Flexibility Analytics and Register (FAR) service

The Flexibility Analytics and Register Service (FAR) intends to provide a combined service and tool that from one side allows for the registration of assets, their portfolio management into resource groups featuring both programmatical and user interfaces data ingestion. The main objective of the FAR service is promoting the significance of sharing (anonymized) flexibility data and characteristic with an analytics service to retrieve thereafter knowledge on the current state of the grid (upcoming flexibility needs) and installed flexibility in the grid at different temporal stages.

3.8.1 Development progress

At this cycle of reporting, the FAR service has been updated to adopt and accommodate the Slovenian pilot's needs. The initial documentation provided in the D6.1 for FAR has considered that all data exchanges would rely on XML-based Common information model (CIM) European style market profile (ESMP) standard profiles. As the remainder demo services are based on custom output profiles CSV and JSON formats, adaptation have been developed to ingest such data.

A major improvement has been the addition of the role of the Flexibility Service Provider (FSP). To implement this there is a consent management interface which maintains the contracts of the represented resources. The current TRL of the service is considered at TRL6 as already tests and necessary API interfaces are released to allow interaction with actual user or FSP platforms.

3.8.2 Documentation updates

The latest updates on the APIs documentation is presented on the [link](#). The current version also reports UI features that are also presented in the Appendix.

3.8.3 Tests of the services with the Data Space

Within this stage the FAR service and its exploitation from grid users (residential and end-users) has been integrated with an energy data space utilizing the [OneNet \(H2020 project\) connector](#), using the architecture depicted in Figure 65. More specifically, by deploying locally at end-users' premises an OneNet connector that can discover the FAR service and interact with it upon an agreement. The integration with the connector APIs has been performed via the Register UI tool which is thereafter responsible to interact with the connectors API. Through the OneNet



connector local App there is need to create a subscription to the FAR service to allow end-to-end data exchanges.

An architectural difference is that the OneNet connector relies on the middleware meta-data sink which is essentially responsible for the discovery of data provider, consumer and service providers. Note that the figure below shows the detailed components on how the end-users exploit via the OneNet connector the analytics from FAR. It is important to mention that this service analytics rely on the principle that multiple energy stakeholders share data for a common goal, to get access on flexibility analytics, which in turn provides them insights for participation in flexibility or ancillary markets. The analytical discussion on the integration is provided in the Appendix.

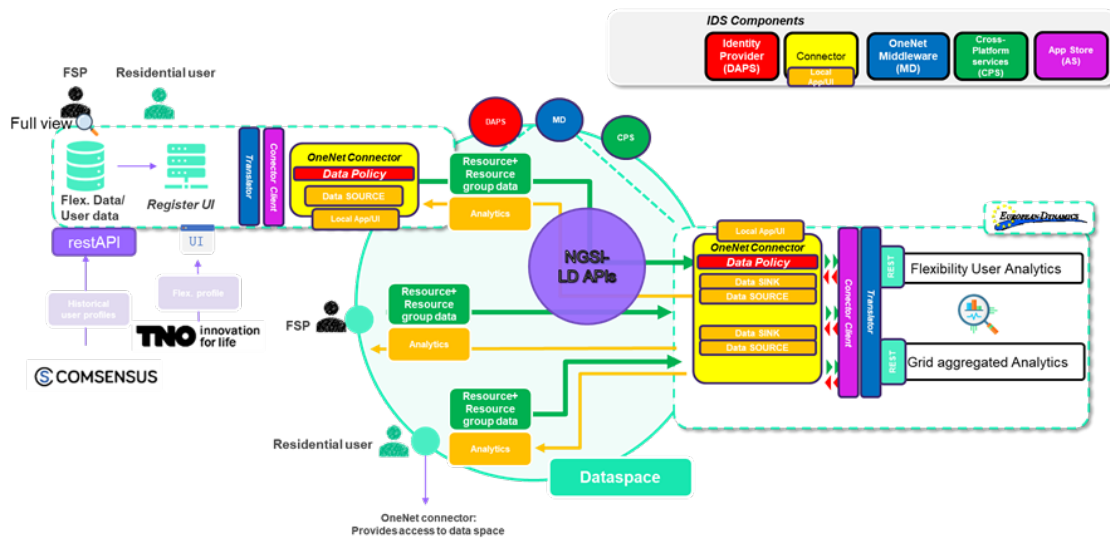


Figure 65: Architecture of the integration of the FAR with Enershare data space using OneNet connector (NGSI-LD APIs)

3.8.4 Next steps

The next steps will focus on the improvement of the analytics, exploring the options to provide locational characteristics to the end-users. The next software cycle will consider the finalization of all the list of flexibility and grid analytics for the users, fact which will allow the preparation of the visualisation dashboards at Flexibility Register User Interface level. Different scenarios will be demonstrated with multiple FSPs managing different types of resources/assets (PV, BESS, EV, heating flexibility) showcasing the importance of the analytics. Finally, the integration with COP service via the data space ecosystem to allow cross-energy flexibility sharing will be highlighted, validating the importance of data and analytics sharing.



3.9 Aggregation of flexibility from end users

3.9.1 Development progress

3.9.1.1 Overview of developments

In D6.1, a first version of a service that aggregates residential EVSEs and controls them to provide Demand Response services was developed. In this initial version, the Demand Response service connects to an EV/EVSE aggregator, e.g. Hiven¹ - the internal Fortum startup on whose behalf this service is being developed for - and based on (simulated) grid frequency measurements, attenuates the charging rate of individual EVs/EVSEs by an appropriate amount computed by a disaggregation algorithm, and then sends the calculated charging power setpoints back to the aggregator to be forwarded to end-user devices. In what is to follow, the new developments since D6.1 will be outlined.

The disaggregation algorithm has received improvements over the version present in D6.1. In short, the differences from the previous deliverable can be summarized in the following points:

- **Improved fairness:** To minimize sent number of commands, the previous version of the algorithm was heavily biased towards chargers with large flexibility, i.e., the difference between maximum and minimum power. The updated version is much fairer to the user since it cycles through the chargers in order of earliest activation. In other words, the new algorithm will always prefer to issue a command to the charger that has not been used in the longest time (or ever). This ensures that all chargers are used evenly, and any biases are eliminated.
- **Robustness to real-life conditions:** Since a command is not executed immediately upon receipt from a charger device, the steering algorithm now implements optimistic caching: when a command is sent to a charger, the local view of the charger is updated as if the command is applied immediately. This prevents resending the same command to the same charger and in general allows for better handling of delayed telemetries. In addition to this, the round-robin queue mentioned before also helps since for each charger, sent commands are spaced as far apart as possible, thereby giving ample time to chargers to implement commands. Finally, the algorithm has been tuned to undershoot instead of overshoot (even when it is allowed by the rules), as a less aggressive response has been shown to be more reliable and exhibit less artifacts.

¹<https://hiven.energy>



- Adaptations to the new SVK rules:** The rules under which a potential FCR provider would be prequalified for provision in the Nordics have changed, as of September 1st, 2023. This required retuning various components of the algorithm, as the new requirements have much stricter latency requirements. This required revisiting the whole service pipeline and reducing imposed latencies in each component when possible. Although progress has been made this work is still unfinished, and similar improvements will also be featured in the next deliverable.

In addition to the above, significant development effort has been placed in fixing various bugs and improving testing. To facilitate this, as mentioned on the previous deliverable, an accompanying simulator has also been developed, which has received various improvements in this deliverable. Specifically, the updated version of the simulator is now more configurable, allowing for tests scenarios to be reproducible by being deterministically specified from a configuration file. Additionally, there is more fine-grained control and simulation of network and actuation latencies and dropout rates. The simulator can be used in an offline setting but can also be deployed in cloud instance allowing for more realistic test conditions.

3.9.1.2 New subsystems

The improvements mentioned above have been enabled by the development of new components and subsystems. This section aims to provide further detail into these components, and provide a high-level view of the updated disaggregation algorithm itself. An overview of the components and how they are interconnected is given in Figure 66.

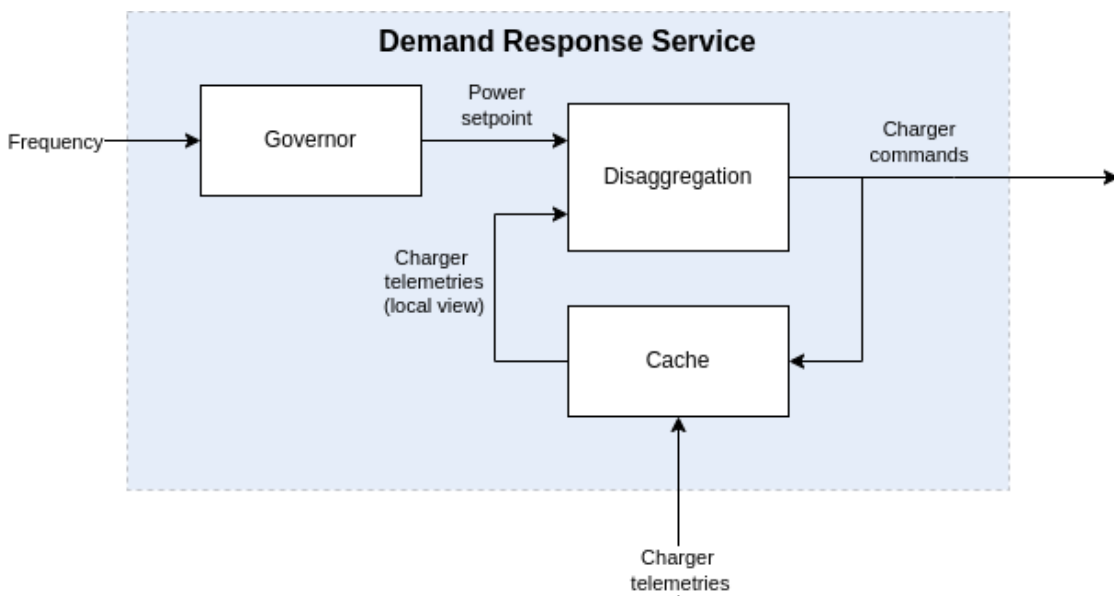


Figure 66: Overview of components comprising the Demand Response Service**Governor**

This component is responsible for converting the measured grid frequency into power setpoints for the disaggregation algorithm. Depending on the FCR product (or any combination thereof) the aggregator has opted for at any given time, this component calculates to what aggregate power setpoint the charger fleet should be steered to. This component includes the required frequency filters that guarantee the stability and performance of the response towards the grid, as required by all Dynamic FCR products (FCR-N and FCR-D), as well as frequency filters that are required to ensure smooth activation and deactivation of the response for Static FCR-D products. Additionally, should the aggregator opt for multi-market participation, e.g., by combining FCR-N and FCR-D, the governor is responsible for determining the composite setpoint for the disaggregation algorithm.

Round-Robin Selection Queue

As mentioned before, instead of sorting chargers in terms of flexibility, the algorithm now first makes sure that all chargers in the fleet have been sent a command before resending a command to the same charger. In the case where there are no chargers that have not been commanded, the queue prefers the charger that received a command the earliest. This allows ample time between successive commands to the same charger, allowing for a more reliable and deterministic charger behaviour, as this selection queue ensures that when there is a change in charger telemetry it is due to a known command. From empirical testing, this component improved the smoothness of the fleet response but made it less reactive to abrupt changes, improving overall response stability.

Local Command Cache

This component keeps a local view/copy of each charger telemetry and updates it when a command is issued to a specific charger device. Specifically, the cache is optimistic regarding command execution and updates the local charger telemetry view as if the command sent is immediately implemented by the device. This is necessary because command execution is not instant from the charger side, and even when a command is implemented, there is an additional delay until the new telemetry reaches the algorithm itself. Without caching, the fleet response would be problematic, as the disaggregation algorithm could repeat the same command to a charger or even request more activation than required from the fleet. Nevertheless, occasional instances arise wherein the charger becomes unresponsive, or a command is dropped, whether due to a poor network connection or other factors. To mitigate such situations, each cache element is endowed with a configurable expiration time. If this time elapses without the true telemetry being updated to reflect the execution of a sent command, the charger telemetry is considered outdated and is excluded from the algorithm.



Disaggregation Algorithm

As previously stated, the disaggregation algorithm has undergone updates since the last deliverable, by incorporating the Round-Robin Queue mentioned earlier and being tuned to consistently undershoot. This adjustment is aimed at preventing any undesirable consequences, as it was noted that even the slightest overshoot could result in chattering behaviour. This phenomenon manifests when commands to activate and deactivate chargers are generated rapidly near an activation threshold. As mentioned before, while this approach does not minimize number of commands, nor activation error, it is improved on the fairness axis, since in the updated version all chargers of the fleet will be used equally.

A pseudocode description of the updated algorithm can be found below:

Disaggregation Algorithm

01. **Input:** Power setpoint P_t^{set} ; Charger List C ; Import, Minimum, and Maximum power for each charger $\{(P_{c,t}^{\text{im}}, P_c^{\text{min}}, P_c^{\text{max}})\}_{c \in C}$; Time of last sent command for each charger $\{t_c^{\text{cm}}\}_{c \in C}$; Current time t
 02. Calculate total aggregate consumption $P_t^{\text{im}} \leftarrow \sum P_{c,t}^{\text{im}}$
 03. Calculate power deficit $\Delta P_t \leftarrow P_t^{\text{set}} - P_t^{\text{im}}$
 04. Initialize command list $U_t \leftarrow \{\emptyset\}$
 05. **While** True :
 06. **If** $\Delta P_t > 0$:
 07. Get oldest-commanded charger $c^* \leftarrow \operatorname{argmax}_{c \in C} \{t - t_c^{\text{cm}} : P_{c,t}^{\text{im}} = P_c^{\text{min}}\}$
 08. **If** $\Delta P_t - P_{c^*,t} + P_{c^*}^{\text{max}} \geq 0$:
 09. Prepare command for selected charger $u_{c^*,t} \leftarrow P_{c^*}^{\text{max}}$
 10. Append command to command list $U_t \leftarrow U_t \cup \{u_{c^*,t}\}$
 11. **Else:**
 12. **Break**
 13. **Else:**
 14. Get oldest-commanded charger $c^* \leftarrow \operatorname{argmax}_{c \in C} \{t - t_c^{\text{cm}} : P_{c,t}^{\text{im}} = P_c^{\text{max}}\}$
 15. **If** $\Delta P_t - P_{c^*,t} + P_{c^*}^{\text{min}} \leq 0$:
 16. Prepare command for selected charger $u_{c^*,t} \leftarrow P_{c^*}^{\text{min}}$
 17. Append command to command list $U_t \leftarrow U_t \cup \{u_{c^*,t}\}$
 18. **Else:**
 19. **Break**
 20. **Return:** command list U_t
-

It should be noted that, while not documented in the above pseudocode, the commands are forwarded to the Cache at the same as they are sent out to chargers, so the times of the last issued commands for each charger are updated. Furthermore, the algorithm is also fed power



measurements for each charger from the cached version of the charger telemetries and is not interfaced directly with the true telemetries. As mentioned before, the power setpoint input is calculated from the Governor, it is reflecting any potential composition of multiple products, and is additionally the output of the stability filters present in the Governor.

3.9.1.3 Technology Readiness Level

We classify this work as TRL 6. The previous deliverable was classified as TRL 5 but given the number of improvements to the algorithm to handle real-life constraints and situations, as well as further testing with real charger devices, an increase in the readiness level is warranted. Specifically, during the time between D6.1 and D6.2, various successful end-to-end tests with multiple real-charger devices have been performed, but not yet enough to have a prequalification-like test that would justify a TRL 7.

Additionally, a screenshot from the developed simulator is included in Figure 67, where the overall response of the various components is depicted. The flexibility bid size corresponds to the minimum size of 100 kW, so the charger fleet is over-provisioned. The top plot refers to the view from the simulator perspective, i.e., (simulated) ground truth. The fleet view from the algorithm is seen at the second plot, whereas the third plot indicates the requested (by the governor) flexibility activation and commanded activation as issued by the disaggregation algorithm. Finally, the bottom plot depicts the input frequency signal, along with the activation zones for different FCR products.



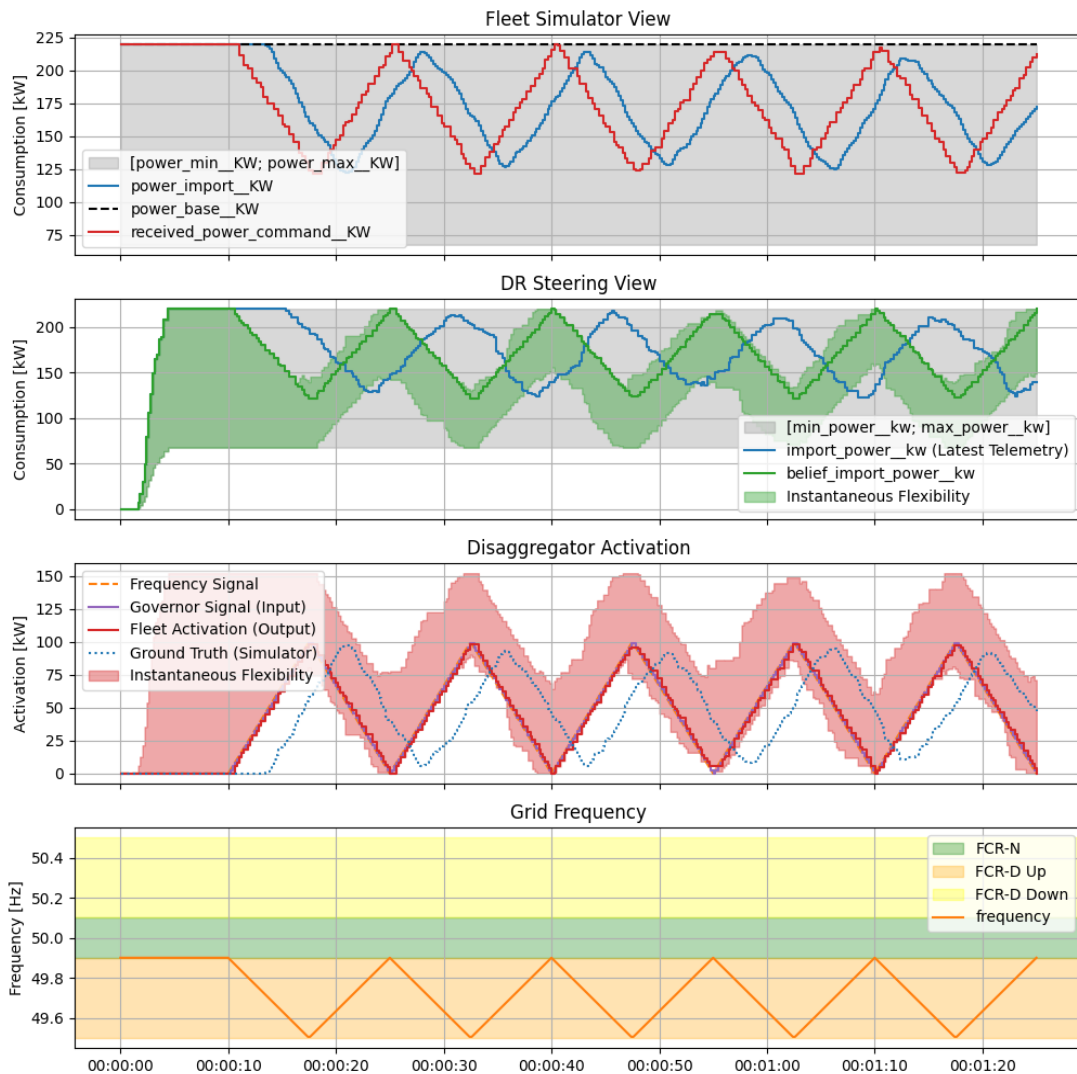


Figure 67: Simulated response to a triangular frequency signal of a charger fleet of 67 units with baseline consumption of 225 kW and a flexibility potential of 150kW

3.9.2 Documentation updates

The data schema used has not changed from the last deliverable. The API of the Service under development shall not be documented here, as it is an interim one that will change upon successful Data Space integration. Additionally, the current communication paradigm of the Service will change, since now it is a Client to an internal Server, but upon Data Space integration, the Service itself will be the Server. Further details will be provided in the next Section, where different APIs will be presented according to the integration options available from the connector capabilities.



3.9.3 Data Space integration plan

Depending on connector capabilities and technical limitations we have two plans in mind, one with a RESTful implementation and a stream-based one using WebSockets. Both plans will be explored in what follows.

3.9.3.1 RESTful Data Space integration

In Figure 68 a potential plan is presented, outlining two potential options for integrating with the Data Space in a RESTful manner: one where the frequency measurements required for FCR originate from the Service User (Use-case 1), and one where they are provided to the Service through an external-to-the-data-space source (Use-case 2). Both options are presented as there has been no frequency source integrated to the Service yet, so it's unclear what technical limitations in terms of latency it would impose. In the first case, the Client would receive the charger commands as a response to each frequency measurement they sent, but in case this setup is deemed infeasible due to either technical requirements on the frequency measurements, or due to it not being convenient as a business case for the Client to have a frequency measurement device of their own, the required setup will need to follow a setup akin to that of Use-case 2. In this scenario, the charger commands to be forwarded will need to be sent through an external connection to the Data Space, e.g., a Webhook as depicted in the diagram. In both scenarios, the endpoints are the following:

- **/static_config:** needs to be called once by the Client to provide bid size, fleet characteristics and schedules, and other relevant information. Such an endpoint does not yet exist nor is there anything equivalent in the Service; instead configuration is provided by either environment variables or static configuration files.
- **/telemetries:** This endpoint will expect to receive DeviceTelemetry messages (c.f. D6.1) on frequent update intervals (e.g., every 1s).
- **/frequencies:** This endpoint will expect to receive FrequencyEvent (c.f. D6.1) messages. In Use-case 1 the response will be a list of ChargerSetCurrentLimitCommand (c.f. D6.1) messages. In Use-case 2, this endpoint will not be exposed to Data Space, but instead use an external-to-the-data-space source for performance considerations.

Since the Service relies on high-frequency communication, the REST implementation might suffer from significant limitations, as HTTP calls were not designed for such a purpose. A stream-based approach might be a more appropriate design choice. The communication between the connector and the Service is still also under investigation, but some kind of high-performance channel, such as Intra-Process Communication might be required.



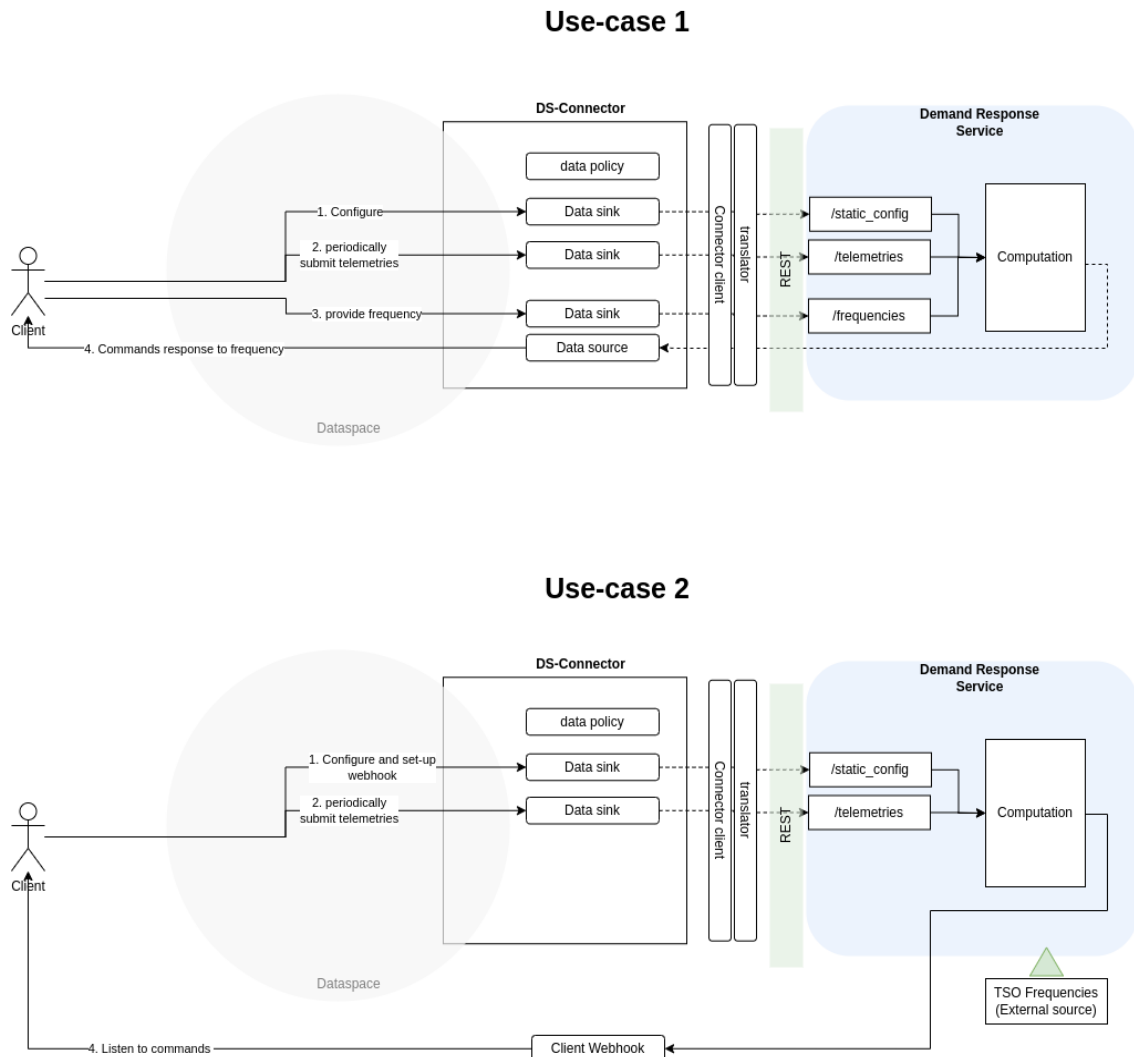


Figure 68: RESTful Data Space integration plan for the aggregation of flexibility from end-users service

3.9.3.2 Stream-based Data Space integration

In Figure 69, an alternative, more preferable, plan is presented that is based on streams. Use case 1 refers to a scenario where the Client provides frequency measurements, along with all the other required information, through a Websocket-based communication channel, from which they also receive the commands to be forwarded to their Chargers. In Use-case 2, akin to the RESTful implementation, the frequency measurements are provided from an external source, but due to the bidirectional nature of the Websocket connection, the Client still receives charger commands through the same channel they sent telemetry and configuration data from.



Specifically, the Client initiates a bidirectional Websocket-based communication channel, through which they can exchange messages and data with the Service in a fast and efficient manner. Both use-cases are presented as before: frequency messages being transmitted to the service through the Data Space and outside of it. A significant difference with the REST-based implementation is that moving the frequency source out of the Data Space does not impact the overall architecture, since the computed commands are still forwarded to the Client by the bidirectional Websocket stream. In the stream-based approach there are no distinct endpoints, but instead the same messages as before, i.e., the static configuration, DeviceTelemetry, FrequencyEvent, and ChargerSetCurrentLimitCommand, are sent through the same connection/channel. Furthermore, another advantage of this approach is that it is a more performant one, as there is no overhead imposed by the HTTP protocol for instance.

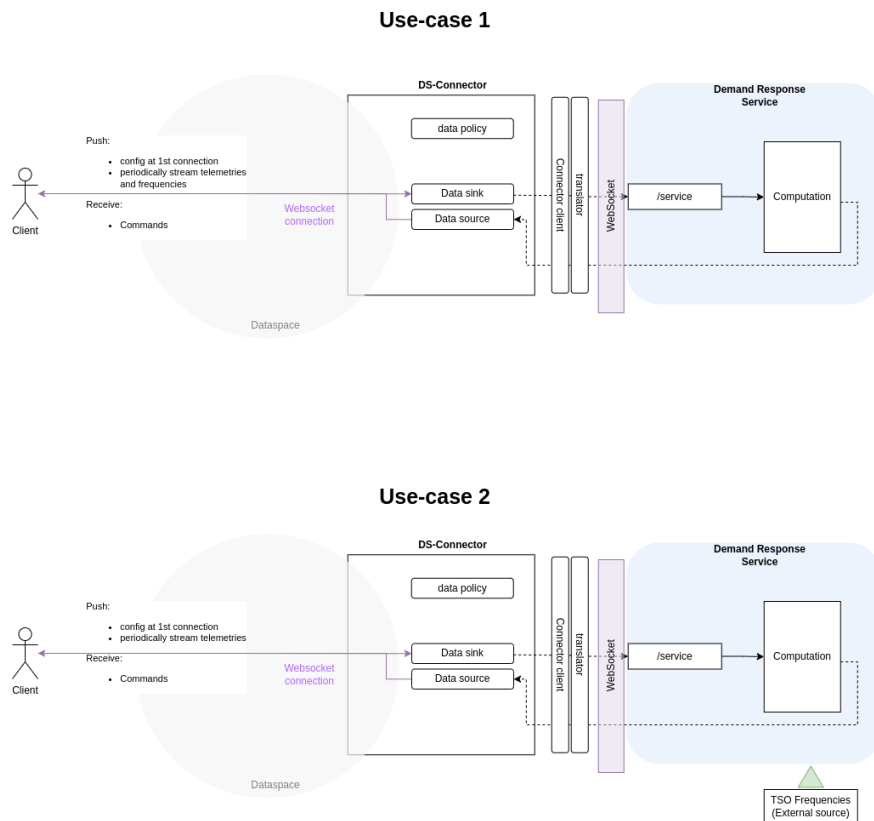


Figure 69: Websocket-based Data Space integration plan for the aggregation of flexibility from end-users service

3.9.4 Next steps

In this deliverable, some potential plans were outlined for integrating with the ENERSHARE Data Space. However, these are tentative and subject to change upon receiving new information



regarding the capabilities of connectors and their technical limitations. In the next deliverable we aim for complete integration to the Data Space.

Furthermore, the algorithm development of the service is almost complete, but there could be further improvements in the final deliverable. Finally, we aim for extended end-to-end tests with a whole EV fleet and a frequency measurement device. When these are in place prequalification of the service with SVK will be attempted, which if successful will deem this service at TRL 7+.

3.10 Data-driven management of surplus RES generation in distribution systems

3.10.1 Description of the service

Renewable energy sources (RES), such as rooftop PVs, are becoming increasingly popular in the distribution networks due to their climate friendly nature. It has been pointed out that the large-scale integration of these RES influences the customer load profile and introduce uncertainty to the distribution system's net load especially in residential areas. Because such resources are typically located behind the meter (BTM), it becomes necessary for the distribution systems operators to accurately predict the BTM load and PV power for several time steps ahead for smooth operation of the distribution network. The resulting surplus generation from such resources can be utilized for other purposes such as pumping of water into a reservoir or EV charging needs.

In this regard, this service aims to provide a data-driven solution for managing/optimizing RES surplus in the distribution network. The service consists of two parts. The first part, which is a subset of the second part is a machine learning module that dis-aggregates net-load into its component parts: the RES (e.g. PV generation) and load. After the disaggregation process is completed, the values of PV generation and load are predicted for several time steps ahead. Here, we employ spatio-temporal graph learning approach to automatically dis-aggregate historical net-loads of residential housing units into their respective BTM load and RES generation component and perform multi-step prediction of their values. The second part is the application module. This module realizes the optimization objective of the service which is to determine the most effective way to utilize the surplus generation (re-purpose it for other) and consequently minimize negative effects of RES integration such as reverse power flow. It uses predictions from the machine learning models as input in addition to some extra data for optimal scheduling of the surplus RES. It can be considered as superset consisting of the first part (machine learning module) plus extra components (extra data for instance, scheduling



preference, real-time water system or EV data). This can be regarded as the complete application.

The service will be validated in Pilot 5 (Italy). However, the current development was carried out using the benchmark dataset as we are currently in the process of receiving the pilot data. Figure 70 shows the data used in the current implementation stage of the service while Figure 71 shows a comparison between the actual data and the predictions by the multivariate spatio-temporal graph model trained for the service.

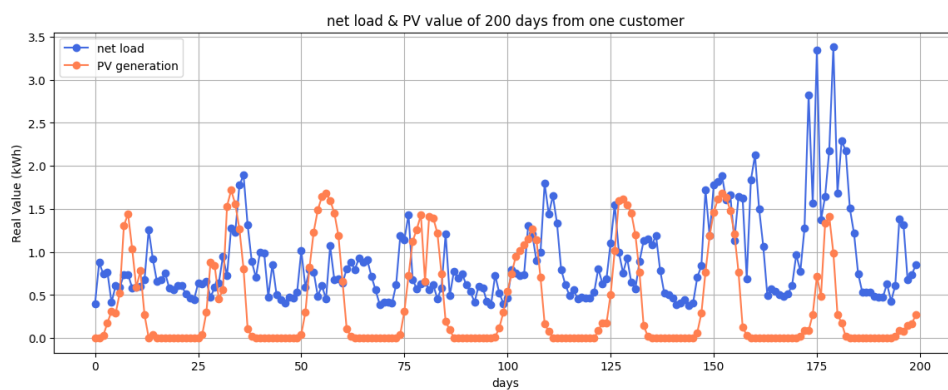


Figure 70: Sample net load and PV generation data

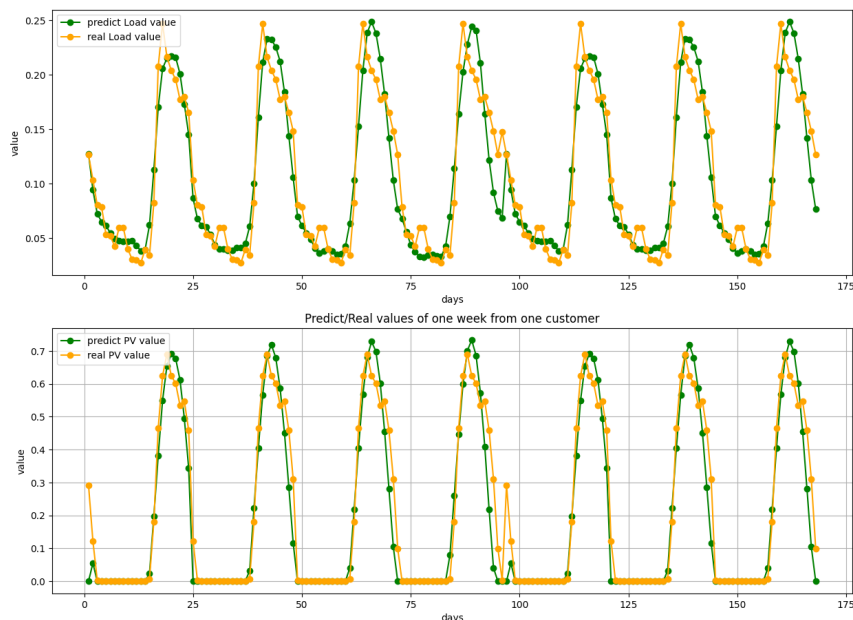


Figure 71: Comparison between the actual data and the predictions by the multivariate spatio-temporal graph model



3.10.2 Innovation

The service provides the ability to automatically dis-aggregate and jointly learn both load and BTM RES generation for multiple housing units in a multi-step fashion by using the net-load and weather information. The service also makes it possible to either specify or automatically construct multivariate spatio-temporal graph representation of the housing units considered based on the correlation between their net-loads. By adopting a graph approach, it is possible to not only estimate surplus RES generation but to also know at which parts of the grid they will be injected. This can be helpful, for instance, when trying to prevent reverse power flow.

3.10.3 Functions

The service consists of two parts. The first part represents activities involving the use of historic and weather data up to the time communication with the broker (e.g., MQTT) is established. We refer to this part as the machine learning module while we refer to the second part involving the use of real-time data from as the application module.

3.10.3.1 Machine Learning Module

This module includes functionalities that realizes data handling, model training and visualisation of data and model prediction results. Figure 72 shows an UML diagram of this module with explanation of its key components provided hereunder.



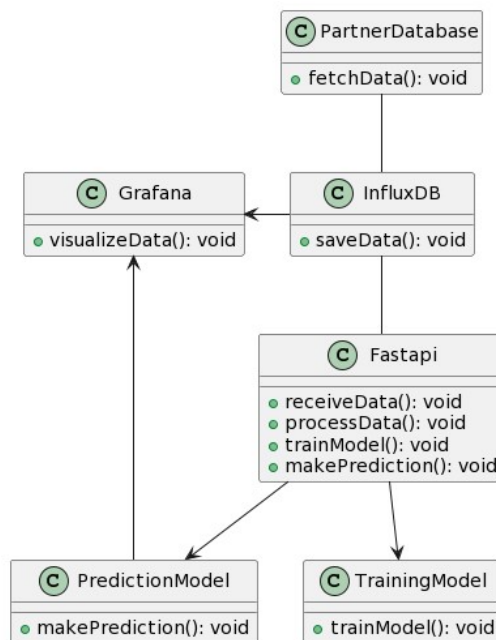


Figure 72: Machine learning module of data-driven service for managing surplus renewable energy generation in distribution systems

Partner DB: Historic data are stored in databases such as MySQL. The partner (pilot) provides credentials to access the DB.

InfluxDB: This is a timeseries database where the original data from the partner's database is stored after for use by the service components. Preprocessed data is also stored in this database.

Fastapi: This is an asynchronous python-based framework to create APIs. Here the APIs will make a connection to the influxDB. Particularly, server-side events can be applied to make the connection to influxDB.

Prediction Model: The model predicts the future outcomes based on the trained data. For this reason, it has to get data of interest from influxDB as an input. To do so relevant API's has to be called.

Grafana: The tool to visualize the predicted output from the model.

Data pipeline: There is a historic database with historic data stored in MySQL that can be accessed using login details provided by the pilot owner. Data from this database is read via telegraph and written into the influxDB. An API request can be performed using the FastAPI to



get the data from the influxDB in the desired format for the machine learning model. The served model gives predictions based on the input data; the predicted data is written back to the influxDB. Further, the predicated values are shown in the grafana together with the input data (the data which is passed to the machine learning prediction model). To be more specific, the data flow for model training and inference are as follows.

(i) Model training: Mysql → Telegraph (Mysql plugin, Influx DB) → Influx DB → Fastapi → Training Model and

(ii) Inference: Mysql → Telegraph (SQL plug in, Influx DB plugin) → Influx DB → Fastapi → Prediction Model → Grafana and Influx DB

3.10.3.2 Application Module

This module includes functionalities that realizes real-time data handling, CI/CD pipeline, model serving, visualisation of data and model prediction results. It also includes the application component which is where the optimization of surplus RES for water pumping into the reservoir or EV charging is realized. Figure 73 shows a UML diagram of this module with brief explanation of its key components provided hereunder.



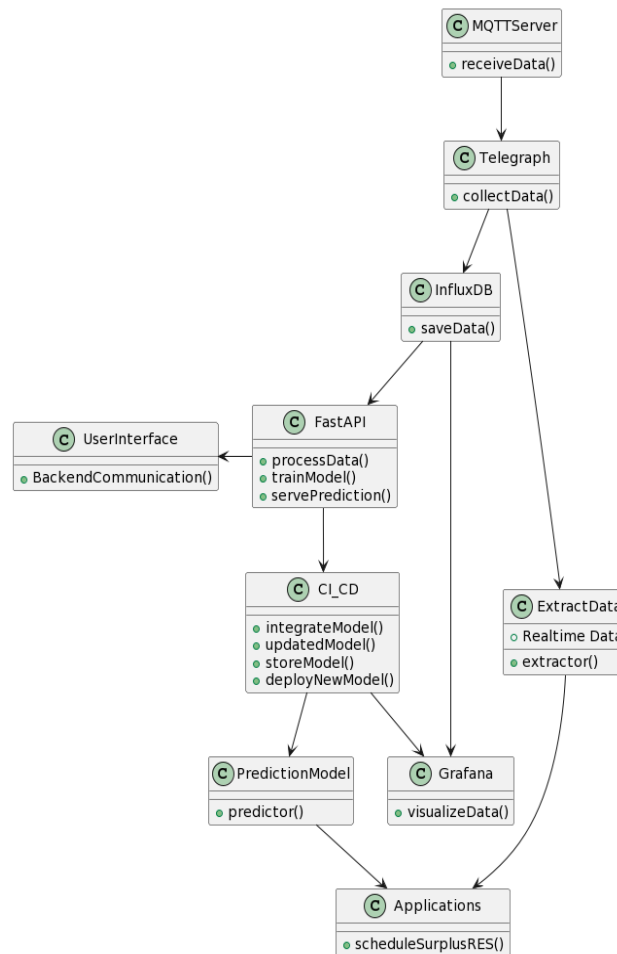


Figure 73: Application module component of data-driven service for managing surplus renewable energy generation in distribution systems

MQTT server: The real-time data can be accessed from the MQTT server with credentials provided by partners.

Telegraph: The telegraph is used to enable connection between the influxDB and MQTT broker. The MQTT plugin is installed in the telegraph to facilitate the operation.

CI /CD Pipeline: We set up a CI/CD pipeline to ease incremental improvement of the service as relevant data are not fully available currently but will be continuously collected in incremental manner in addition to changes in the service functionalities/requirements. For instance, full versioning of the models as well as the datasets used in their training are tracked and improved models can be automatically deployed for the service with the aid of the CI/CD pipeline. The same is the case for the entire application as incremental improvement in functionalities can be easily integrated.



The dataflow for the service components is as follows.

MQTT Server (External) → Telegraph (MQTT plugin, Influx DB Plugin) → Influx DB → Fastapi (Websockets) → Training Model → Frequent Update of the Prediction Model (CI/CD approach) → Graphana and Influx DB

Fastapi: In the case of historic data it's better to do the server side events to get the data from the influx db to training model but in the case of real time data the communication protocol will be websockets between influx db and fastapi.

UserInterface: The UserInterface shall coordinate the action triggered by the user.

Deployment: The service contains a docker compose file that can setup each service components to run as docker containers easy deployment. The inclusion of a CI/CD pipeline has to perform testing and deployment of updates in an incremental fashion.

3.10.4 Input and Output Data Formats

The input to the service is a tabular data containing the net-load, PV generation at various housing units and weather data such as temperature and irradiation. The output data include the multi-step prediction results in JSON format and optimization result for EV charging schedule and pumping of water into the reservoir. This can be retrieved in JSON or CSV format.

3.10.5 Integration with the Data Space

The service is currently not yet integrated with the ENERSHARE Data Space as the development started recently due to delay in defining the service and the associated pilot. The service is planned to be integrated into the ENERSHARE Data Space.

The TSG Connector will be utilized in the integration of the service to the Data Space. The connector will enable interoperability between our service, *Data-driven service for managing surplus renewable energy generation in the distribution systems* and the Data space as presented in Figure 74. This integration will require configuration of the TSG Connectors to ensure smooth and secure exchange of input and output data between our service and the associated clients via the Data Space. Thus, the input data from the clients is received with the aid of the Data Space connector. Similarly, the clients also can only receive the output data (results) from the service by first establishing a connection via the DS Connection. Evaluation tests will be conducted to verify the effectiveness of the proposed integration approach.



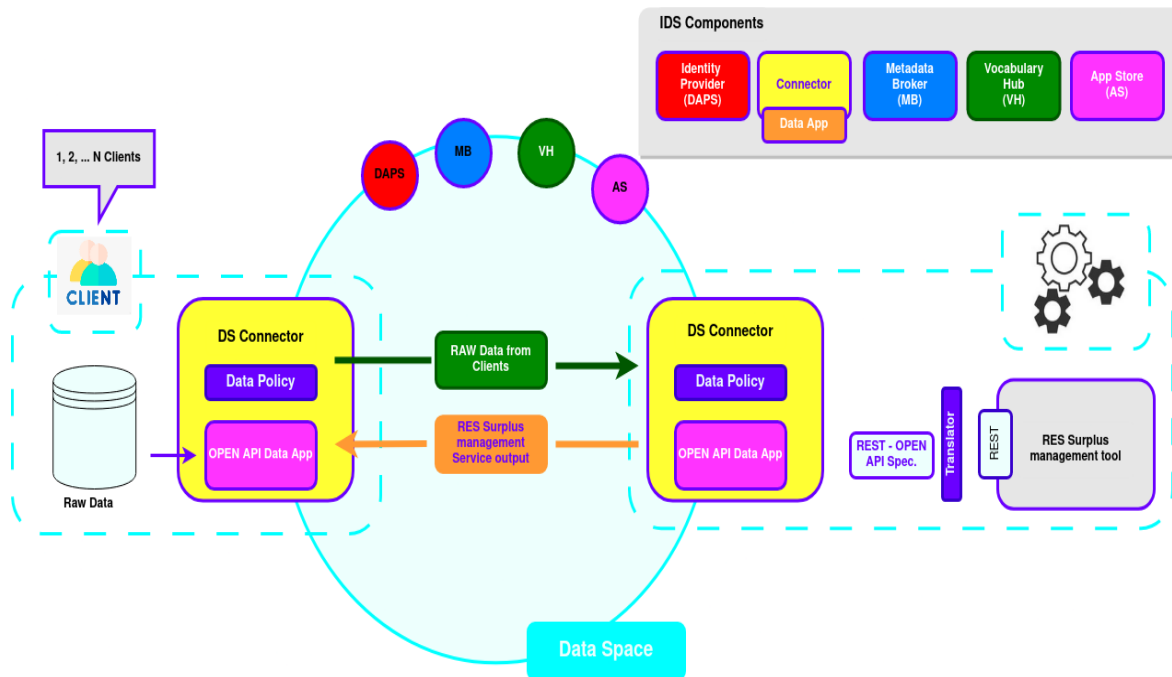


Figure 74: Architecture of Integration of data-driven service for managing surplus renewable energy generation in distribution systems with the ENERSHARE Data Space

3.10.6 Next steps

The next steps regarding this service are as follows.

1. Fine-tune the models with the pilot data

- The AI models used by the service needs to be re-trained with the actual data from the pilot as we have recently established the data sharing process with the pilot. Depending on the performance of the models on the data, it may be necessary to refine the architecture of the model to achieve the desired result.

2. Complete service development and integrate it with the ENERSHARE Data Space

- We aim to complete the development of all service components and integrate it with the ENERSHARE Data Space as described above. Also, the validation of the service will be conducted.



4 Data-driven Cross-Sector Services

This Chapter discusses the implementation updates of cross-sector services that integrate data and business processes stemming from the energy sector with other sector services such as water, transport, finance. The benefit of sharing data, in trustworthy manner, with other sectors is essentially exploited with the extraction of knowledge that can steer the under-coupling sectors based on the integration of such services with data space instances. The services that are described in the following sections will be part of the Enershare App Store environment for their discoverability and accessibility from all potential involved users in the Enershare data space ecosystem. Beyond their availability on App Store vital aspect which is reflected in the following sections is their integration with the Enershare data space implying not only the retrieval of processed data and analytics from the service itself but also the data sharing among stakeholders.

Table 9 presents a summary of the current development status and outlines forthcoming contributions for the data-driven energy services.

Table 9: Overview of the different data-driven cross-sector services releases in WP6

Data-driven cross-sector service	Features in deliverable D6.1 (Alpha version)	New features in D6.2 (Beta version)	Future features for D6.3 (Final version)
Multi-energy flexibility potential assessment (COMS)	<ul style="list-style-type: none"> - Upgraded the tool to support multiple energy vectors (electricity and heat) - initial REST API 	<ul style="list-style-type: none"> - Importing daily profiles from the user energy forecasting tool - REST API - forecasting capabilities 	<ul style="list-style-type: none"> - Evaluating the integrated models - Data Space integration -Enhancing service with emissions and environmental footprint for different multi-energy mixes
Cross-operators' portal (COP) (ED)	<ul style="list-style-type: none"> - Development of main openAPIs documentation, initial/basic analytics based on static information max up and down regulation 	<ul style="list-style-type: none"> - Integration with TNO substation load forecast service - COP user database modification - API docs updates to adapt on pilot custom needs - User Interface for registering assets - 1st cycle of DS integration 	<ul style="list-style-type: none"> - Locational flexibility analytics - Development of visualisation dashboard for analytics retrieved - Full cycle e.g., actual compilation of demo data of analytics testing - Integration with COP service to allow cross-energy flexibility orders



Data-driven cross-sector service	Features in deliverable D6.1 (Alpha version)	New features in D6.2 (Beta version)	Future features for D6.3 (Final version)
Emissions and ecological footprint (ENVI)	<ul style="list-style-type: none"> - Initial prototype DNN regression model based on a small sample - Simple implementation of REST API 	<ul style="list-style-type: none"> - Enhanced DNN model with much larger sample coupled with SCAM model - REST API to obtain emissions and EF - DS integration plan 	<ul style="list-style-type: none"> - Integration with multi-energy flexibility potential service - Data Space integration
EV charging monitoring and remote management (EMOT)	<ul style="list-style-type: none"> - IoT devices installed in charging stations - Charging station real-time monitoring and remote management services implementation 	<ul style="list-style-type: none"> - EV real-time monitoring service implementation - In-lab service validation phase 	<ul style="list-style-type: none"> - Prioritisation of charging and dynamic pricing aligned to the grid congestion level services implementation - Data Space integration
ML-based models for assessing renovation actions in residential buildings (AI4EF) (NTUA)	<ul style="list-style-type: none"> - Design of the methodological framework and algorithms for i) assessing retrofitting actions and ii) predicting the generation of rooftop solar panels and test of the models with sample data provided from Pilot 7 	<ul style="list-style-type: none"> - Full-stack technical solution, e.g., DB, User Interface - Data Sharing Infrastructure - API 	<ul style="list-style-type: none"> - Dataset augmentation with generative ML - Data Space integration - Front-end, back-end refinements.
Health insurance alarms for senior living alone (SEL)	<ul style="list-style-type: none"> - Service designed - Data monitoring campaign started 	<ul style="list-style-type: none"> - DB integration with existing backend - Frontend - Algorithm for pattern recognition - API 	<ul style="list-style-type: none"> - Complete the development of the event detection component d - Finalize API - Data Space integration
Appliances maintenance or retrofit (SEL)	<ul style="list-style-type: none"> - Service designed - Identification of data providers - Extra sensors installed and data collection period started. Library datasets created 	<ul style="list-style-type: none"> - Validation phase with supervised classification - Internal architecture with backend service with NILM algorithms, local DB, and API for third-party data access 	<ul style="list-style-type: none"> - Final version of the NILM algorithms - Data Space integration

4.1 Multi-energy flexibility potential assessment

4.1.1 Development progress

The multi-energy flexibility assessment service has been enhanced by using the weather forecast functions. This means that the integrated statistical models (regarding heat and



electricity consumption and production) can use the forecasted weather parameters instead of the historical PVGIS statistical data. Since the weather forecast has higher temperature and solar radiation variations, we have observed more dynamic profiles that better reflect real conditions, Figure 76.

Furthermore, instead of using statistical models by defining house parameters, we can now use the energy forecasting service to obtain energy consumption profiles for district heating supply to households and electricity consumption profiles based on smart meters. In both cases, the generated profiles support PVGIS or the weather forecast, which is selected via the user interface. For example, when using the district heating model, this is included in the definition of the household's heating and cooling parameters by selecting district heating and entering the house ID in the service's user interface, as shown in Figure 75. This method represents a move away from the previous reliance on generic statistical models and standard house parameters and leads to a more personalised and accurate multi-energy management process.

Furthermore, the service was enhanced with RESTful API so the flexibility analytics register service can collect the profiles of assets and houses to enable seamless communication between flexibility service providers and market operators for optimised grid operations and energy market participation.

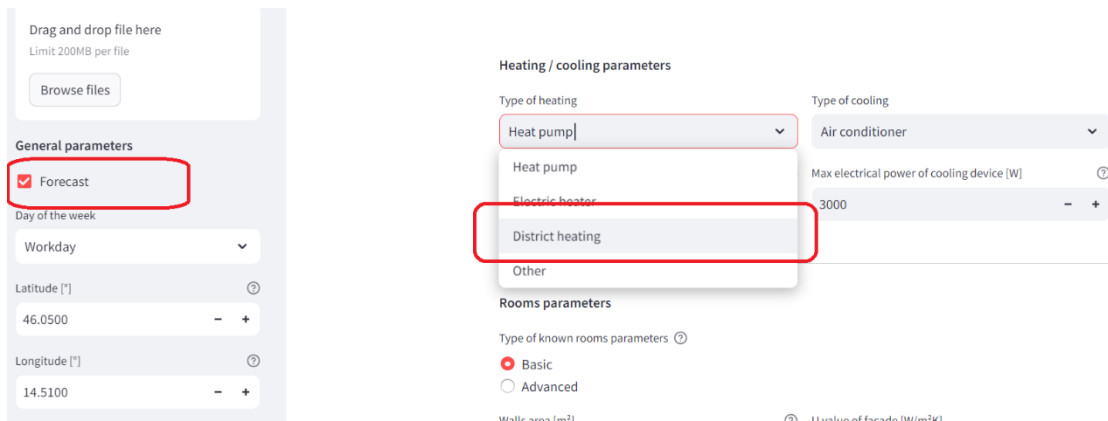


Figure 75: UI of Multi-energy flexibility potential assessment service with marked new features.



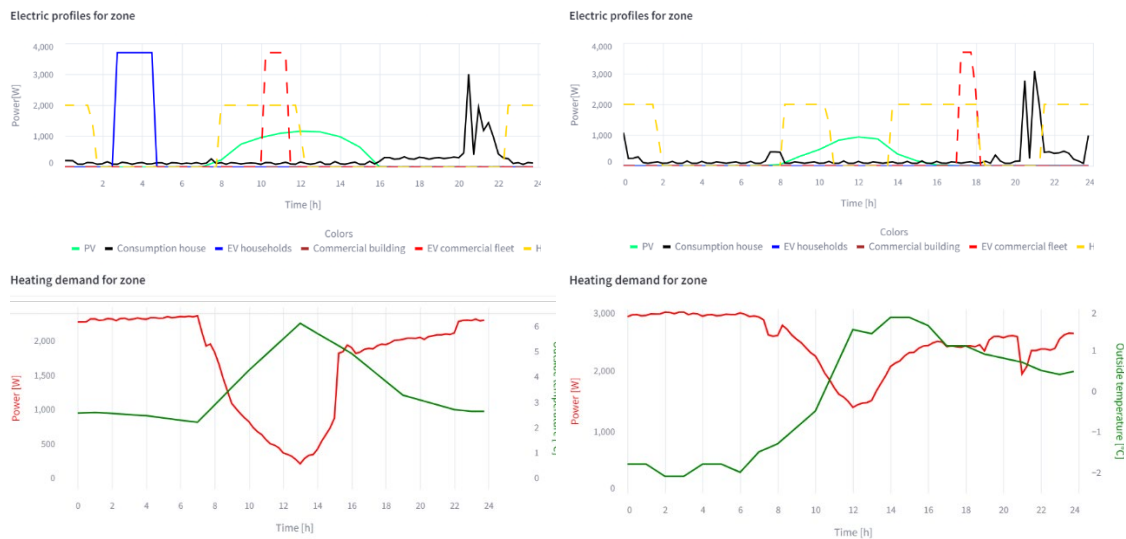


Figure 76: Comparing calculated profiles generated using historical weather data (PVGIS, left graphs) and weather forecast (right graph) for zone location of Slovenian pilot

The TRL of the tool has increased from 4 to 5, as all models were created using the real values and the RESTful API has been completed. Integration into the overall system is now possible. The service UI is deployed at <http://comsensus.eu:20988/>

4.1.2 Documentation updates

The tool provides a RESTful API for accessing and interacting with forecasting data.

Endpoints:

- 1. Root Endpoint ("/"):**
 - Description: Provides information about the API and the date of the last change.
 - HTTP Method: GET
- 2. List Models Endpoint ("/List"):**
 - Description: Lists of available household models in the zone (heating demand, outside temperature, EV, PV and consumption of building).
 - HTTP Method: GET
- 3. Forecasting Endpoint ("/forecasting"):**
 - Description: Generates profiles for individual household related to the planning tool assessment service.
 - HTTP Method: POST

The documentation is published on <http://comsensus.eu:20987/docs/>



4.1.3 Integration with the Data Space

As part of the integration plan for incorporating the multi-energy flexibility potential assessment Service into the Data Space, we will utilise the TNO TSG Connector as a standardised interface for secure data exchange. This connector will facilitate the interoperability between our service and the data sources within the Data Space, Figure 77. Furthermore, the service will also act as data source for Flexibility Analytics and Register service. Our integration approach will involve configuring the TSG Connector to ensure it aligns with the data formats and communication protocols required by the multi-energy flexibility potential assessment service. We will establish authentication and authorization mechanisms, leveraging the Identity Provider to manage access controls. Adequate testing will be conducted to guarantee that the connector reliably transmits data while adhering to the strict privacy and security standards mandated in the Data Space.

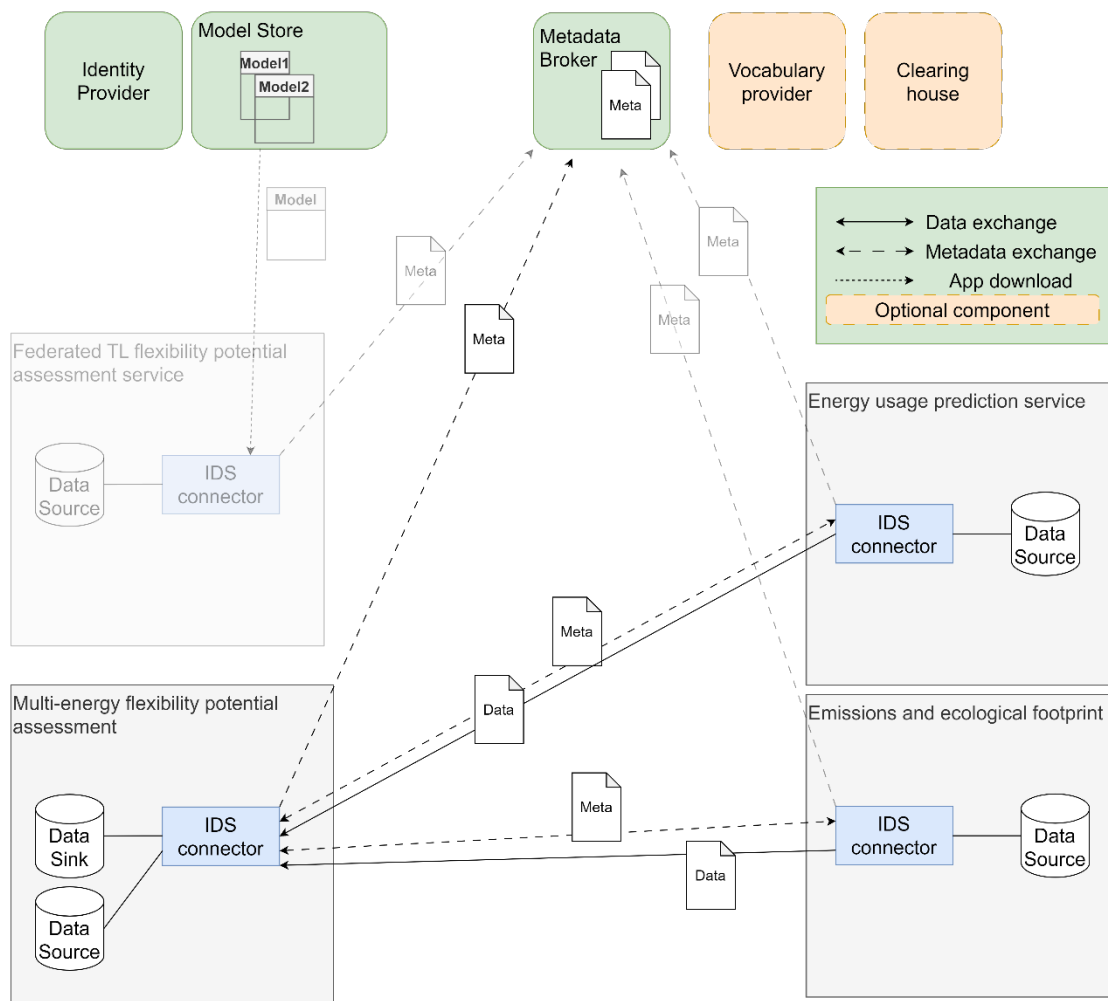


Figure 77: IDS Integration schema and possible interactions inside the pilot

4.1.4 Next steps

The service will undergo extensive testing with the data obtained from ELCE and KPV to refine the accuracy of the parameters used. This will not only involve tweaking the underlying models to represent real-world scenarios better but also upgrading the user interface for a more intuitive user experience that aligns with both the model capabilities and user requirements. In addition, the service will be enhanced with statistics on emissions and the ecological footprint. The final stage also involves integrating the service into the data space.

4.2 Cross-operators' portal (COP)

The Cross-operator's portal (COP) intends to provide a combined service and tool that from one side allows for system operators from different energy sectors and potentially domain to share information amongst them on grid conditions and share flexibility and on the other side them allow domain specific temporal flexibility analytics based on the assigned products/services.

4.2.1 Development progress

At this cycle of reporting, the COP service has been updated to adopt and accommodate the Slovenian pilot's (Pilot 3) needs. The initial documentation provided in the D6.1 for FAR has considered that all data exchanges would rely on XML-based Common information model (CIM) European style market profile (ESMP) standard profiles. As the remainder demo services are based on custom output profiles CSV and JSON formats, adaptation have been developed to ingest such data. Discussions with the pilot have indicated specific needs for cross-domain flexibility sharing (district heating peaks) which are encapsulated in the current COP service design.

The software cycle reported presents critical improvement on the APIs, featuring new APIs to retrieve data (resource groups updates, notifications, grid, and user analytics). The current TRL of the service is considered at TRL6 as already tests and necessary API interfaces are released to allow interaction with actual operators, end-user and FSP platforms.

4.2.2 Documentation updates

The latest updates on the APIs documentation are presented here:

<https://www.postman.com/crimson-escape-67103/workspace/far-service/documentation/13905546-4939170d-92e3-4853-88ba-1419699611cb>



The current version also reports UI features which are also presented on the Appendix. It shall be mentioned that this version contains streamlined UI features that allow user operator to react with received flexibility data but most critically to retrieve analytics available on the dataspace ecosystem. The OneNet connector also features UI characteristics that allow DSO and District Heating Operators (DHO) to negotiate with the COP service.

4.2.3 Tests of the services with the Data Space

Within this stage the COP service and its exploitation from DSO and DHO has been tested and integrated with an energy data space utilizing the OneNet connector and its UI features, following the architecture from Figure 78.

The DSO and DHO to interact with the COP shall deploy locally at their premises an OneNet connector. This in turn allows them to discover the FAR service and interact with it upon an agreement. Upon an active subscription data flow in p2p fashion can be performed, allowing for DSO/DHOs to share flexibility grid data with the COP service. The integration with the connector APIs has been performed via the COPs UI tool which is thereafter responsible to interact with the connectors NGSI-LD API.

The analytical discussion on the integration is provided in the Appendix.

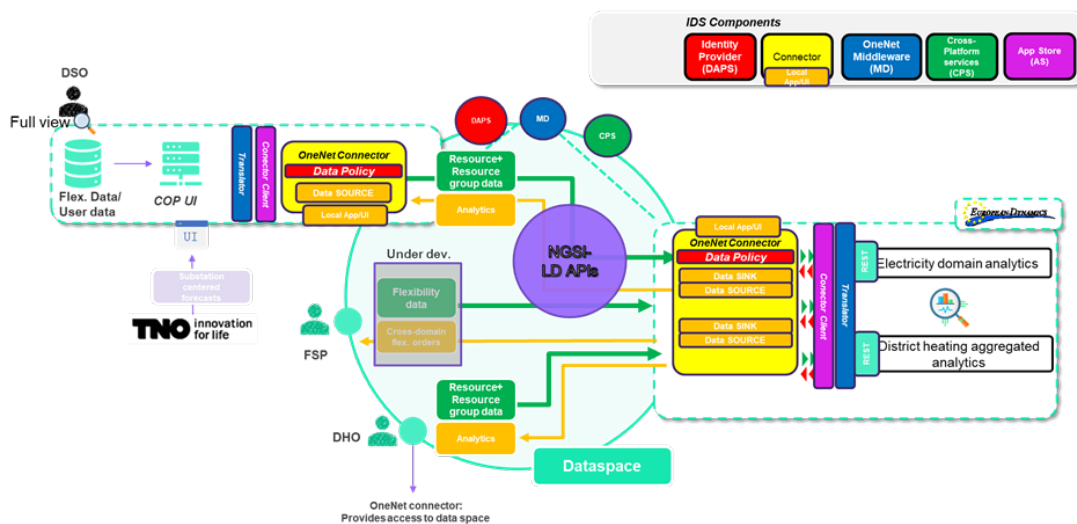


Figure 78: Architecture of the integration of the COP with Enershare data space using OneNet connector (NGSI-LD APIs)



4.2.4 Next steps

The next steps will focus on the improvement of the analytics, exploring the options to provide locational characteristics for flexibility needs among the domains needed. The next software cycle will consider the finalization of all the list of flexibility and grid analytics for all users, fact which will allow the preparation of the visualisation dashboards at Cross-Operators portal Interface level. Different scenarios will be demonstrated with multiple FSPs managing different types of resources (PV, BESS, EV, heating flexibility) showcasing the importance of the analytics for all (cross-domain flexibility, better condition for FSP to market flexibility). Finally, the integration with COP service via the data space ecosystem to allow cross-energy flexibility sharing will be highlighted, validating the importance of data and analytics sharing.

4.3 Emissions and ecological footprint

4.3.1 Development progress

The emissions and ecological footprint service has undergone a significant upgrade integrating a hybrid model that merges the strengths of a Deep Neural Network (DNN) model with a Shape Constrained Generalised Additive model (SCAM). This enhancement addresses specific edge cases where the DNN model previously failed to yield sensible results. In a noteworthy improvement the service now utilizes the Open-Meteo free open-source weather API allowing users to input the longitude and latitude of a building directly instead of temperatures. This approach replaces the need for user providing minimum, maximum, and average monthly temperatures. Moreover, the model's capabilities have been substantially expanded, moving from a few thousand samples to an impressive database of over one hundred thousand samples. This vast improvement in data volume significantly enhances the accuracy and reliability of the model and service.

Upgraded service has been tested on our production server, ensuring robust and reliable performance. One of the most notable applications of this enhanced capability was its use in generating the annual report for the Municipality of Ljubljana. In this report service results of ecological footprint were successfully incorporated. Additionally, the service now includes a fully integrated RESTful API. This allows for seamless interaction with other partners.

The TRL of the service has increased from 4 to 5 as model is now more robust giving sensible results for edge cases and results were used in annual report, and the RESTful API has been completed. The temporary service for testing is deployed at: <http://enershare.epa.si:20988> (see API documentation for further details).



4.3.2 API documentation

The tool provides RESTful API for accessing and interacting with service:

Endpoints:

1. **Root Endpoint ("/"):**
 - Description: Provides information about the API and the date of the last change.
 - HTTP Method: GET
2. **List types endpoint ("/description"):**
 - Description: Lists all building types, construction types, fuel types and emissions types.
 - HTTP Method: GET
3. **Fuel change emissions calculator endpoint ("/fuel_change_emissions"):**
 - Description: Heating system change emissions matrix calculator
 - HTTP Method: GET
4. **Emissions calculator endpoint ("/emissions"):**
 - Description: Calculation of different emissions given building's useful energy for heating, type of heating system and installation year of a heating system.
 - HTTP Method: GET
5. **Emissions estimation calculator endpoint ("/estimation"):**
 - Description: Emissions estimation from building attributes calculator
 - HTTP Method: POST

The documentation is published on http://enershare.epa.si:20988/_docs/.

4.3.3 Tests of the services with the Data Space

In our strategy for integrating the Emissions and Ecological Footprint Service into the Data Space framework, we plan to adopt the TNO TSG Connector, as depicted in Figure 79. This tool will serve as a primary channel for safe and efficient data transfer. Its role is pivotal in ensuring seamless interaction between our service and various data repositories within the Data Space. Moreover, this service will contribute as a valuable data source for multi-energy flexibility potential (see section 4.1).

To achieve this integration, we'll configure the TSG Connector to be compatible with the specific data structures and communication protocols of the Emissions and Ecological Footprint Service. Secure access will be a top priority, and we will implement robust authentication and authorization processes, supported by the Identity Provider, to control access. Comprehensive



testing will be performed to ensure that the connector not only transfers data effectively but also complies with the stringent privacy and security guidelines required in the Data Space.

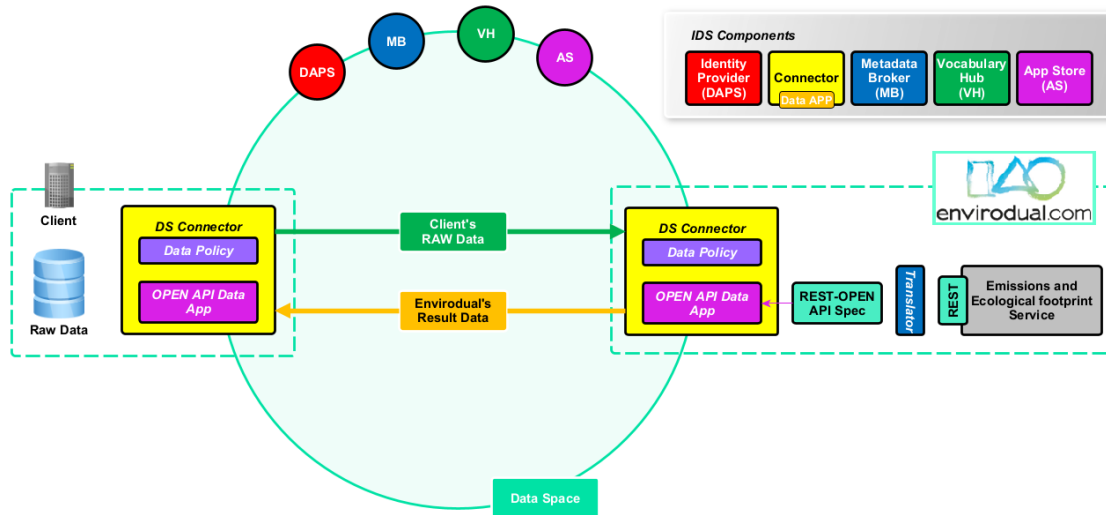


Figure 79: Schematic of the Data Space integration for the emissions and ecological footprint service

4.3.4 Next steps

The hybrid model employed by the service will be subjected to thorough testing, utilizing extra data acquired from ELCE, KPV and other partners (e.g., COMS). Also, we will upgrade our current DNN model with Constrained Monotonic Neural Networks in hope to use pure DNN model without help of SCAM model to cover edge cases. The last phase includes incorporating the service into the Data Space as well.

4.4 EV charging monitoring and remote management

4.4.1 Development progress

The continuous increase of EV causes a necessary strengthening of the power lines to supply enlarged amount of energy. However, through a cooperation mechanism between DSOs, CPOs (Charging Point Operators) and EV users, it is possible to reduce the power grid upgrade magnitude by coordinating the EV charging. DSO monitors the electricity grid via distributed smart meters and, thanks to accurate forecasting systems, can identify how, when and where to charge EV to avoid congestion problems. CPO, thanks to EV charging monitoring and remote management service, will thus be able to provide a dynamic charging session based on real-



time/forecasted DSO needs, bringing a benefit to DSO who will have a balanced grid and, meanwhile, offering an advantageous charging price in charging stations located in congested areas and attract a greater number of EV users increasing CPO revenue. DSO, CPO, and EV data will be hosted in ENERSHARE Data Space.

Charging station real-time monitoring and remote management services implementation was already reported in deliverable D6.1. The beta version is about EV real-time data collection implementation for smart charging session execution.

Regarding EV monitoring, EMOT installed an on-board diagnostic device (OBD) to retrieve data from the EV - Figure 80; OBD is a IoT component that utilise a TCP/IP communication to a TCP/IP server. The network connectivity of the OBD device is via data SIM (UMTS) and the server is a python software; OBD protocol is MQTT, and the sampling rate is 2 seconds. The OBD connects to the diagnostic interface from which it can extract the information from the electric vehicle control unit using the CAN-bus protocol. The output data format of the OBD is an ASCII string; when the data is sent to the server, it is reorganised into a wrapper, thus obtaining a grouping of the data in JSON format.



Figure 80: On-Board Diagnostic Device (OBD) connected to EV diagnostic interface

4.4.2 Documentation updates

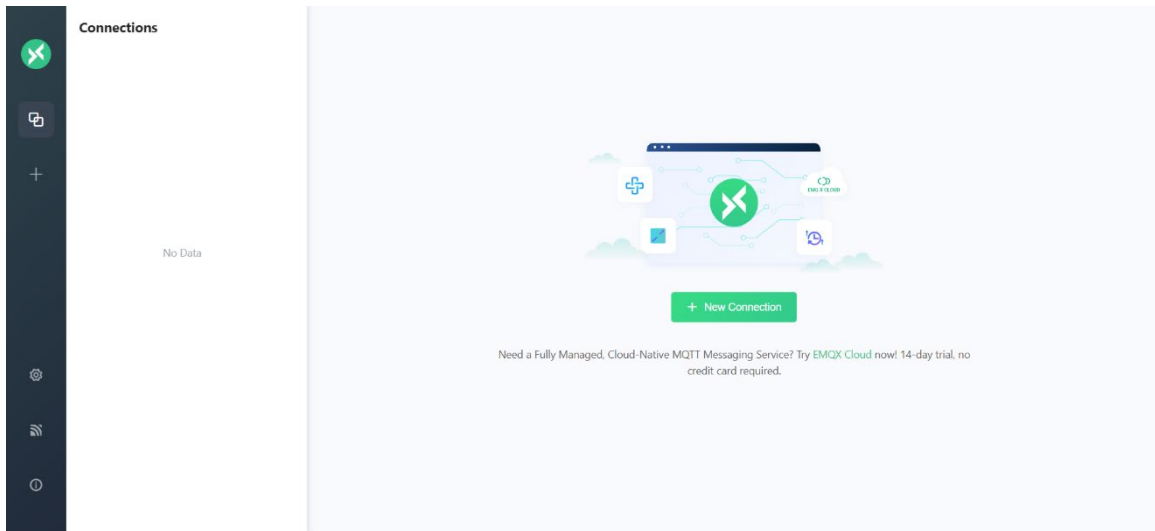
To access real-time data collected from EV and charging stations deployed on Italian pilot site, following instructions must be considered:



ENERSHARE has received funding from [European Union's Horizon Europe Research and Innovation programme](#) under the Grant Agreement No 101069831

- Access to the following link: http://www.emqx.io/online-mqtt-client#/recent_connections/4f3b31fd-a177-426f-a1d5-6d646c57b1f2

- Click on “+New Connection”



- Fill in with the following information:

*Name: GianFilippo

*Host: wss:// emotion-projects.eu

*Port: 443

Activate SSL/TLS

-Fill in only “General” part, not “Advanced” part



General

* Name ⓘ

* Client ID ⓘ ⓘ

* Host

* Port

* Path

Username

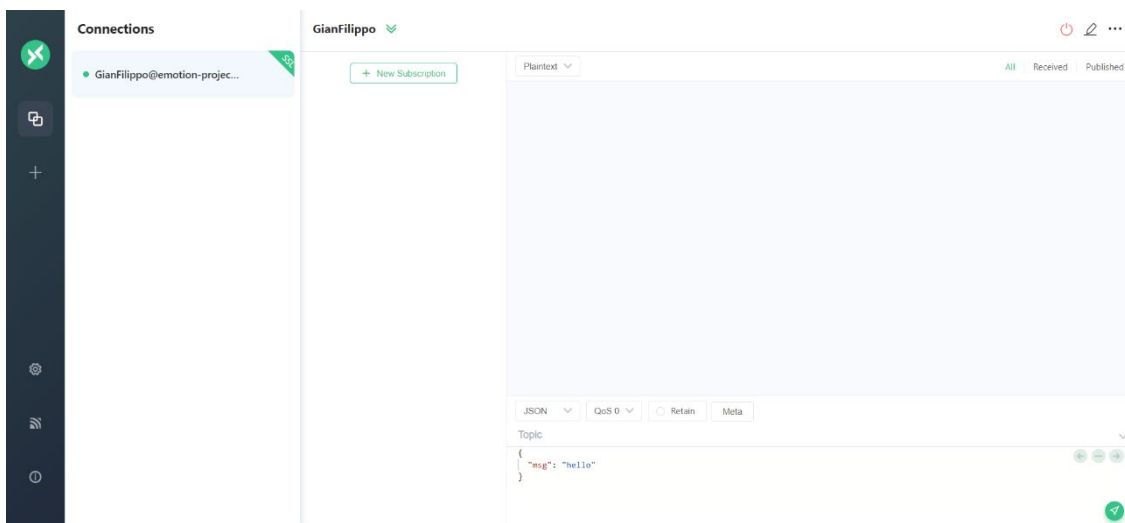
Password

SSL/TLS

SSL Secure ⓘ

ALPN

- Click on “Connect”



The screenshot shows the MQTT client interface. On the left, there is a sidebar with navigation icons. The main area is titled 'Connections' and shows a connection to 'GianFilippo' with a status indicator. Below the connection name, there is a '+ New Subscription' button. The right pane shows the received message in plaintext format: `{ "msg": "hello" }`. The interface also includes options for message format (JSON, QoS 0), Retain, and Meta.

- Click on “ + New Subscription” to add following topics:

vehicle-states/Omega-3749

vehicle-states

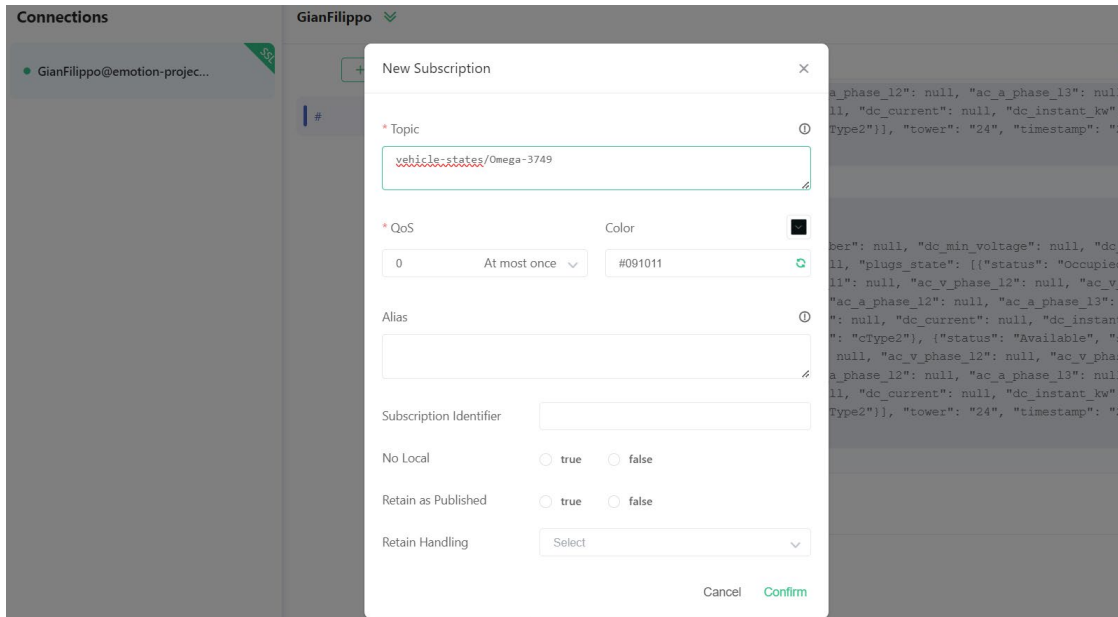
vehicle-states/Omega-378E

tower-states/24

tower-states/55



ENERSHARE has received funding from [European Union's Horizon Europe Research and Innovation programme](#) under the Grant Agreement No 101069831



4.4.3 Integration with the Data Space

Exchange of data and commands between ENERSHARE Data Space and EMOT system will be granted by TNO Security Gateway (TSG) IDS connector, to be integrated with the rest of ENERSHARE Common IDS. Data from charging stations and electric vehicles will be available in EMOT electric mobility platform data lake, containing both historical and real-time data. EMOT data will be wrapped and provided in JSON format according to the common ENERSHARE Data Models. These created JSON files will be sent as an IDS artifact and an orchestrator will be developed to call the different service components that will be dockerised.

In particular, as shown in Figure 81, charging stations and electric vehicles deployed for Terni pilot demonstration activities will be integrated in an electric mobility platform dedicated to ENERSHARE project, that will be implemented by EMOT during Full Pilot Operation phase. Charging station and EV real-time and historical data will be shown in the electric mobility platform and based on those data CPO will prioritise charging for EV users sharing their charging profiles allowing them to take advantage of a dynamic pricing aligned to the grid congestion level; in this way, EV will be charged at lower cost and electric grid congestion level will be reduced. Charging schedule prioritisation will be derived thanks to the Energy Forecasting and Energy Flexibility Management services applied to electric grid (smart meters and phasor measurement units) data and electric mobility (charging stations and electric vehicles) data; charging dynamic price will be based on monitored grid congestion level and will be carried out via the P2P congestion marketplace.



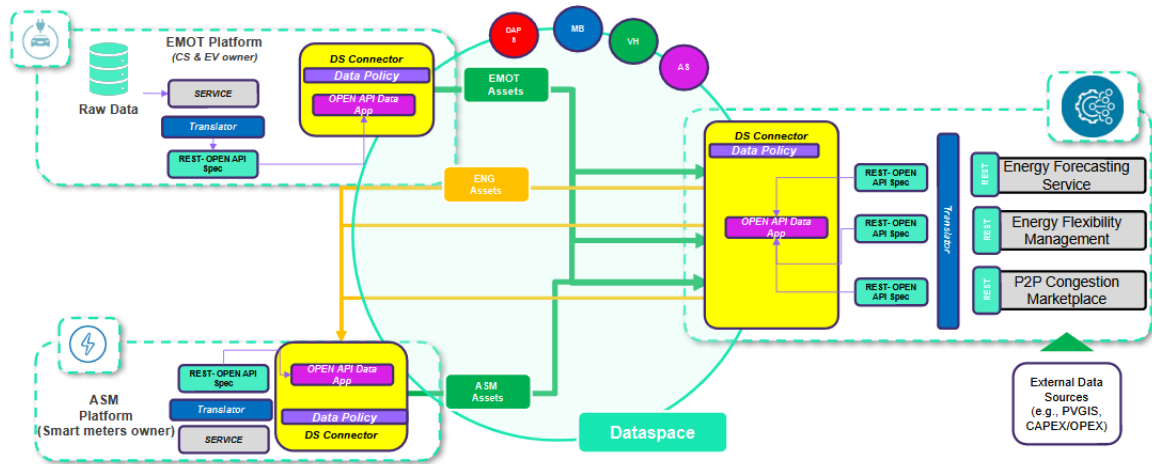


Figure 81: Schematic for “EV charging monitoring and remote management” service integration with the Data Space

4.4.4 Next steps

EV real-time data will be used to increase charging station remote management service effectiveness to avoid DSO congestion problems, providing EV fleet flexibility potential provision to better manage smart charging. EV charging profiles, derived by monitoring service enabling, will be used to prioritise charging and/or parking slot availability in peak hours. Furthermore, a dynamic parking/charging price will be tested based on grid congestion level, thanks to the cooperation between DSO and CPO, as indicated above. The final stage involves the connection with ASM (DSO) energy flexibility service and the Data Space integration.

4.5 ML-based models for assessing renovation actions in residential buildings (AI4EF)

4.5.1 Purpose and overview of AI4EF

The need for AI4EF tool arises from the growing importance of sustainable and energy-efficient practices in today's world. Private house owners are often faced with the dilemma of deciding whether to invest in improving the energy efficiency of their existing building or in the installation of renewable energy technologies. Several factors drive the demand for such a tool, as follows:

- **Environmental Concerns:** With increasing environmental awareness and concerns about climate change, individuals and communities are seeking ways to reduce their



- carbon footprint. Renewable energy technologies can contribute to this goal, but not every investment in this area is cost-effective.
- *Energy Costs:* Rising energy costs make it imperative for homeowners to find ways to reduce their utility bills.
 - *Government Incentives:* Many governments provide incentives and subsidies for renewable energy installations or energy-efficient upgrades. A tool can help homeowners navigate these complex programs and understand their financial implications.
 - *Long-term Savings:* Investments in energy-efficient buildings or renewable energy sources often lead to long-term savings. The tool can help homeowners understand the potential return on investment over time.
 - *Property Value:* A more energy-efficient and sustainable house can have a higher resale value. The tool can assist homeowners in making informed decisions that benefit their long-term financial interests.

In summary, this tool is needed to provide homeowners with the information and analysis required to make informed decisions about investments that align with their environmental goals, financial interests, and the changing landscape of energy costs and incentives. It can serve as a valuable resource for individuals seeking to create a more sustainable and economically sound future for their homes.

The suggested solution is intended to be initially applied in the Latvian energy system - as a publicly available tool for the owners of private houses to determine whether to invest in their building or in the installation of renewable energy resource technologies. The stakeholders include owners of private houses, ministries and institutions that grant public support. Note here that AI4EF comprises the following two subservices: i) service 1: Energy efficiency measures recommender; ii) service 2: PV investments profit estimator).

Regarding data privacy, compliance with data privacy regulations, such as the General Data Protection Regulation (GDPR) and other local data protection laws will be accomplished through:

1. sensitive data protection through anonymisation by the pilot to minimize the risk of data breaches.
2. user consent will be obtained for data collection and processing. Users will have the option to opt in or out of data collection, and that their data will not be used without their explicit consent.
3. description how long the collected data will be retained and when it will be securely disposed of, in accordance with relevant data protection laws.



4.5.2 Development progress

In deliverable D6.1, we described the conceptual description of the service with regards to:

- service functionalities
- data preparation, wrangling and fitting.
- model selection, training, and evaluation
- input/output data formats
- employed technologies.

D6.2 aims to present a comprehensive analysis of the service's implementation, including updates from D6.1 and an up-to-date summary of the aforementioned aspects. Additionally, it's critical to give ample justification for the elements that remained constant given the similarities among deliverables (such as service functionalities).

4.5.2.1 ML model developments

This section is organised based on basic steps of a standard machine learning pipeline, presenting (a) data preparation (wrangling, feature extraction/justification, fitting) (b) model engineering (selection, training) and (c) model evaluation (results, metrics). Note here that the ML models developed for service 1 and service 2 are based on two different datasets respectively as described in section 4.8.4 of D6.1.

4.5.2.1.1 *Data preprocessing*

Initially, data are being processed, removing potential outliers and duplicates. With regards to model training, we apply label encoding, since several features (e.g. energy class, region) are categorical, but our models require numerical inputs for both features and target variables. Additionally, feature scaling is performed to ensure that features contribute equally to the model training and remove any bias towards magnitude of feature inputs. Our analysis points out that the small datasets display class imbalances in several key features (e.g. above-ground floors, energy class after). Considering this, we stratify our dataset during training, so that each class is adequately represented in both the training and testing sets.



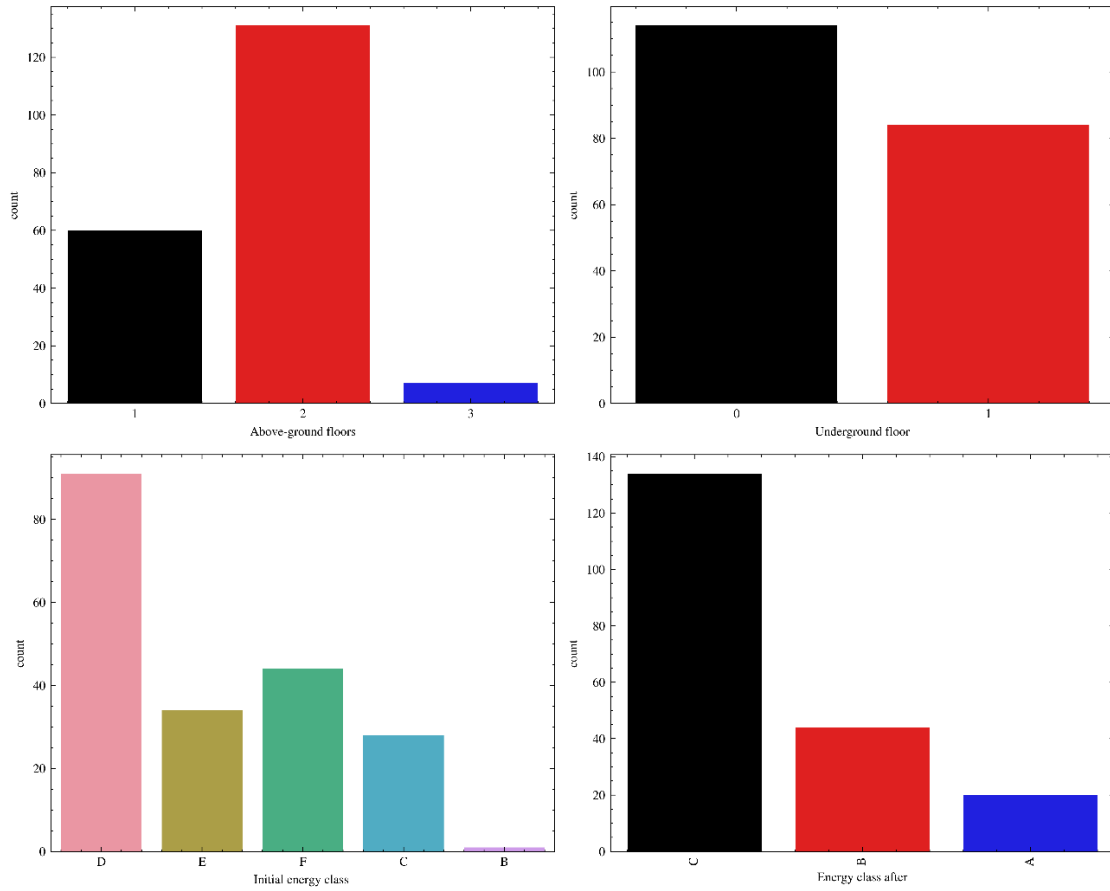


Figure 82: Count plots describing the frequency of observations in our features in service 1.



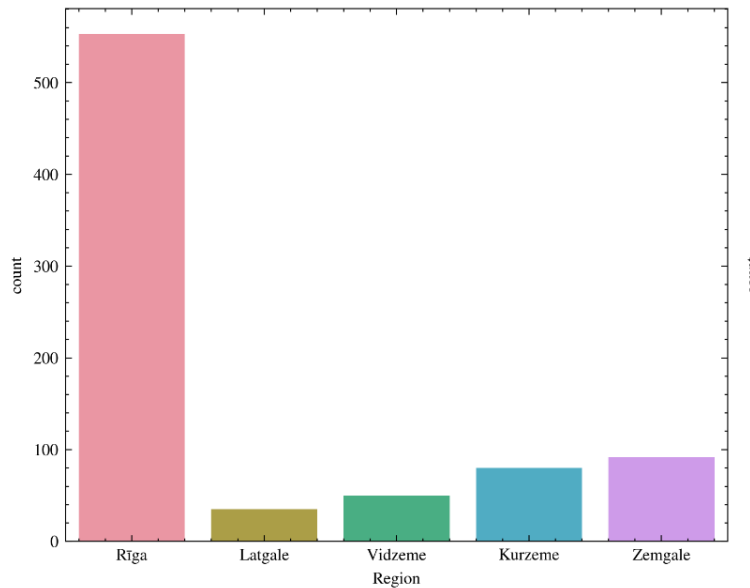


Figure 83: Count plots describing the frequency of observations in the "Region" feature in service 2

Proceeding with feature extraction, our aim is to select features from our dataset that are (a) strongly correlated with our product targets and (b) easily accessible to our users and do not require elaborate calculations. As a result, Table 10 displays the input features that fit these conditions, as well as the respective product targets.

Table 10: Features and targets of the two ML services developed.

Service 1		Service 2	
Inputs	Targets	Inputs	Targets
Building Total Area	Carrying out construction works	Region	Electricity produced by solar panels
Reference area	Reconstruction of engineering systems	Electricity consumption of the grid before	Primary energy consumption after
Above-ground floors	Heat installation	Primary energy consumption before	Reduction of primary energy consumption
Underground floor	Water heating system	Current inverter set power	CO2 emissions reduction
Energy consumption before		Inverter power in project	
Initial energy class			



Energy class after renovation			
-------------------------------	--	--	--

4.5.2.1.2 Model Training

Service 1 - Energy efficiency measures recommender: Contains a classification model that provides one or more proposed retrofitting actions to reach the target energy class. The algorithm applies grid search cross validation to determine the most fitting architecture and their respective hyperparameter values. The resulted model architecture is the multi-layer perceptron (MLP): an effective architecture for a range of machine learning applications because of their capacity to automatically extract pertinent features, learn intricate, non-linear relationships, and function as universal approximators. Their versatility is attributed to their capacity for parallel processing, availability of regularization techniques, adaptability to various data types, and potential for transfer learning.

Service 2 - PV investments profit estimator: Contains a regression model that forecasts the amount of electricity produced by the solar panels installed in the project, which in turn is used to calculate the remaining targets. This service also utilized the MLP architecture, following its success in service 1, using Pytorch Lighting: a lightweight PyTorch wrapper offering a high-level neural network training interface, most notable for its simplified training loop and code modularity that grants great accessibility to its developers. The algorithm tunes our MLP's hyperparameters using a multi-dimensional grid of hyperparameter values to determine the right combination that optimizes the model's performance.

4.5.2.1.3 Results

We evaluate our models using several metrics depending on the problem, ensuring the quality of our forecasts and providing a quantitative measure of the model's accuracy, precision, recall, etc.

Our analysis in service 1, can be summarized by the micro average and weighted average f1-score. Weighted average calculates metrics for each class, but it gives more weight to classes with more instances. Micro-average calculates metrics globally by counting the total true positives, false negatives, and false positives across all classes. As seen in Table 11, the results produce a weighted average and micro-average F1-score of 0.80, suggesting that our model is achieving both good precision and good recall on average. This can be further proved by both the precision and recall column, indicating high accuracy of true positive predictions as well as our model's ability to correctly identify instances of a given class respectively. Concerning the



rows of classes 2 and 3, we can see a significant drop in our scores, which is expected given the low number of actual instances of each class in the dataset (support column)

Table 11: Classification Report for service 1 model

Class	Precision	Recall	F1-score	Support
0	0.83871	0.86667	0.85246	30
1	0.89474	0.70833	0.79070	24
2	0.00000	0.00000	0.00000	0
3	0.50000	0.33333	0.40000	3
Micro avg	0.83019	0.77193	0.80000	57
Macro avg	0.55836	0.47708	0.51079	57
Weighted avg	0.84447	0.77193	0.80264	57
Samples avg	0.81197	0.80342	0.77692	57

With regards to service 2, we evaluated using multiple metrics including RMSE, SMAPE, MAE as seen in Table 12. The root mean squared error (RMSE) was selected as our main metric: a well-known metric that quantifies the average magnitude of errors between predicted and actual values: After evaluation on a small testing dataset, the resulted RMSE value reaches 0.14 which, in unison with the remaining metrics, suggests relatively small prediction errors on average.

Table 12: Evaluation metrics for service 2 model

Metrics	Value
SMAPE	0.29
MAE	0.11
MSE	0.01
RMSE	0.14



4.5.2.2 Software architecture

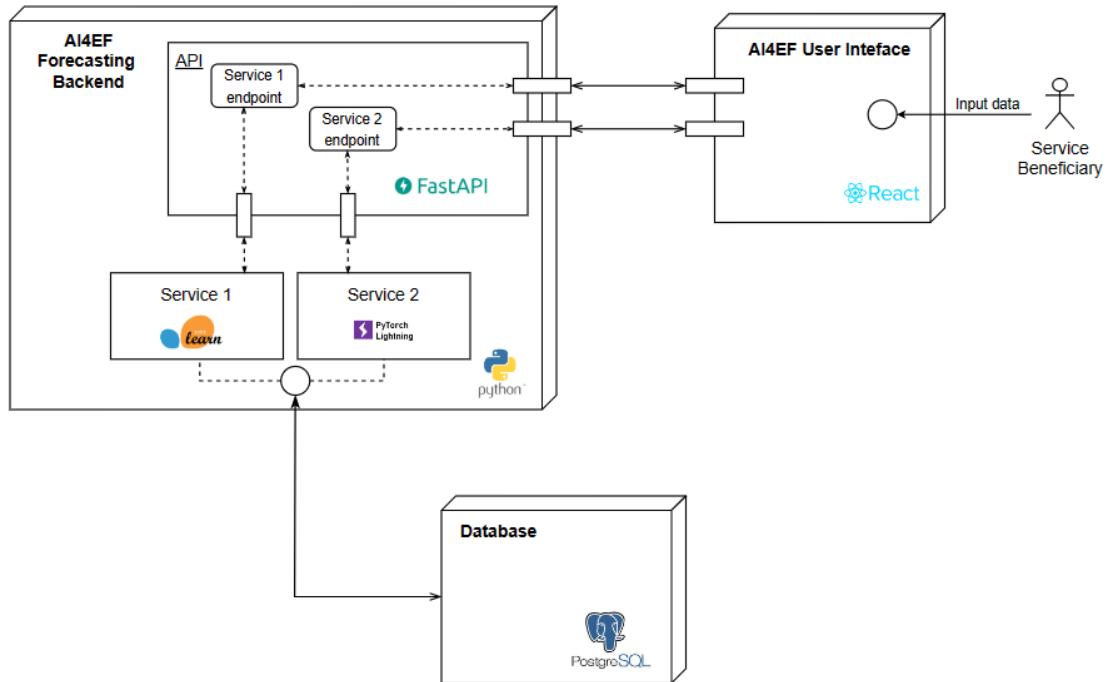


Figure 84: Architectural diagram of the designed technology (deployment)

The development of AI4EF architecture follows a modular, micro-service-oriented development, composed of several layers that each carry out distinct tasks and implement different application functionalities, as illustrated in Figure 84.

- Database:** A database containing user-provided buildings, as well as our models and scalers used to generate forecasts. We employ PostgreSQL: an open-source relational database management system that is, which presents a multitude of benefits for both businesses and projects. It guarantees data integrity by adhering to ACID compliance, supports advanced data types, and offers robust concurrency control. It is also equipped with scalability features such as parallel queries and table partitioning, making it suitable for handling a wide range of workloads. Finally, its optimization capabilities and strong security features further enhance the efficiency and security of data management. As a means of interacting with our databases easier, we additionally employ PostgREST: a web server that serves a fully RESTful API, that allows to expose our databases as RESTful APIs.



- **Forecasting backend:** This component serves as the foundation of AI4EF and assumes the following responsibilities: (a) retrieving data from the database, (b) conducting preprocessing and harmonization in alignment with the input requirements of the forecasting model, (c) training the model and fine-tuning its hyperparameters, (d) evaluating the model, and (e) managing the storage, versioning, and serving of the model (f) handling user requests, providing the appropriate responses. The backend's code is organized in several files executing different parts of the process:
 - **classifier.py/regressor.py:** python scripts responsible for the pipeline process in service 1 and 2 respectively (see more details in section 4.5.2.1.2)
 - **api.py:** python script deploying the necessary API endpoints to communicate with both services' models as well as the front-end user interface. It is responsible for receiving requests from the user interface (front-end), containing user building data. Requests undergo a wrangling process to extract the necessary information, which is then used as input features for the models of our service. Once our models generate the forecasts, the API formats and sends them to the front-end as a reply. Our API is built using Fast API: a modern, fast (high-performance), web framework for building APIs with Python. Its ability to generate automatic and interactive API documentation, and compatibility with existing Python tools/frameworks contribute to its comprehensive appeal as a commonly preferred framework for building robust and scalable APIs.
 - **service1_outputs.json/service2_outputs.json:** JSON files that contain each target's name, description, data type. Paired with the respective model forecasts, they are utilized by API as format for the user response sent to front-end.

Graphical User Interface: The front-end environment of the AI4EF incorporates a Graphical User Interface (GUI) that has been developed using React. React is an open-source JavaScript library that is widely used and extensively documented. It offers great flexibility and efficiency for building user interfaces. One of its key advantages is its component-based architecture, which allows developers to create reusable and modular UI components. This leads to faster rendering and optimal performance of the overall application. React also provides a robust ecosystem of libraries and tools, ensuring scalability and maintainability of the applications created with it. Furthermore, React has a vibrant community of developers who actively contribute and provide support for the library.



4.5.2.3 Front-end

The dashboard that is currently under development is Single Page Application (SPA) which is developed using the React² library. SPAs are a state-of-the-art approach to web development and offer several advantages compared to Multi-Page Applications. Specifically, SPAs are faster than Multi-Page Applications since most resources (HTML/CSS/Scripts) are only loaded once throughout the lifespan of an application and from that point on they do not update the entire page but only the required content. SPAs also offer better caching, easier debugging and in general a better user experience.

The dashboard is designed and implemented with user-friendliness in mind, meaning that it should be simple, legible, and fully responsive, with an interface that adjusts in both desktop and mobile devices.

The CSS framework used for the creation of the dashboard is MUI³, one of most popular libraries used with React. It has been created by Google and can be used as a single resource for all the styling needs, since it provides components, styles, themes, and icons.

In the following sub-sections, a presentation of the pages that have been developed until now is provided. Specifically, each page will be presented, along with a description of the input it requires (where applicable) and a screenshot of its current status.

² <https://react.dev/>

³ <https://mui.com/>



4.5.2.3.1 Homepage

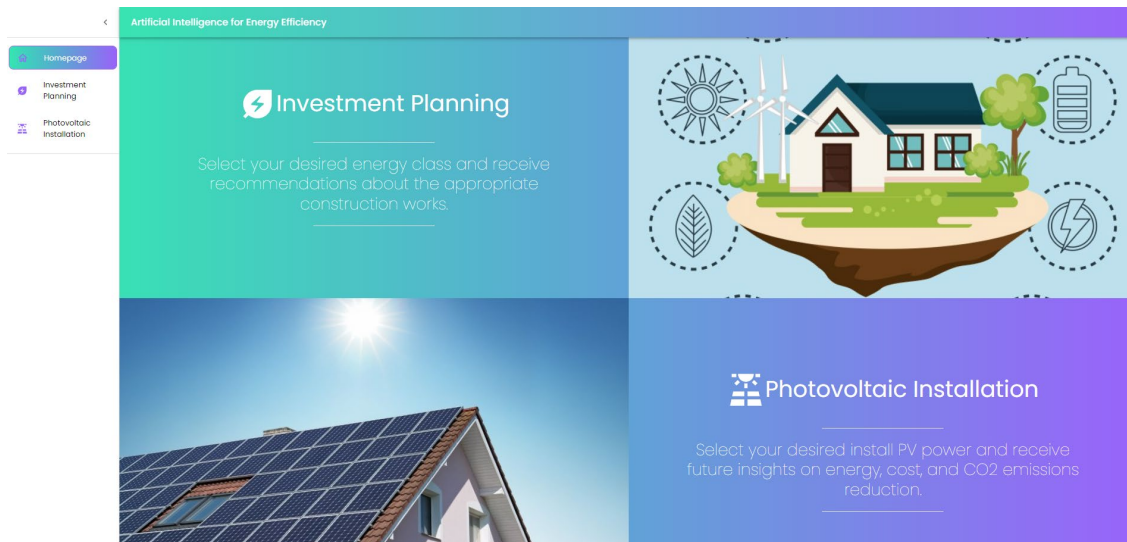


Figure 85: AI4EF dashboard - Homepage (screenshot)

In the homepage (Figure 85), the main capabilities of the dashboard are displayed, along with a brief description and an indicative image. On the left side of the screen, a togglable side-bar menu appears, and it is present in all the pages of the dashboard.

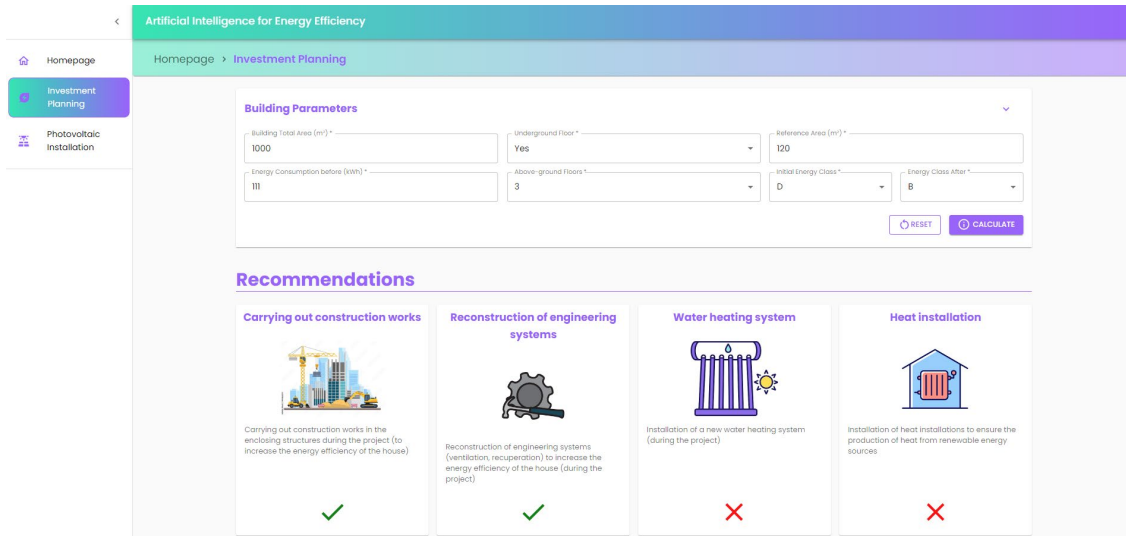
4.5.2.3.2 Investment Planning page

Upon loading this page, the user can see a form, entitled **“Building Parameters”**. In this form there are seven fields, namely **“Building Total Area”**, **“Underground Floor”**, **“Reference Area”**, **“Energy Consumption before”**, **“Above-ground Floors”**, **“Initial Energy Class”** and **“Energy Class After”**. All the fields are required, and the form cannot be submitted without all of them being filled. Error messages below each one of the required fields are displayed in case the user tries to submit the form without having filled it in.

By clicking the button **“CALCULATE”**, this information is sent to the backend, in order for the ML model to make the appropriate calculations and sent back to the front-end the recommended innovation measures. There is also a **“RESET”** button, which clears the form, allowing the user to re-fill it.

After the calculations have been completed, another section appears on this page, listing the calculated recommendations. All the aforementioned components are displayed in Figure 86. The form has been filled with random numbers.





The screenshot displays the 'Artificial Intelligence for Energy Efficiency' (AI4EF) - Investment Planning page. The interface includes a sidebar with 'Investment Planning' and 'Photovoltaic Installation' options. The main content area features a 'Building Parameters' form with the following inputs: Building Total Area (1000), Underground floor (Yes), Reference Area (120), Energy Consumption before (III), Above-ground floors (3), Initial Energy Class (D), and Energy Class After (B). 'RESET' and 'CALCULATE' buttons are located at the bottom right of the form. Below the form, a 'Recommendations' section lists four items: 'Carrying out construction works' (marked with a green check), 'Reconstruction of engineering systems' (marked with a green check), 'Water heating system' (marked with a red X), and 'Heat installation' (marked with a red X).

Figure 86: AI4EF - Investment Planning page (screenshot)

4.5.2.3.3 Photovoltaic Installation

This page follows the same logic as the “**Investment Planning**” page. Upon loading this page, the user can see a form, entitled “**Photovoltaic Installation Parameters**”. In this form there are seven fields, namely “**Electricity Consumption of the Grid**”, “**Primary Energy Consumption Before**”, “**Current Inverter Set Power**”, “**Inverter Power in Project**” and “**Region**”. All the fields are required, and the form cannot be submitted without all of them being filled. Error messages below each one of the required fields are displayed in case the user tries to submit the form without having filled it in.

By clicking the button “**CALCULATE**”, this information is sent to the backend, for the ML model to make the appropriate calculations and sent them back to the front-end. There is also a “**RESET**” button, which clears the form, allowing the user to re-fill it.

After the calculations have been completed, another section appears on this page, listing the forecasted values. All the aforementioned components are displayed in Figure 87. The form has been filled with random numbers.



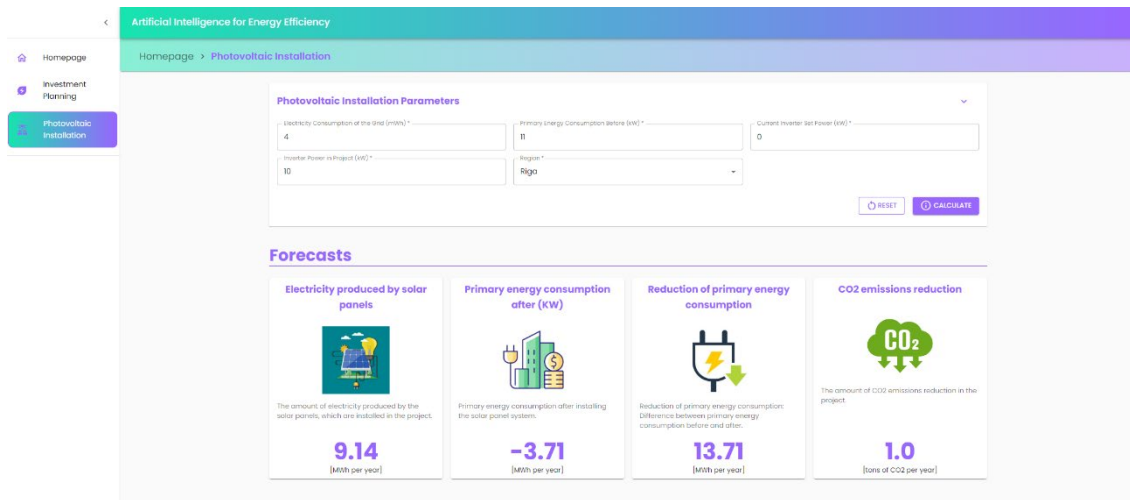


Figure 87: AI4EF - Photovoltaic Installation Planning page (screenshot)

4.5.2.4 Data model

Two datasets related to energy efficiency investments, used for the two above-described services (EF_comp, Sol_pan_comp) have been integrated into the ENERSHARE semantic data model through continuous collaboration with WP3 and specifically Task 3.1 led by ENGIE. More details can be found in the related deliverable.

4.5.3 Documentation

4.5.3.1 API description

The backend service offers two endpoints (service API), one for each service, used for communication between the frontend and the backend components. Additionally, there two endpoints exposing GET operations from the AI4EF database, which will be also used in the context of the Data Space for data sharing purposes. Below, we provide the API description tables for the services APIs (communication between frontend and backend) as well as database APIs (communication with database). However, they are also provided in the GitHub Pages of AI4EF⁴.

4.5.3.1.1 Services API

Enershare AI4EF API: service 1 endpoint	
Description	Get building parameters service 1
HTTP Method	POST

⁴ <https://epu-ntua.github.io/enershare-ai4ef/>



Endpoint URL	<host_ip>:8888/service_1/inference
Parameters	No parameters
Output example	<pre>[{ "title": "Carrying out construction works", "description": "Carrying out construction works in the enclosing structures during the project (to increase the energy efficiency of the house).", "id": "1", "value": "True" }, { "title": "Reconstruction of engineering systems", "description": "Reconstruction of engineering systems (ventilation, recuperation) to increase the energy efficiency of the house (during the project).", "id": "2", "value": "False" }, { "title": "Water heating system", "description": "Installation of a new water heating system (during the project).", "id": "3", "value": "False" }, { "title": "Heat installation", "description": "Installation of heat installations to ensure the production of heat from renewable energy sources.", "id": "4", "value": "False" }]</pre>
Example request	CURL <pre>curl -X 'POST' \ 'http://host_ip>:8888/service_1/inference'</pre>



	<pre>-H 'accept: application/json' \ -H 'Content-Type: application/json' \ -d' {"building_total_area": 351.6, "reference_area": 277.4, "above_ground_floors": 3, "underground_floor": 0, "initial_energy_class": "D", "energy_consumption_before": 106.04, "energy_class_after": "B"}</pre>
--	---

Enershare AI4EF API: service 2 endpoint	
Description	Get buiidling parameters service 2
HTTP Method	POST
Endpoint URL	<host_ip>::8888/service_2/inference
Parameters	No parameters
Output example	<pre>[{ "title": "Electricity produced by solar panels", "description": "The amount of electricity produced by the solar panels, which are installed in the project.", "id": "5", "unit": "[MWh per year]", "value": "7.45" }, { "title": "Primary energy consumption after (KW)", "description": "Primary energy consumption after installing the solar panel system.", "id": "6", "unit": "[MWh per year]", "value": "0.45" }, { "title": "Reduction of primary energy consumption",</pre>





		<pre> "description": "Reduction of primary energy consumption: Difference between primary energy consumption before and after.", "id": "7", "unit": "[MWh per year]", "value": "11.18" }, { "title": "CO2 emissions reduction", "description": "The amount of CO2 emissions reduction in the project.", "id": "8", "unit": "[tons of CO2 per year]", "value": "0.81" } </pre>
Example request	CURL	<pre> curl -X 'POST' \ 'http://<host_ip>:8888/service_2/inference' \ -H 'accept: application/json' \ -H 'Content-Type: application/json' \ -d ' {"region": "Rīga", "electricity_consumption_of_the_grid": 4.65, "primary_energy_consumption_before": 11.63, "current_inverter_set_power": 0.0, "inverter_power_in_project": 10} </pre>

4.5.3.1.2 Database API

Enershare AI4EF postgREST API (EF_comp)	
Description	Get EF_comp entry
HTTP Method	GET
Endpoint URL	<host_ip>:8080/#/efcomp/get_efcomp_entry
Parameters	Project Number
Output example	<pre> [{ "Project number": "PME2-9999", "Region": "Rīga", "Granted support": "4000", </pre>





		<pre>"Electricity consumption of the grid": "4.65", "Primary energy consumption before": "11.63", "Current inverter set power": "0", "Inverter power in project": "10" }]</pre>
Example request	CURL	<pre>curl -X GET \ 'http://<host_ip>:8080/efcomp/get_efcomp_entry' \ -H 'accept: application/json' \ -H 'Content-Type: application/json' \ -d '{"title": "PME2-9999"}</pre>

Enershare AI4EF postgREST API (Sol_pan_comp)		
Description	Get Sol_pan_comp entry	
HTTP Method	GET	
Endpoint URL	<host_ip>:8080/## solpancomp /get_ solpancomp _entry	
Parameters	Project Number	
Output example	<pre>[{ "Project Number": "PME2-9999", "building_total_area": 351.6, "reference_area": 277.4, "above_ground_floors": 3, "underground_floor": 0, "initial_energy_class": "D", "energy_consumption_before": 106.04, "energy_class_after": "B" }]</pre>	
Example request	CURL	<pre>curl -X GET \ 'http://<host_ip>:8080/ solpancomp / get_ solpancomp_entry' \ -H 'accept: application/json' \ -H 'Content-Type: application/json' \</pre>



```
-d ' {"title": "PME2-9999"}'
```

4.5.4 Integration plan with the Data Space

The use case is expected to contribute to the Data Space by providing access to the aforementioned datasets and/or a summary of the inputs (ML model parametrisation) / outputs (ML model predictions) of the AI4EF service. Figure 88 describes the integration plan with the Data Space for the current use case. The diagram shows the envisioned structure for the integration of various services into the Enershare Data Space, highlighting the strategic use of the IDS Connector for data interoperability and security.

The central component of this framework is the IDS Connector, a key element that facilitates the integration of different services, including AI4EF and most importantly its main database which allows for storage of datasets and simulation results, while offering an OpenAPI for their provision to the Data Space. These services will be designed to connect seamlessly with data sources and sinks within the data space, using the IDS Connector to exchange data and metadata. Some of the components of the Enershare IDS framework could be integrated at a later stage depending on demand and availability. Note here that the integration plan might be reduced to one of the two connections presented in the schema most probably containing only the anonymised datasets.

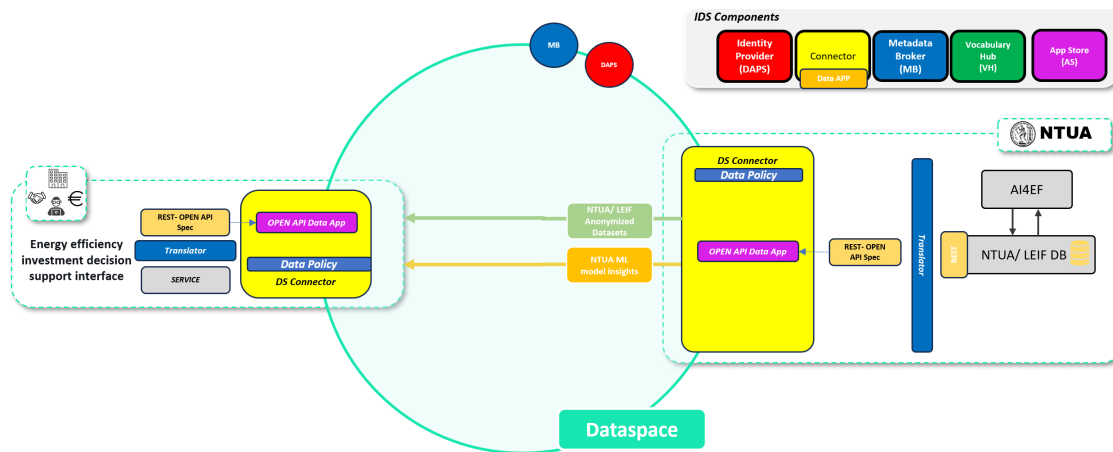


Figure 88: Data Space integration plan for AI4EF and pilot 7

The International Data Spaces specifies the interactions taking place between the different components in an IDS ecosystem:



1. **Onboarding**, i.e., what to do to be granted access to the International Data Spaces as a Data Provider or Data Consumer. Pilot 7 will use the Identity provider of the Data Space to obtain the certificates as Participant and for the connectors.
2. **Data Offering**, i.e., offering data or searching for a suitable data. Through the Open Api the different services will be exposed. The pilot 7 connector provider will register in the Metadata Broker of the Data Space and in this way their data, access conditions and how to call it will be part of the Enershare catalogue and the rest of the participants could search and use this information.
3. **Contract Negotiation**, i.e., accept data offers by negotiating the usage policies. Pilot 7 will define and check data usage control.
4. **Exchanging Data**, i.e., transfer data between IDS Participants. Pilot 7 will use an IDS connector to exchange the data.

The process will be as follows:

- A participant of the Data Space will query the Metadata broker to see the available services/datasets and their restrictions.
- A participant of the Data Space wants to use one service of pilot 7.
- Through their connector consumer, they will call to pilot 7 connector provider.
- Then a contract negotiation will be held and if a contract agreement is reached, the data will be exchanged. All the interchange messages must be logged into the Clearing House.
- Sometimes a service will not be able to obtain the expected result immediately because some algorithms must be called and the processing time of them could be huge. In this case, the connector of the Pilot 7 will send a notification message to the connector of the Participant when the data is available.

4.5.4.1.1 MVP deployment

For the next version of the MVP, a REST service is being developed. This first version of the REST API contains a call that lets the user obtain the ML model insights of one of the two AI4EF subservices given a set of input parameters. The output of the REST service will be a JSON object with the output parameters. The REST endpoint will be exposed to the Data Space through a TSG connector.

The next steps for the subsequent MVP releases will include the following tasks:

- Evolve the REST API to include all the required methods and services.
- Register the connector in the metadata broker.
- Define the Contract conditions of the different services and data.



4.5.4.1.2 Internal Deployment

AI4EF development

AI4EF is being developed as an AI driven tool to allow for the preliminary estimation of the benefits that can come from energy efficiency investments (i) service 1: Energy efficiency measures recommender; ii) service 2: PV investments profit estimator).

Database development

A Postgres database has been set for storing the datasets to be used by AI4EF alongside the ML model predictions to enable their exploitation by stakeholders and investors of the building sector. These results will be provided by the Data Space.

Development of the Open API

The REST service will be developed according to the Open API and data model defined in WP3. The developed REST API will be exposed to the Data Space through an IDS connector to ensure interoperability and secure access to the data.

4.5.5 Next steps

As next steps we will consider the following:

- Dataset augmentation (GAN, CT-GAN, Smote, oversampling) and class imbalance regarding ML model development.
- Full Data Space integration, as described in section 4.5.4, in terms of interoperability and data sharing.
- Front-end, back-end refinements.

4.6 Health insurance alarms for senior living alone

4.6.1 Development progress

The progress since D6.1 when the initially requirements and implementation strategies were defined are as stated below:

- The initial requirements have gone through several iterations to overcome limitations while adhering to the scope and expectations. During these iterations, system integration processes, parameters and alarm metrics for anomaly detection were re-evaluated and/or specified.



- The system architecture for implementation of the use case was conceptualized and designed leveraging on standard practises. The design took into consideration integration with data space and SEL's Living Energy platform.
- Some of the components and services identified in the system architecture design are now being developed.
- Aside Random Forest algorithm that was originally anticipated to be used for modelling, SEL began exploring other algorithms as a substitute or for an ensemble to improve accuracy and performance.
- The process for securely storing and accessing personal data of Senior Citizen that will be gathered was defined in compliance with GDPR, while also leveraging on existing methodologies within SEL.
- Data required to identify habits to compliment the model accuracy and anomaly detection were formulated. Relevant questions to obtain these data were then created and administered during one-on-one interview conducted with Senior Citizens. Data collected were grouped and securely stored.
- Internal data model and ontology for data was adopted leveraging on existing approaches having in mind ease of integration with the data space connector.
- Data from energy sensors are now being collected and stored in real-time.

The current TRL of the service is considered at TRL 5 since some tests need to be performed and necessary API interfaces will be released to allow interaction with end-users and potential clients.

4.6.2 Implementation details

The implementation of the use case by SEL is done using several interconnected services. Most of the services are developed as packages that can be containerized to run in suitable environment. Similarly, many of the services are built on python taking advantage of vast standard python libraries already available. All services are deployed on EC2 instances on AWS. The system consists of two main parts: data acquisition and operations management. Each of these parts are described in the following:

Data Acquisition

This includes services responsible for the collection of measurement data including storage, transformation, and making available for retrieval or for use by other components. Measurement data are collected at different periodicity from installed energy sensor devices in the homes of the Senior Citizens. The system architecture of this part is shown in Figure 89, and the various services are briefly described thereafter.



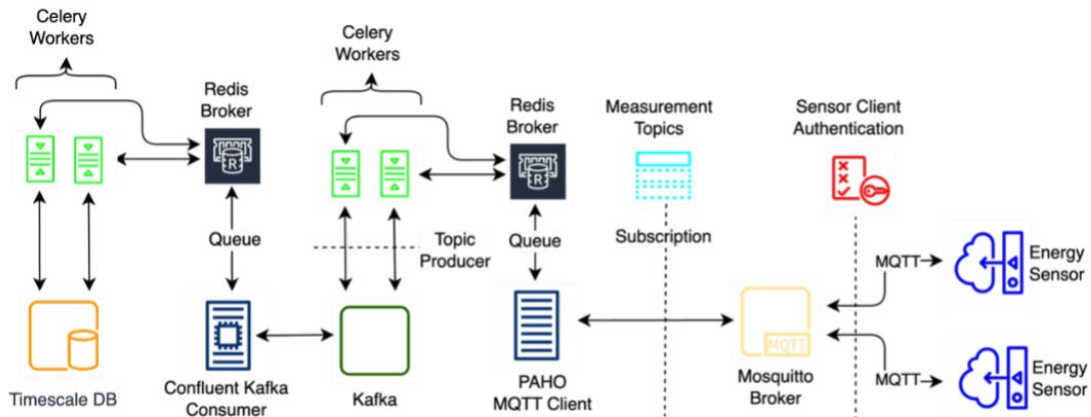


Figure 89: Data acquisition system architecture

MQTT Broker: Measurements are published from the installed sensors over defined topic to SEL’s dockerised mosquito MQTT v3 broker provisioned on an EC2 instance on AWS with message sent as encrypted over TLS with a QoS of 1. Authentication of publisher clients is done using username and password and the broker enforces topic restrictions to which publications can be made to. This combination enforces high level of security.

Subscriber Client: A dockerised subscriber client implemented using python PAHO MQTT subscribes to measurement topics on the MQTT broker. Received payloads re-maps the payload on-the-fly to expected format through a queuing system over a celery using Redis broker. These then are passed to a Kafka producer which then publishes it to a Kafka topic for temporarily storage where it can be consumed by other services.

Kafka: Different measurements and data reside here for varying period. Producers for different topics write to it from various services. Similarly, consumers consume data over various topics from it to external services.

Data Logger: This service implements the python confluent Kafka library to consume measurement data from Kafka and stores in a database over a celery queuing system using Redis broker.

Operations Management

This includes services for collection of non-measurement data, model training, event/anomaly detection, monitoring, and notification. The non-measurement data include data collected from through interviews and questionnaires (activity) administered to the Senior Citizen, Caretakers and or Relatives. The system architecture of this part is shown in Figure 90 and the components/services are explained thereafter.



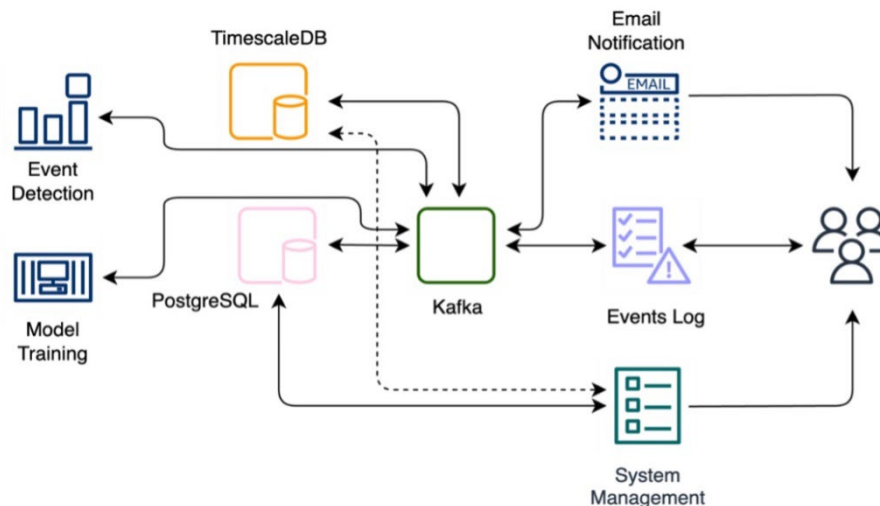


Figure 90: Operations management system architecture

System Management: The system management takes advantage of the Living Energy (LE) back-office management platform and provides functionalities for:

Installation and setting-up of energy sensors

Troubleshooting and monitoring of energy sensors

Senior Citizen setup and management

User (including Senior Citizen and Administrators) information entry for non-measurement data (like preference, interview response, questionnaire response)

The system management enforces authentication for use, role-based permissions for access, encryption at storage level for personal data and secure access only over https.

Database: The system implements the PostgreSQL database for non-measurement data. For measurement data (timeseries), it uses Timescale DB plugin leveraging on PostgreSQL base.

Model Training

This component which is currently being developed is responsible for training model with historical data gathered from measurement and non-measurement data. It implements a service for timeseries analysis using ARIMA and anomaly detection using Statistical Anomaly Detection and Random Forest. Some other machine models are also being tested. The model is



trained automatically in batch over a 24-hour cycle for each Senior citizen. The resulting model parameters per Senior citizen is sent to Kafka via producers over specified topics.

Event Detection

This component which is also currently being developed runs some event detection services that detects and logs possible anomaly or pattern changes using trained model parameters obtained from Kafka, non-measurement data and real-time measurement data. Detected pattern changes are logged on Kafka.

Notification

Logs generated from event detection of pattern changes are notified internally as they happen and to care providers of the Senior citizens where it has been opted for. Notification is mainly through email. This component relies on the event detection component to become fully operational.

Visualisation

The visualisation component provides views for inspecting measurement data, model results, event detection logs. This service is provided partly by the Living Energy (LE) backoffice and a dockerised instance of Grafana dashboard.

Reference Python Libraries

- Confluent-kafka: <https://pypi.org/project/confluent-kafka/>
- PAHO MQTT: <https://pypi.org/project/paho-mqtt/>
- Mosquitto Docker: <https://github.com/eclipse/mosquitto>



4.6.3 Integration with the Data Space

Some of the services developed being dockerised can easily be adapted as DAPPs in the data space. Particularly the Model Training service will be upgraded to allow inputs via http endpoints and output to an endpoint and as such allowing it to be configured for use on the data space. Besides this, using the connector, measurement data and event detection data can be exchange via the data space in accordance with the data space regulations. The system architecture of the approaches for integration described are summarised in Figure 91.

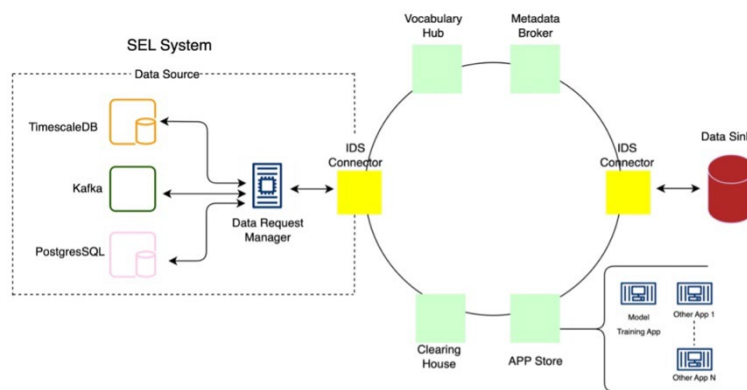


Figure 91: System diagram for “health insurance alarms for senior living alone” service integration with the Data Space

As depicted, the Model Training app will be made available in the App store so it can be deployed by any entity in the data space. SEL system will provide real-time and historic data via the IDS connector. When data request is sent, upon verification and authentication in accordance with internal policies and data space rules, the request is routed to a data request manager that fetches the data requested. Real-time data are fetched from the Kafka service while historic request is sent to a queue where it is processed, and data is either retrieved from the Timescale DB or PostgreSQL.

4.6.4 Next steps

The planned next steps are to:

- Continue to gather and monitor real-time data from energy sensors in Senior Citizen homes such that problems are swiftly identified and resolved, ensuring continuous stream of data.
- Complete the development of the model training component described in section 4.7.3. SEL is currently improving the models and exploring other models for better performance. Moreover, as more data is collected, more options can be explored,



- asides re-parameterisation may be required. Hence, this component is expected to evolve over time with improvements to enhance performance and accuracy.
- Complete the development of the event detection component described in section 4.7.3. This component relies on the model training component to be fully operational and may require minor modifications with changes in the model training component.
 - Develop API documentation for the model training component as soon as the component becomes quite stable such that it can be deployed as a DAPP on the data space for use by any service consumer.

4.7 Appliances maintenance or retrofit

4.7.1 Development progress

Functionality

Residential NILM (non-intrusive load monitoring) algorithms are designed to identify the energy consumption of each individual appliance within each house or building (which aggregated is equal to the total consumption) by analysing overall electricity consumption data, eliminating the need for sensors on each device. Using a single meter, NILM distinguishes energy consumption patterns by analysing voltage and current waveforms. Its applications include detailed 'Energy Monitoring', allowing users to understand contributions to the electricity bill, and 'Energy Conservation' by identifying energy-intensive appliances. NILM also serves as a 'Fault Detection' system, signalling potential malfunctions. In 'Home Automation', it integrates with smart home systems for intelligent control based on real-time energy usage. Additionally, NILM aids in 'Billing and Tariff Optimization', offering accurate billing and tailored tariff structures. Overall, NILM enhances energy management and infrastructure efficiency through monitoring, conservation, fault detection, home automation, and billing optimization.

The current TRL of the service is considered at TRL 5 since some tests need to be performed and necessary API interfaces will be released to allow interaction with end-users and potential clients.

General architecture

The residential NILM runs within the SEL platform. Moreover, the main flow of information for the real-time function of the tool is that the gateway of the residence (and building in general) will post the real-time consumption data to the SEL Data platform. Then, the residential NILM algorithms will get this data from the SEL Data platform, process it, and then post the results again to the SEL Data platform to be available for other services.



The development of the NILM algorithms is based on the Light Gradient Boosting Machine (LGBM), which provides a more powerful, thus efficient machine learning algorithm, and minimizes the errors of previous/older models by adding new models that correct the residuals, gradually improving the overall prediction. LGBM is a type of gradient boosting framework algorithm, has proven to be effective for NILM algorithm performance requirement due to its speed, accuracy, ability to handle large datasets and real-time applications. It builds a strong predictive model by combining the predictions of multiple weak models, often decision trees. It uses a histogram-based learning method for tree building, which reduces the amount of memory required, which makes this model faster.

Typically, the aggregated power consumption in residential households is measured from a single point and none of the appliances are equipped with metering devices. Therefore, the approach was to develop a non-intrusive model that can cope with the complexity of an installation of smart meters at every single home appliance, to create as little disturbance as possible to the occupants. Moreover, installing multiple smart meters is not viable economically in most cases. On the contrary, disaggregating energy from a single-point meter using AI methods, is a low-cost solution that can be deployed with minimum external interference into the user's residence. Therefore, the NILM algorithms were designed to be cloud-based to get by the constraints presented previously.

Additionally, and leveraging from the deployment of individual metering devices and several appliances, SEL is developing the algorithms based on the general (aggregated) and individual power consumption of different energy consumption profile households, which tailors and trains the algorithms to match the required results.



The proposed architecture for energy disaggregation focuses on a series of transformer blocks, as illustrated in Figure 92.

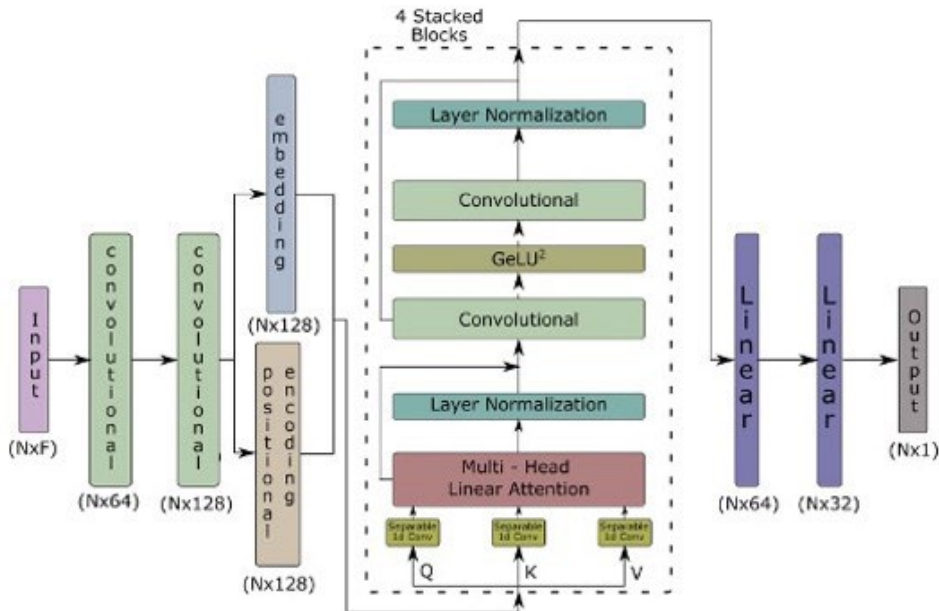


Figure 92: The proposed transformer-based architecture for energy disaggregation

Complementing these blocks are convolutional layers, positional encoding layers, and embedding modules. Notably, the conventional scaled dot-product are replaced by linear attention to preserve predictive accuracy while reducing computational time. To enhance disaggregation accuracy, 1D depth-wise separable convolutions follow the creation of queries (Q), keys (K) and values (V) matrices in each self-attention sublayer, which significantly improves predictive accuracy.

In lieu of a Feed-forward neural network (FNN), the two 1D convolutional layers were set for a kernel of 1. The first layer with dimension d_{cc} precedes the second, which outputs a d_{model} -dimensional vector, maintaining consistent dimensions between the input and output of each transformer block. Additionally, a squared Gaussian Error Linear Unit (GeLU) serves as an activation function within the modified pointwise two-layer convolutional network, several studies have indicated that GeLU and its variants can enhance the predictive accuracy of transformer architectures.

The overall network topology, depicted in Figure above, outlines each layer alongside its dimensions. The initial layer represents the input holding the aggregate signal, denoted by N for sequence length and F for the number of features. Following the input layer, two 1D convolutional layers map input feature vectors to d_{model} -dimensional vectors, extracting features from the input sequence. Due to the absence of a recurrence mechanism, positional



embedding and encoding preserve position information. The resulting vector is fed into the transformer block. Post the fourth transformer block, the output enters the final feed-forward network with three layers, reducing output signal dimensionality. Rectified Linear Unit (ReLU) activation functions are applied between linear layers. The architecture produces a final representation, a sequence N , matching the input sliding window's size.

Selecting the input sequence length is a crucial hyperparameter. Most existing approaches estimate or empirically select this length; thus, this method employs the Akaike Information Criterion (AIC) to accurately determine the optimal input sequence length. AIC, a model selection criterion, calculates the Kullback-Leibler distance between models. Specifically, the AIC is used to find the optimal sequence length. Experiments with the LGBM algorithm estimate the sub-metered signal, demonstrating high performance in predictive accuracy and training time. AIC is leveraged to determine the optimal sequence length through iterative experiments with varying length values, performed using the LGBM regressor.

Ultimately, the sequence length N in which the AIC metric reaches its minimum value, corresponds to the optimal sequence length, which has its analytical procedure described in Figure 93.

Algorithm 1 Optimal Window Length Estimation Pipeline

```

1:  $min \leftarrow 0$ 
2:  $pos \leftarrow 0$ 
3: for  $k \leftarrow length_{min}, length_{max}$  do
4:   Create  $X_{train}$  values from  $data$ .  $X_{train} = X_{1:t_0+k}$ 
5:   Train the Regressor  $LGBM(X_{train}, Y_{train})$ 
6:   Compute  $Y_{pred}$  by the  $LGBM$  for test data:
      $Y_{pred} = LGBM(X_{test_{1:t_0+k}})$ 
7:   Calculate  $MSE$  for  $Y_{test}, Y_{pred_k}$ :
      $MSE_k = \frac{1}{N} \sum_{n=1}^N (Y_{test} - Y_{pred_k})^2$ 
8:   Compute AIC for  $k$  and test set samples  $N$ :
      $AIC_k = 2k - N \log(MSE_k * N)$ 
9:   if  $pos = 0$  then
10:      $min \leftarrow AIC_k$ 
11:      $pos \leftarrow k$ 
12:   else
13:     if  $min > AIC_k$  then
14:        $min \leftarrow AIC_k$ 
15:        $pos \leftarrow k$ 
16:     end if
17:   end if
18: end for
19: return  $k$ 

```

Figure 93: Optimal window length estimation pipeline



The minimum and maximum sequence lengths were set to $\text{length}_{\min} = 50$ and $\text{length}_{\max} = 650$, respectively, with a step size of $\text{step} = 1$ (600 iterations). LGBM's efficiency facilitates a detailed investigation of the optimal solution, balancing high accuracy with low computational cost.

Internal architecture

The service's architecture will consist of the backend service that will contain the NILM algorithms and functions for requesting data from the SEL Data platform and a local database used to store / load the trained models. Additionally, we will add a separate API of this service to make data available to third parties when necessary.

4.7.2 Documentation updates

The API of the service under development shall not be documented here, as it is a test version that will change upon successful Data Space integration. Additionally, the current communication of the service will change, since now it is internal data used to run the service but upon Data Space integration the service will be fed by data provided by external data providers which will be the service buyers.

4.7.3 Tests of the services with the Data Space

This service will be tested with the Dataspace during the first semester of 2024, no testing was done so far.

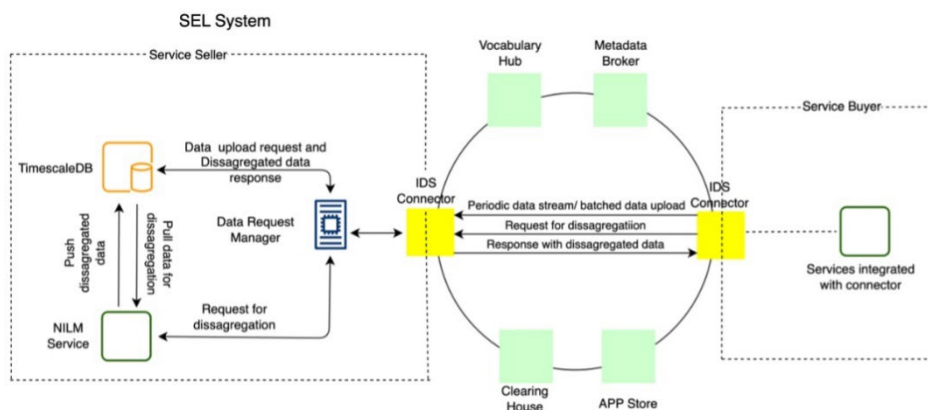


Figure 94: System diagram of the integration of service “appliances maintenance or retrofit” with Data Space

However, as shown in Figure 94, the planned test should allow real-time streaming of anonymized data or upload of historic data with unique identifier by a service buyer for the purpose of disaggregation. The service buyer can then at any point request for a disaggregation



of streamed or uploaded data. Upon completion of the disaggregation, the disaggregated data is sent to back to the service buyer.

4.7.4 Next steps

Presently, the NILM pseudo-code algorithms, as outlined earlier, are undergoing development and customization to align with the project requirements. Multiple tests are currently in progress, and an initial set of performance evaluation results is anticipated by the mid of 2024.

Following the analysis of the initial results, the subsequent steps will be evaluated and specified. This assessment will involve a comprehensive process, encompassing a minimum of 4 months for data pre-processing and algorithm training, followed by no less than 6 months dedicated to algorithm validation, optimization, and the collection of Key Performance Indicators (KPIs)

5 Data Visualisation

This chapter aims to describe the Data Visualisation Layer / Visualisation Engine (VE), which is developed in the context of Task 6.4 – “Toolkit for interactive data visualisation services”. For the realisation of this Task, Apache Superset⁵ has been selected to serve as the basic visualisation dashboard and reporting tool. Thus, in the following subsections, the focus will be put on presenting an overview of the Visualisation Engine functionalities and capabilities, as well as the envisioned way of integrating the tool with the various datasets deriving from the project’s activities and other implementation details related to the deployment approach and the overall solution that will be offered.

5.1 Visualisation Engine demo

In the following sections, some of the main capabilities of the tool will be described, with an emphasis on the flexibility and extendibility offered by its connection with external databases, the file uploading procedure, which enables users to visualise their data, the feature for creating dashboards, which allows various queries and charts to be grouped together, offering a holistic

⁵ <https://superset.apache.org/>



overview, and the integrated user and role management system, which ensures security throughout the implementation.

5.1.1 Connection with external databases

One of the main advantages of VE is its versatility, allowing connection with a wide range of external databases. Superset supports a multitude of databases, including but not limited to MySQL⁶, PostgreSQL⁷, Oracle⁸, and more, allowing users to establish connections and harness the data within these repositories. Beyond traditional databases, Apache Superset also works seamlessly with modern tools such as Presto⁹, Trino¹⁰, and Apache Druid¹¹. This means users can quickly analyse data across different sources using high-performance distributed querying with Presto and Trino, while also benefiting from Apache Druid's real-time analytics capabilities for speedy insights into event-driven data.

To connect with an external database in VE, the user first needs to navigate to the **"Data"** menu and select **"Database Connections"**. Clicking on the **"+DATABASE"** button on the top right of the window, a modal window (Figure 95) opens. Here, after choosing the database type, users need to connection details, such as database type, hostname, username, and password. Once connection is established, users explore, analyse, and visualise data from the connected external database using VE's user-friendly interface.

⁶ <https://www.mysql.com/>

⁷ <https://www.postgresql.org/>

⁸ <https://www.oracle.com/database/>

⁹ <https://prestodb.io/>

¹⁰ <https://trino.io/>

¹¹ <https://druid.apache.org/>



Connect a database
✕

STEP 2 OF 3

Enter the required PostgreSQL credentials

Need help? Learn more about connecting to PostgreSQL..

HOST * ?

PORT *

DATABASE NAME *

Copy the name of the database you are trying to connect to.

USERNAME *

PASSWORD

DISPLAY NAME *

Pick a nickname for how the database will display in Superset.

ADDITIONAL PARAMETERS

Add additional custom parameters

SSL ?

SSH Tunnel ?

BACK
CONNECT

Figure 95: Visualisation Engine - Database connection details required

5.1.2 File uploading

Uploading files into VE is an easy process after connecting to a database. Users initiate the file upload by pressing the “+” button and navigating, via the “Data” menu item, to the “**Upload CSV to database**” page. There, the users are prompted to choose the target database and table where they want to store the uploaded data, as displayed in Figure 96.



CSV to Database configuration

CSV Upload *	<input type="button" value="Choose File"/> No file chosen <small>Select a file to be uploaded to the database</small>
Table Name *	<input type="text" value="Table Name"/> <small>Name of table to be created with CSV file</small>
Database	<input type="text" value="examples"/> ▼ <small>Select a database to upload the file to</small>
Schema	<input type="text" value="Schema"/> <small>Select a schema if the database supports this</small>
Delimiter *	<input type="text" value=","/> ▼ <small>Enter a delimiter for this data</small>

▼ File Settings

▼ Columns

▼ Rows

Figure 96: Visualisation Engine - File upload form

Except from uploading files via the VE, users are also given the option to upload them via a REST API endpoint we have developed, using FAST API¹², a modern, fast (high-performance), web framework for building APIs. This REST API endpoint has been deployed on NTUA’s premises and can be accessed through the following link: http://enershare.epu.ntua.gr:8000/docs#/default/upload_csv_upload_csv_post. It allows the user to choose the file to be uploaded, as well as the name of the table it will be uploaded to. In case a table with the specified name does not exist, it is created in the database. In case a table with the specified name already exists, the file is stored in this existing table. All the aforementioned files are stored in a PostgreSQL database we have also deployed and connected to the VE instance.

5.1.3 Dashboard creation

Apart from creating individual charts, Apache Superset offers the capability of creating dashboards, where various charts can be grouped, empowering users to simultaneously display

¹² <https://fastapi.tiangolo.com/>



various visualisations of their data, thus transforming those complex datasets into clear and exploitable insights.

In the context of Task 6.4, and with the purpose of thoroughly describing and presenting the VE, a dashboard with various charts, visualizing data from the project’s pilot 7, has been created. Dashboards can also be separated into sections that group the different charts, based on the data/insights they represent. As a result, three sections were created, namely **“Data Exploration”**, **“Energy Efficiency Improvements”** and **“Energy Sources & Emissions Factors”**. Part of the page is presented in Figure 97:

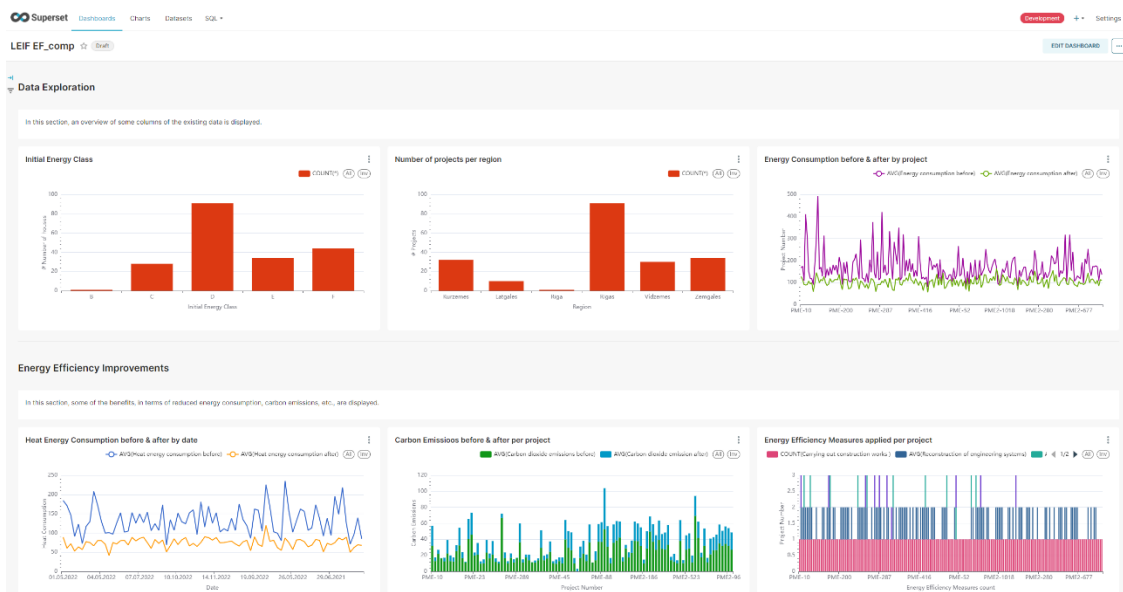


Figure 97: Visualisation Engine - Dashboard with pilot 7 dataset

This dashboard displays all the charts that have been created. Upon creating a new chart, the user can save it to a dashboard, by choosing one of the existing ones. The users can also edit the order of the displayed charts, add new sections, along with their descriptions, while they can also navigate to the dedicated page of any chart (by pressing on the chart’s card), where they are given the option to alter the performed query and edit the chart. This interface has been described in the previous deliverable, D6.1 - Federated learning, data-driven services, data visualisation and Digital Twins, and more specifically in Section 5.2.

The users can navigate to the Dashboards page, via the **“Dashboards”** link on the header of the page. There, a list of the existing dashboards is displayed, while the users can add a new one by pressing the **+ DASHBOARD** button on the top right of the page. Following, they are navigated to a new page, where they are given the option to edit the name of the newly created



dashboard, create new charts for it, or even choose already stored charts to be displayed in this new dashboard.

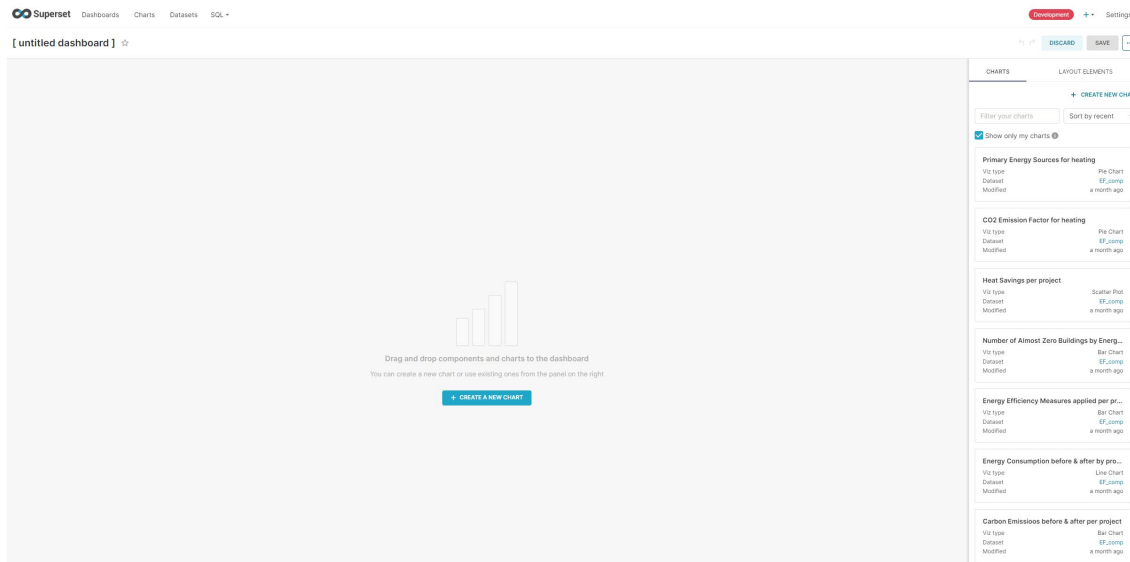


Figure 98: Visualisation Engine - New dashboard page

5.1.4 User and Role management

VE also offers a comprehensive user and role management system, which ensures secure and efficient data access and governance within the data analytics environment. It incorporates various authentication methods, including LDAP and OAuth support, allowing it to accommodate diverse authentication requirements. Additionally, Single Sign-On (SSO) integration is supported.

The Role-Based Access Control system of Apache Superset allows the users to create precise access permissions regarding their assets. Specifically, predefined roles (Admin, Editor and Viewer) can be assigned to the users, while custom roles can also be created if needed. Granular permissions cover actions such as dashboard creation, database management, and dataset definition. All the User and Role management actions can be easily performed via VE's user-friendly interfaces.

5.1.5 Deployment configuration

An instance of Apache Superset has been successfully deployed on NTUA's premises. Leveraging Docker¹³ containers ensure a consistent and portable environment, enhancing the ease of

¹³ <https://www.docker.com/>



deployment across diverse systems. The encapsulation of VE and its dependencies within containers facilitates a modular and isolated deployment process. This deployment has been orchestrated using Docker Compose¹⁴, where a YAML¹⁵ file defines the necessary services, networks, and volumes, simplifying configuration and ensuring reproducibility. With this approach, the deployment is not only scalable but also easily replicated across different environments, promoting a consistent and reliable setup. Official documentation on how to deploy it on the website of Apache Superset, so the process followed for the needs of ENERSHARE is the most safe and reliable one.

Depending on the needs of the project, either the existing deployment of VE will be used by all the interested parties (user and role management capability, which will be analysed in one of the following sections, is a valuable asset that will help towards this direction), or each one will be provided with deployment guidance in order to deploy it on their premises.

Regarding the PostgreSQL database and backend application with the REST API endpoint for file uploading, they have also been deployed using Docker and Docker Compose. In fact, they are deployed using one docker-compose file, which is shown in Table 13:

¹⁴ <https://docs.docker.com/compose/>

¹⁵ <https://en.wikipedia.org/wiki/YAML>



Table 13: docker-compose file for PostgreSQL DB and FAST API backend

```
version: '3.8'
services:
  postgres_db:
    image: postgres:latest
    container_name: PostgresCont
    restart: always
    environment:
      - POSTGRES_USER=*****
      - POSTGRES_PASSWORD=*****
    ports:
      - '5556:5432' # Expose port 5556 on the host
    networks:
      - my_network

  fastapi_app:
    build:
      context: .
      dockerfile: Dockerfile
    container_name: FastAPICont
    restart: always
    ports:
      - '8000:8000'
    depends_on:
      - postgres_db
    networks:
      - my_network

networks:
  my_network:
    driver: bridge
```

5.1.6 Connection with Data Spaces

As already mentioned in Section 5.1.1, one of the most important features of VE is its capability of connecting to a wide variety of external databases. This feature offers numerous benefits, allowing the end users to consolidate data from disparate data sources into a single platform, promoting data integration for comprehensive analysis of the data and reducing the need for different tools and interfaces.

Incorporating VE's database connectivity features can significantly enhance decision-making processes, while also improving data accessibility and integration and also enabling cross-database analysis, support of real-time insights, and offering extensibility and reusability.



There is thorough documentation of how to connect Apache Superset with various databases, such as MySQL¹⁶, PostgreSQL¹⁷, Oracle¹⁸, Presto/Trino¹⁹ and Apache Druid²⁰.

Cross-platform data analysis allows users to join, transform and visualise data from multiple databases into a single dashboard, while Apache Superset's support for both traditional and real-time analytics engines (like Apache Druid) offer the flexibility to derive insights from both historical and real-time data and the users can seamlessly combine these features to create dashboards that offer holistic perspective on their data assets.

For ENERSHARE pilots / technical partners / stakeholders that desire to produce visualisations using the NTUA Visualisation Engine within the IDS context, an implementation of the relevant Data Space connection is required from their side according to the specifications of EnerShare (TSG connector). Subsequently, given that the connection at the data consumer side has been developed to translate the datasets in question to one of the data storage options supported by the NTUA VE, then NTUA will be involved to deploy the VE at the consumer side, hence enabling visualisations and data analysis as a service for the related end-users. Additionally, the already presented centralized version of the Visualisation Engine will be always available at NTUA's premises in case the data storage locations of stakeholders are publicly available, allowing the direct cloud connection with the tool. Said cloud connection requires the deployment of a TSG connector from the end-user of VE, that can handle the OpenAPIs related to the databases supported by VE. A good example of that is Postgrest²¹, which directly exposes a Postgres database to OpenAPI. The above-described options are depicted in Figure 99.

¹⁶ <https://superset.apache.org/docs/databases/mysql/>

¹⁷ <https://superset.apache.org/docs/databases/postgres/>

¹⁸ <https://superset.apache.org/docs/databases/oracle/>

¹⁹ <https://superset.apache.org/docs/databases/presto/>

²⁰ <https://superset.apache.org/docs/databases/druid/>

²¹ <https://postgrest.org/en/stable/references/api/openapi.html>



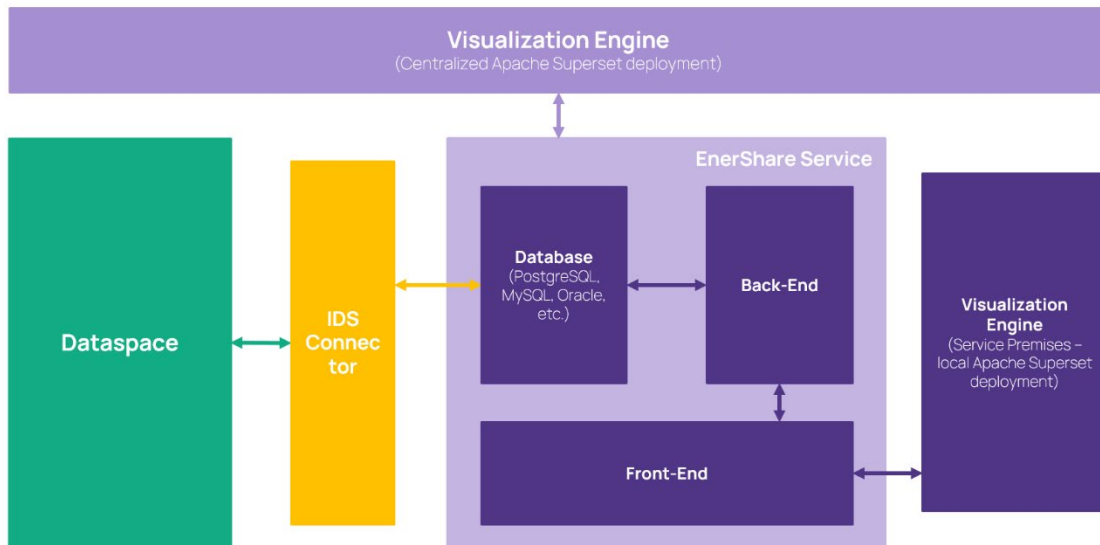


Figure 99: Visualisation Engine connection schema

5.1.7 User Testing and Feedback

The Visualisation Engine will be tested with a sample of users once a demonstration version is available. Testing will be conducted remotely using Teams. The following methodology draws from relevant references to propose a robust approach for sampling, preparation, task execution, and result analysis.

Sampling

The methodology will adhere to the principles outlined by (Virzi, 1992), who emphasised the importance of refining the test phase of usability evaluation. To ensure the representativeness of the sample, a diverse group of business users with varying levels of experience and expertise in data visualisation and energy data will be selected. The sample size will be determined based on the recommendations of (Faulkner, 2003), who highlighted the benefits of increased sample sizes in usability testing, ensuring that the methodology captures a wide range of perspectives and experiences.

Specifically, in the case of the ENERSHARE project, the user sample should include users who are specialists in data analysis but also those who will use it from a management perspective without substantial experience of manipulating, analysing or visualising data.

Preparation



Prior to the user testing, participants will be provided with clear instructions and access to the data visualisation demonstration. The preparation phase will align with the principles of user-centred design and participatory design, as advocated by Tan & Bishu (2002) and (Scandurra et al., 2008). This will involve engaging participants in the design process and ensuring that the software interface meets their needs and expectations.

Part of the preparation will include identification of the users, their profiles and known behavioural characteristics, and a collaborative review of the software to be tested to ensure that the testing team is able to run effective usability testing sessions.

Tasks and Scenarios

During the user testing, participants will be presented with a series of tasks and questions designed to evaluate the usability and effectiveness of the data visualisation program. The tasks will be carefully crafted to assess the program's ability to facilitate understanding of data from the European energy data space. This approach aligns with the recommendations of Saputra et al. (2022), who emphasised the importance of usability testing in estimating user interface effectiveness.

Tasks should be relevant to the type of user engaged in the usability test and should be scenario-based in order to optimally root them in reality. For example, an administrator working for a Transmission System Operator (TSO) needs to review energy usage data through visualisation to identify consumption patterns and trends, assess grid stability, and make informed decisions regarding energy distribution and management. Therefore, a scenario for the testing could be: "A change in energy demand in one of your regions has been detected. How might you use the Visualisation Engine to review the relevant data and understand the nature of the change?"

Result Analysis

The results of the user testing will be analysed using a combination of quantitative and qualitative methods. This approach is consistent with the recommendations of Faulkner (2003), who highlighted the value of both quantitative and qualitative data in usability testing. The analysis will focus on identifying patterns, user preferences, and areas for improvement within the software interface. Additionally, the methodology will draw from the principles of participatory design and collaborative research, as outlined by Dufendach et al. (2017), to involve participants in the analysis process and gather their feedback on the results.

Utilisation of Results

The results will inform iterative design improvements, aligning with the principles of user-centred design and agile software development, as highlighted by Valente et al. (2016). Insights



gained from these usability sessions will be reported to the work package leaders for consideration. Where interface changes, or other conceptual changes, can be made to improve the usability and utility of the data visualisation solution, they should be costed and prioritised under the success criteria for this work. Any highlighted issues from the user testing that cannot be resolved during the project should be noted for future iterations and improvements.

Timeline

Once the user-friendly demonstration of the Visualisation Engine is made available for testing, there follow approximately 2 months of preparation between SIN, ENVIRODUAL and NTUA to create a test plan and to recruit the user sample.

Appointments for user sessions could take 4-6 weeks depending on participant availability. A further 2 weeks will be required for analysis of the findings and 2 weeks for generating reports.

5.2 Tiny spatial visualisations

5.2.1 Description of the service

When it comes to visualizing irregularly spaced spatial data several approaches are available, each with its own set of advantages and disadvantages. One option is to display all the data points on a map complete with their associated attributes. While this method ensures a detailed representation of the data it can become problematic when dealing with a large volume of data points. This is because the map can become cluttered and overwhelming making it difficult to discern patterns or insights.

An alternative to this approach is the application of spatial aggregation. Spatial aggregation involves the consolidation of data points based on their geographic proximity or similarity in attributes. This method effectively reduces the amount of data being displayed allowing for a more comprehensible and visually appealing representation of the data. By aggregating data one can transform a dense cloud of points into a clear and informative visual summary providing a different and often more insightful view of the data.

Our service offers a subset of spatial aggregation methods tailored to suit various needs and objectives. These methods include:

- Kernel smoothing
- Grid aggregation
- Contour lines or polygons



By offering these diverse methods of spatial aggregation our service enables users to choose the most appropriate visualisation technique for their specific dataset and analytical needs. Whether it's for environmental monitoring, urban planning, or any other field where spatial data plays a key role our tools provide a flexible and powerful way to make sense of complex spatial information.

Service can be accessed at: <http://enershare.epa.si:20989/viz>

5.2.2 Functions

All described functions below operate on irregular 3D points (x,y,z) where z coordinate is considered as value aggregated over. In addition to attributes each function expects data in JSON format: as object with three numerical arrays: { "x": [...], "y": [...], "z": [...] }.

Latest API documentation is available at: <http://enershare.epa.si:20989/docs/>

5.2.2.1 Kernel smoothing

This technique involves the creation of a smooth surface over the map to represent the Z-value of data points. It is particularly useful for highlighting areas of high values and identifying underlying trends.

Endpoints:

1. Calculation (“/viz/ks2d”)
 - *Description:* kernel-based smoothing of irregular 3D points.
 - *HTTP method:* POST
 - *Media type:* image/tiff (geotiff)
2. Testing sample (“/viz/ks2d/preview”)
 - *Description:* sample preview of kernel-based smoothing of irregular 3D points.
 - *HTTP method:* GET
 - *Media type:* image/png

Testing sample example:



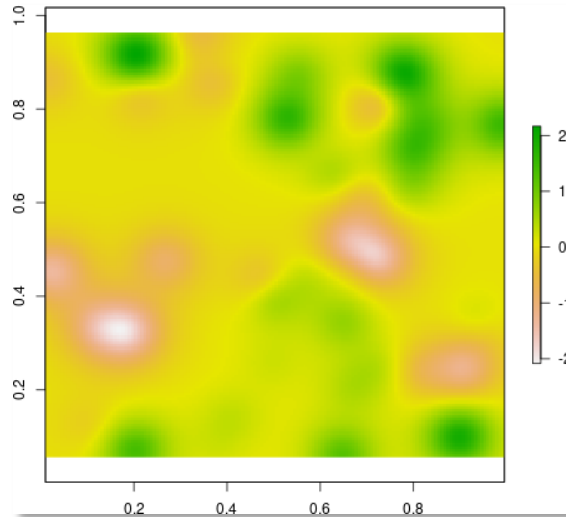


Figure 100: Testing sample example 1

5.2.2.2 Grid aggregation

In this method the area of interest is divided into a grid and data points within each grid cell are aggregated. This approach simplifies the data visualisation by converting numerous data points into a manageable number of cells each representing the aggregated value of the points within it.

Endpoints:

1. Calculation (“/viz/grid”)
 - *Description:* aggregation of irregular 3D points on a grid.
 - *HTTP method:* POST
 - *Media type:* image/tiff (geotiff)
2. Testing sample (“/viz/grid/preview”)
 - *Description:* sample preview of aggregation of irregular 3D points on a grid.
 - *HTTP method:* GET
 - *Media type:* image/png

Testing sample example:



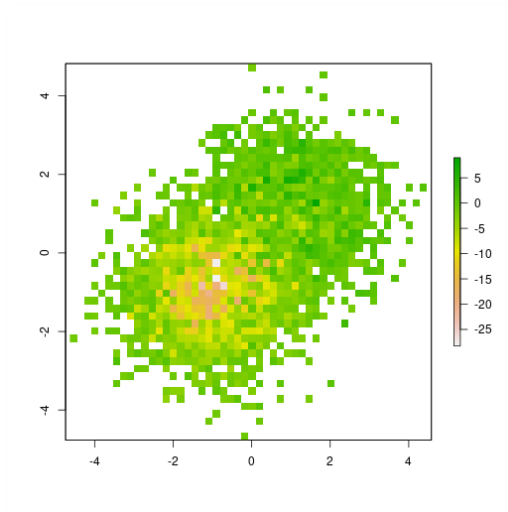


Figure 101: Testing sample example 2

5.2.2.3 Contour lines or polygons

This method is used to create contour lines or polygons that represent levels of a particular variable across the space. It is an effective way to visualize gradations or changes in data values over a geographic area such as elevation, temperature, or pollution levels.

Endpoints:

1. Contour lines (“/viz/ks2d/lines”)
 - *Description:* contour lines of kernel-based smoothing of irregular 3D points.
 - *HTTP method:* POST
 - *Media type:* application/zip
2. Contour lines testing sample (“/viz/ks2d/lines/preview”)
 - *Description:* sample preview of contour lines of kernel-based smoothing of irregular 3D points.
 - *HTTP method:* GET
 - *Media type:* image/png
3. Contour polygons (“/viz/ks2d/polygons”)
 - *Description:* contour polygons of kernel-based smoothing of irregular 3D points.
 - *HTTP method:* POST
 - *Media type:* application/zip
4. Contour polygons testing sample (“/viz/ks2d/polygons/preview”)



- *Description:* sample preview of contour lines of kernel-based smoothing of irregular 3D points.
- *HTTP method:* GET
- *Media type:* image/png

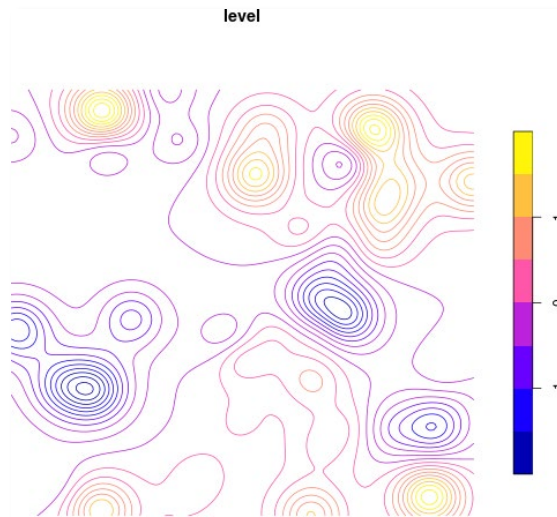


Figure 102: Contour lines testing sample example

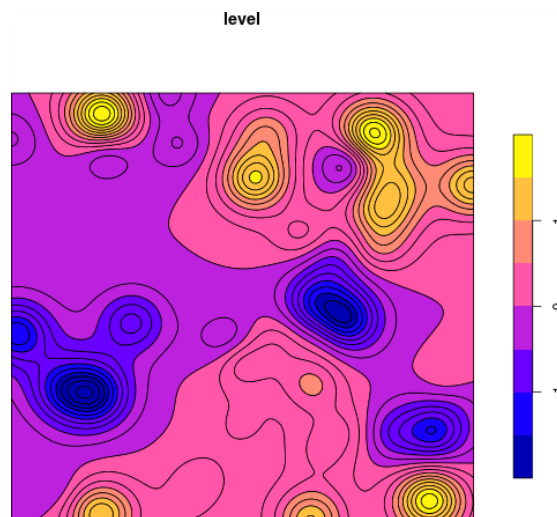


Figure 103: Contour polygons testing sample example

5.2.3 Connection with Data Space

As part of the integration plan for incorporating visualisation service into the Data Space, we will utilize the TNO TSG Connector as a standardized interface for secure data exchange. This connector will facilitate the interoperability between our service and other services within the Data Space.

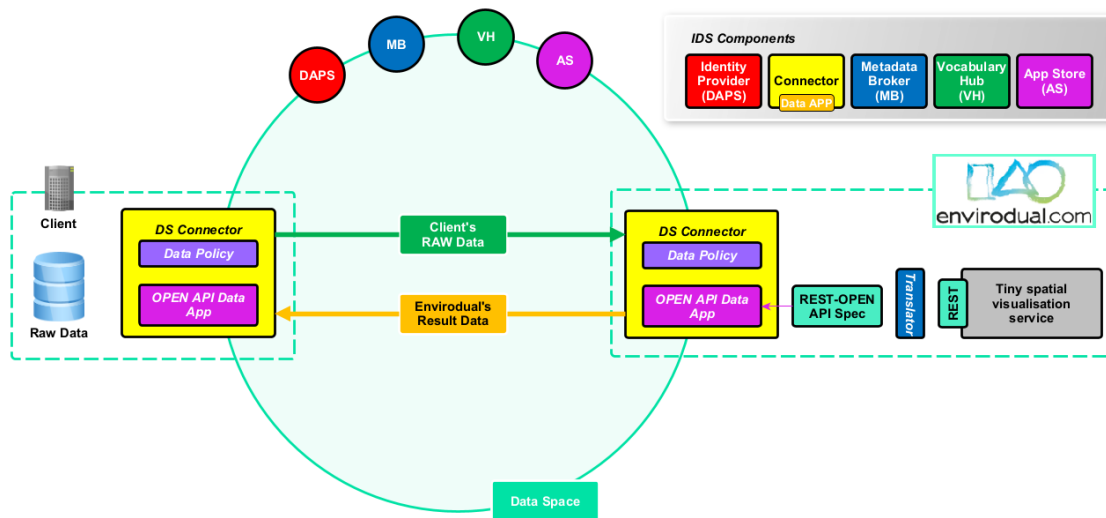


Figure 104: IDS integration schema

5.2.4 Next steps

Current service deploys small subset of spatial methods. Next steps involve integrating additional methods like kernel density estimation, k-nearest neighbours spatial smoothing and simple interactive webpage for previewing methods on user's data. The final stage also involves integrating the service into the data space.



6 System-of-systems Integrated Digital Twins

This chapter is an overview of the status of the development of the digital twin services in ENERSHARE, that were described in deliverable D6.1:

- Digital Twin for optimal data-driven Power-to-Gas optimal planning
- Digital Twin based O&M algorithms and generation of synthetic failures data
- Digital Twin for flexible energy networks

A summary of the current and future envisioned developments is shown in Table 14.

Table 14: Overview of the different digital twins releases in WP6

Digital Twin	Features in deliverable D6.1 (Alpha version)	New features in D6.2 (Beta version)	Future features for D6.3 (Final version)
Power-to-Gas optimal planning (NTUA)	<ul style="list-style-type: none"> -Architecture description - First simulation results - Data sources identified - Data ingestion specifications extracted 	<ul style="list-style-type: none"> - First integration of simulator with DB - Inclusion of real data and costs in the simulation - Data ingestion mechanism MVP - First prototype of front-end - API docs 	<ul style="list-style-type: none"> - Final version of data ingestion mechanism and DB - Full interconnection with T6.4 (visual analytics) - Simulation results through the data space
Wind turbine Digital Twin (TECNALIA)	<ul style="list-style-type: none"> Preliminary architecture description 	<ul style="list-style-type: none"> - Follow-up of failure diagnosis independent services - More detailed architecture description - API docs 	<ul style="list-style-type: none"> - Creation of the DT by means of integration the failure diagnosis services of gearbox, electric generator, and hydraulic pitch system in ENERSHARE Data Space
Flexibility planning in electrical networks (RWTH)	<ul style="list-style-type: none"> - Description of the architecture identification of the potential requirements (users, roles, functions) 	<ul style="list-style-type: none"> - Basic configurations and parametrizations of the toolset - Selection of the best candidate technologies for the implementation of a deployable solution 	<ul style="list-style-type: none"> - First Proof-of-Concept of the software solution stack - API docs - Interconnection with services from other WPs - Data ingestion and Data Space integration



The current development of a minimum viable product of the dataspace is also a driving force in the planning of the integration of services. A high-level description considering the major interactions is mentioned in each one of the services, together with a concept of the software architectural design.

6.1 System-of-system Digital Twins: ENERSHARE Approach

With the progressive availability of the components of the MVP of the ENERSHARE Dataspace, there is a clear path forward for the approach of System-of-system Digital Twins using this Data Space. Table 15 summarizes aspects that we would like to test and demonstrate with the integration of the services implemented as part of this task, and their interconnection to the Data Space.

Table 15: Digital Twin and Data Spaces aspects

Aspect	Target for evaluation in System-of-Systems Digital Twins with Data Spaces
Integration Scope	Simulates behaviors, interactions, and dependencies among multiple subsystems. Facilitates this integration by standardizing the interfaces and data models.
Data Integration & Interoperability	Aggregates and harmonizes data from diverse sources for a comprehensive system view.
Modelling & Simulation	Provides accurate simulations using real-time/historical data, understanding system impacts.
Scenario Analysis & Prediction	Enables scenario testing, predicting outcomes, and assessing interventions across systems.
Decision Support	Offers comprehensive insights for informed decision-making, resource allocation, and risk management.
Continuous Improvement	Facilitates ongoing learning, adaptation, and identification of emerging patterns within systems.
Security & Resilience	Decentralizes monitoring, anomaly detection, and response mechanisms for enhanced system resilience.



6.2 Digital Twin for optimal data-driven power-to-gas optimal planning

The high-level architecture of TwinP2G, as shown in Figure 105, follows a Platform-as-a-Service (PaaS) design meant for multiple user roles. It comprises various subcomponents that utilize state-of-the-art technologies. The architecture and functionalities have been also described in (Pelekis et al., 2023), however a brief and updated version is included in the current document as well. Note here that the descriptions there are not binding for the implementation and several things have changed.

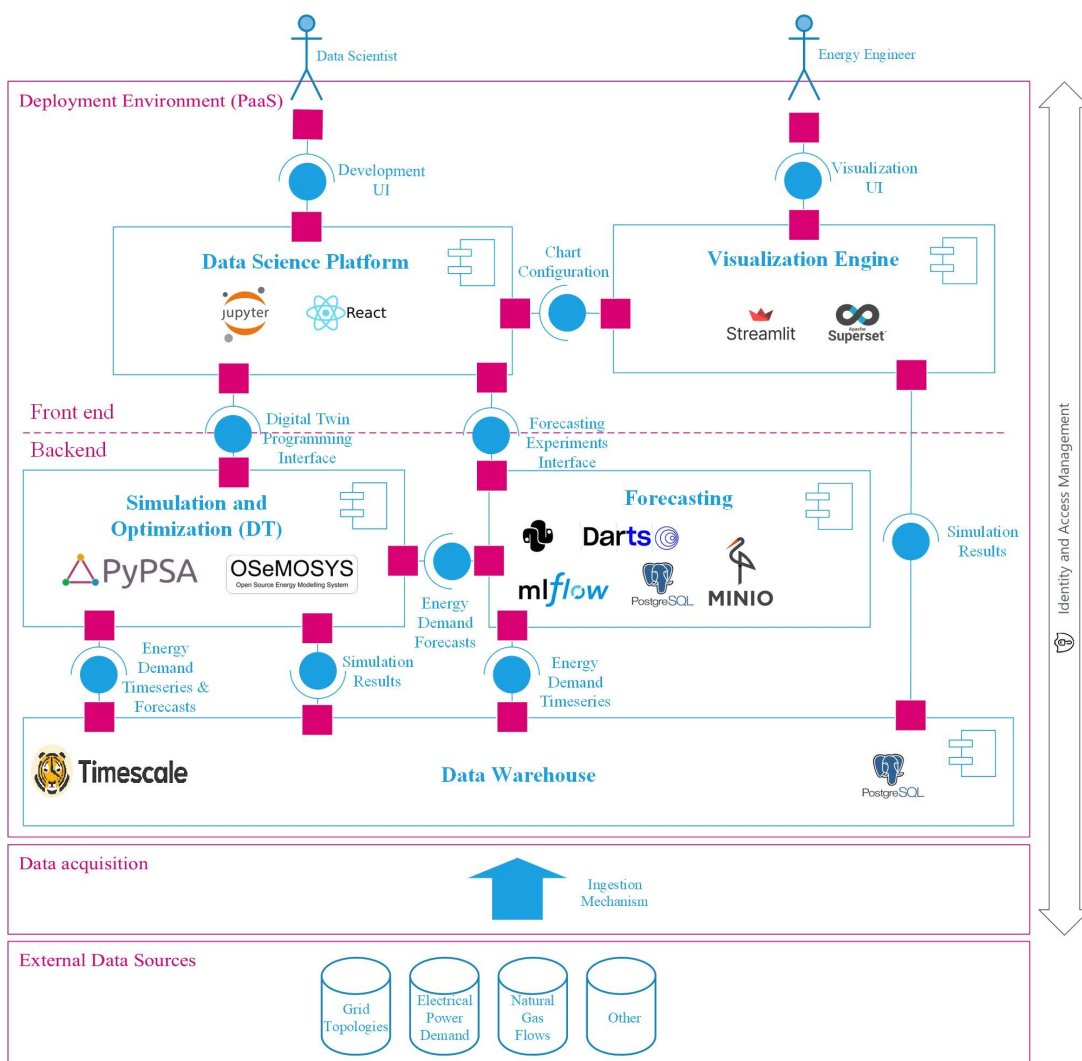


Figure 105: The high-level architecture of TwinP2G

These components of the architecture have been conceptualized and extensively described in D6.1. The current deliverable describes the development progress and changes since then.



ENERSHARE has received funding from [European Union's Horizon Europe Research and Innovation programme](#) under the Grant Agreement No 101069831

6.2.1 Changes since deliverable D6.1

The components of TwinP2G have evolved in terms of maturity as follows:

- **Data warehouse and ingestion mechanism:** fully conceptualized – from initial development stages (D6.1) to finalized (D6.2)
- **Simulation and optimization** fully conceptualized (D6.1) -> first prototype developed (D6.2)
- **Forecasting toolkit:** developed in I-ENERGY project – pending integration with the DT (no progress – awaiting final versions)
- **Front-end:** to be developed (D6.1) -> first prototype developed (D6.2)
- **Identity and access management:** to be developed (no progress – awaiting final versions)

More specifically, compared to D6.1 and the architectural setup of our initial TwinP2G publication (Pelekis et al., 2023), several developments have been performed by NTUA including:

- Development of a Timescale PostgreSQL database²² for the storage of IPTO, ENTSO-E, DESFA, and ENTSO-G datasets that are useful for the simulations conducted within TwinP2G. The database uses Dagster²³ for workflow orchestration in alignment also with T8.2. More details can be found in the next subsection.
- Development of new simulation scenarios including hydrogen or methane flows to the natural gas grid rather than only fuel cell models that were limited to the electricity grid.
- Implementation of a front-end application based on Streamlit²⁴ to enable enhanced end-user experience for triggering simulations and inspecting them through interpretable visualisations. More details can be found in section 0 of the current document.

Two dedicated Github repositories of TwinP2G (simulation²⁵ and database²⁶) have been created by NTUA and can be referred to by the reader for the code related specifics of our developments.

²² <https://www.timescale.com/>

²³ <https://dagster.io/>

²⁴ <https://streamlit.io/>

⁶ <https://www.kaggle.com/datasets/jeanmidev/smart-meters-in-london>

²⁵ <https://github.com/eu-ntua/enershare-twinp2g>

²⁶ <https://github.com/eu-ntua/enershare-twinp2g-db>



6.2.1.1 Database development

The data warehouse fulfils several roles fundamental to the integrity of the project. First, it acts as a source of truth for the historical data fetched from power and gas transmission operators, namely IPTO²⁷ and ENTSO-E²⁸ for power, as well as DESFA²⁹ and ENTSO-G³⁰ for gas. The data is ingested periodically from the sources mentioned and as such is up to date. Second, it acts as a repository for simulation results. Finally, it provides access to its data by serving it through an HTTP API.

6.2.1.1.1 Database architecture

The database system consists of several components, which are deployed in containerized form via Docker. They are as follows:

- **Database.** As all the relevant data being stored is time-series data, we use a database suited to their nature, namely TimescaleDB³¹. TimescaleDB is an extension to PostgreSQL that adds various helpful resources for dealing with time-series data, such as functions helpful in querying and analyzing time-series data and data compression mechanisms, while retaining PostgreSQL's familiar syntax, strong consistency, and data integrity features. It also acts as a normal PostgreSQL instance and as such can be used to fulfil storage needs for other components of the architecture.
- **Data Orchestrator.** This refers to the software used to schedule, run, and monitor the various data pipelines that are essential in keeping the data warehouse up to date. We use Dagster³² for this purpose, a tool written in python that satisfies all our requirements. The Dagster instance also contains the ETL logic (in Python) responsible for consuming data from the external sources to the database. It schedules the data pipelines to run daily, with sensible retry policies, and provides adequate logging and monitoring capabilities. It exposes a web UI that can be used to examine the health of the data pipelines and run them at will.
- **API.** All of the relevant data are served via an API that interface with the database. PostgREST³³ is used for this purpose, a tool that serves an HTTP API, exposing the

²⁷ <https://www.admie.gr/>

²⁸ <https://transparency.entsoe.eu/>

²⁹ <https://desfa.gr/>

³⁰ <https://transparency.entsog.eu/>

³¹ <https://www.timescale.com/>

³² <https://github.com/dagster-io/dagster>

³³ <https://github.com/PostgREST/postgrest>



desired datasets from the database in JSON format with minimal overhead. It also provides an OpenAPI specification.

6.2.1.1.2 User Interface of the database system

- Pipeline monitoring:

The users of the data warehouse are data engineers. They can access Dagster’s web UI to monitor the data pipelines.

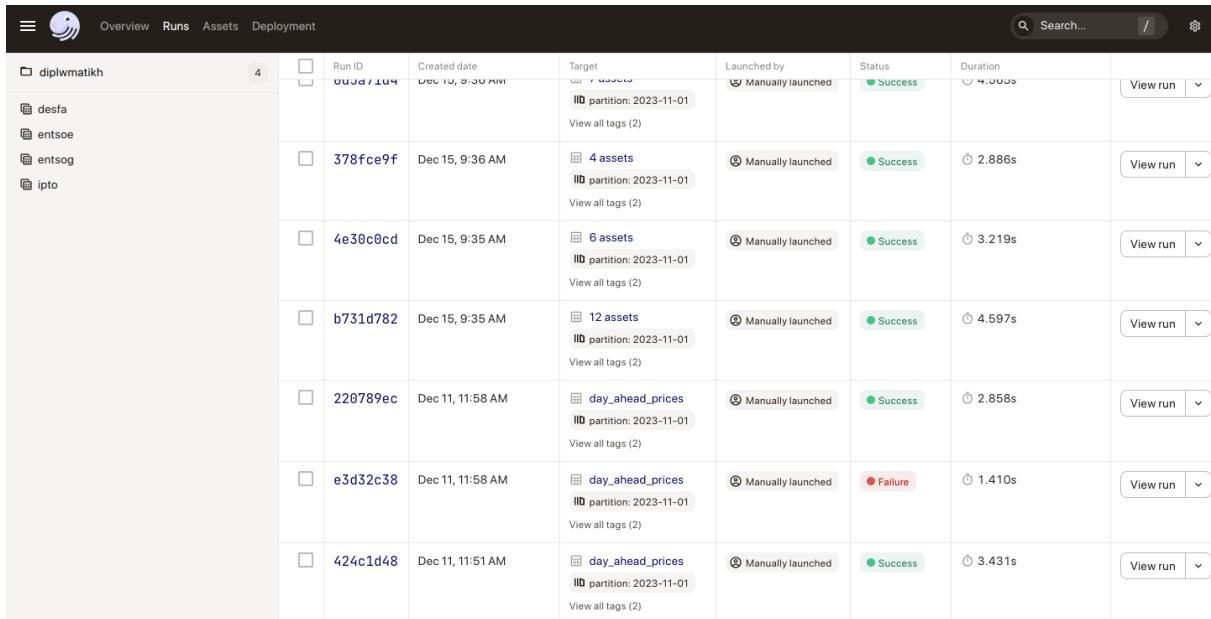
Upon entering the web UI the presented page is as illustrated in Figure 106.

Group name	Missing	Failed/Overdue	In progress	Materialized
entsoe diplwmatikh	0	0	0	12
desfa diplwmatikh	6	0	0	0
entsog diplwmatikh	4	0	0	0
ipto diplwmatikh	7	0	0	0

Figure 106: Landing page of the database system's web UI.

The landing page offers a quick overview of the various pipelines (or “assets”, in Dagster), grouped by source. Any unresolved failure will be highlighted here for quick access. We can click on a specific asset group and run any number of pipelines manually if we wish. By browsing to the “Runs” page via the top bar, we can view the individual runs as shown in Figure 107.



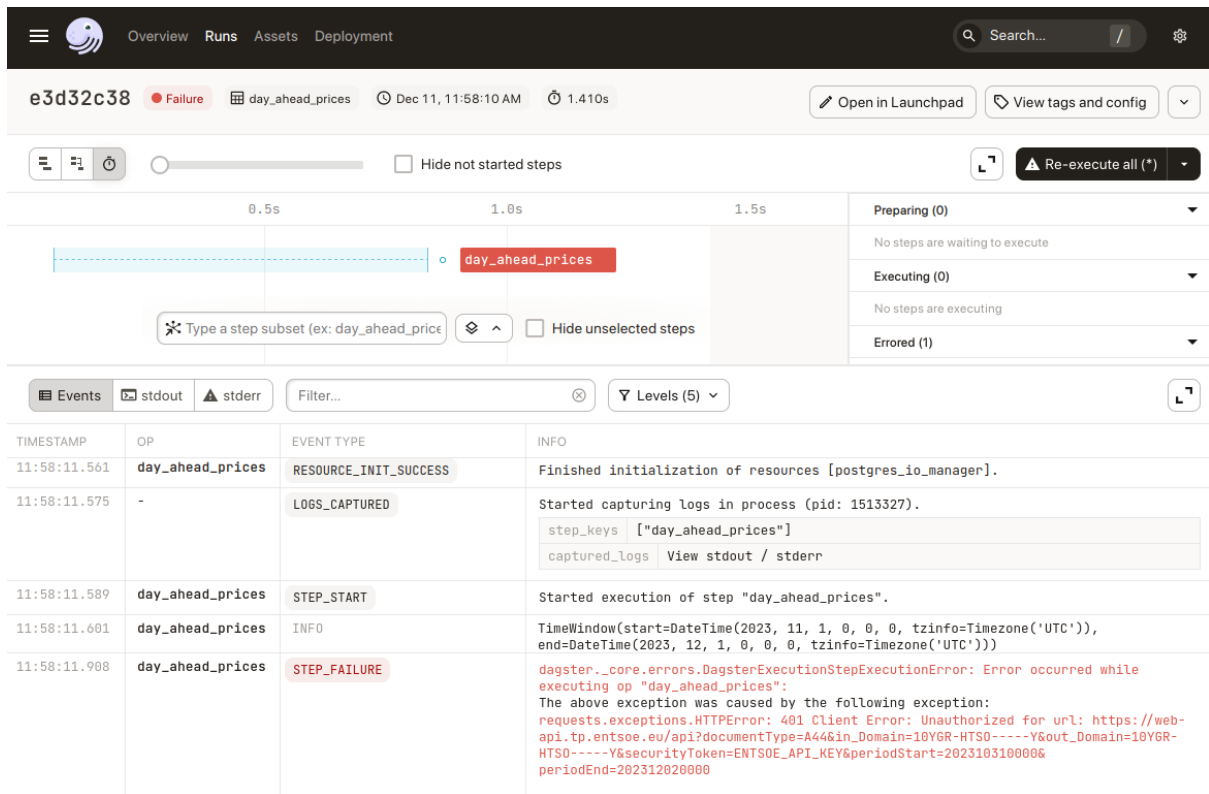


Run ID	Created date	Target	Launched by	Status	Duration
00007104	Dec 10, 9:00 AM	target: /... IID partition: 2023-11-01 View all tags (2)	Manually launched	Success	4.000s
378fce9f	Dec 15, 9:36 AM	4 assets IID partition: 2023-11-01 View all tags (2)	Manually launched	Success	2.886s
4e39c9cd	Dec 15, 9:35 AM	6 assets IID partition: 2023-11-01 View all tags (2)	Manually launched	Success	3.219s
b731d782	Dec 15, 9:35 AM	12 assets IID partition: 2023-11-01 View all tags (2)	Manually launched	Success	4.597s
220789ec	Dec 11, 11:58 AM	day_ahead_prices IID partition: 2023-11-01 View all tags (2)	Manually launched	Success	2.858s
e3d32c38	Dec 11, 11:58 AM	day_ahead_prices IID partition: 2023-11-01 View all tags (2)	Manually launched	Failure	1.410s
424c1d48	Dec 11, 11:51 AM	day_ahead_prices IID partition: 2023-11-01 View all tags (2)	Manually launched	Success	3.431s

Figure 107: Exploring the individual runs in Dagster

In case a previous run of a pipeline had failed, as pictured above, we can examine further by clicking on the individual run, which brings us to the page of Figure 108.





The screenshot shows the Dagster Run inspection page for a failed run. The top navigation bar includes 'Overview', 'Runs', 'Assets', and 'Deployment'. The run ID is 'e3d32c38', and it is marked as 'Failure'. The run name is 'day Ahead Prices', and it started on 'Dec 11, 11:58:10 AM' and lasted for '1.410s'. There are buttons for 'Open in Launchpad' and 'View tags and config'. Below the navigation is a progress bar and a 'Re-execute all (*)' button. The main area shows a timeline of the run with a red bar indicating the failure. A table below shows the events:

TIMESTAMP	OP	EVENT TYPE	INFO
11:58:11.561	day Ahead Prices	RESOURCE_INIT_SUCCESS	Finished initialization of resources [postgres_io_manager].
11:58:11.575	-	LOGS_CAPTURED	Started capturing logs in process (pid: 1513327). step_keys ["day Ahead Prices"] captured_logs View stdout / stderr
11:58:11.589	day Ahead Prices	STEP_START	Started execution of step "day Ahead Prices".
11:58:11.601	day Ahead Prices	INFO	TimeWindow(start=DateTime(2023, 11, 1, 0, 0, 0, tzinfo=Timezone('UTC')), end=DateTime(2023, 12, 1, 0, 0, 0, tzinfo=Timezone('UTC')))
11:58:11.908	day Ahead Prices	STEP_FAILURE	dagster._core.errors.DagsterExecutionStepExecutionError: Error occurred while executing op "day Ahead Prices": The above exception was caused by the following exception: requests.exceptions.HTTPError: 401 Client Error: Unauthorized for url: https://web-api.tp.entsoe.eu/api?documentType=A44&in_Domain=10YGR-HTSO----Y&out_Domain=10YGR-HTSO----Y&securityToken=ENTSOE_API_KEY&periodStart=202310310000&periodEnd=202312020000

Figure 108: Run inspection page in Dagster

In this page we can visualise the logs of the specific run which point out to the specific error encountered. Once the issue is debugged and fixed, we can either wait for the next scheduled run or execute the run again manually via the UI.

6.2.1.1.3 Data model

All datasets related to natural gas and electricity consumption and production in Greece have been integrated into the ENERSHARE semantic data model through continuous collaboration with WP3 and specifically T3.1 led by ENGIE. More details can be found in the related deliverables D3.1 and D3.2.

6.2.1.1.4 API

PostgREST generates an OpenAPI specification of the HTTP API that it serves. The data returned can be filtered through parameters in the HTTP request, as shown in PostgREST's



documentation. The API documentation of the TwinP2G database can be found online in the dedicated GitHub pages of the project ³⁴.

6.2.2 Experimental setup and challenges

This section describes in detail the implementation progress of the new TwinP2G features within NTUA's premises alongside the development progress of the P2G simulation algorithms and their respective results.

6.2.2.1 Additional P2G scenarios

In D6.1 we mainly tested P2G technologies that only involve energy exchanges among the power grid and hydrogen storage through electrolysis and fuel cell technologies. However, this time new simulations were carried out, extending this concept to the natural gas grid as well by introducing energy exchanges through hydrogen or synthetic methane injection. Six additional ten-year-simulations were conducted in particular, for every one of two different scenarios, and for every one of three different P2G configurations, under these two scenarios. The two different scenarios concerned the current 2023 situation, and a future 2033 situation.

- Scenario 1: 2023.
- Scenario 2: 2033 forecast.

The three different P2G configurations were:

- Configuration 1: "No P2G", or "Baseline". In this configuration, there is no coupling between the power and natural gas sectors, meaning no energy transfers between the electricity and natural gas grids.
- Configuration 2: "P2G1". In this configuration, produced electricity can be converted to hydrogen and vice versa, and the green hydrogen can then be converted to synthetic methane, which can then be injected into the natural gas grid for consumption.
- Configuration 3: "P2G2". In this configuration, electricity can again be converted to hydrogen and vice versa exactly like in P2G1. However, the produced green hydrogen can then only be directly injected into the natural gas grid for consumption.

In the next chapters, an explanation of the datasets used in the simulations follows, and then a more detailed description of the three configurations. Then an analysis of the parameters used in the simulations, and finally we present the results of the simulations, together with conclusions.

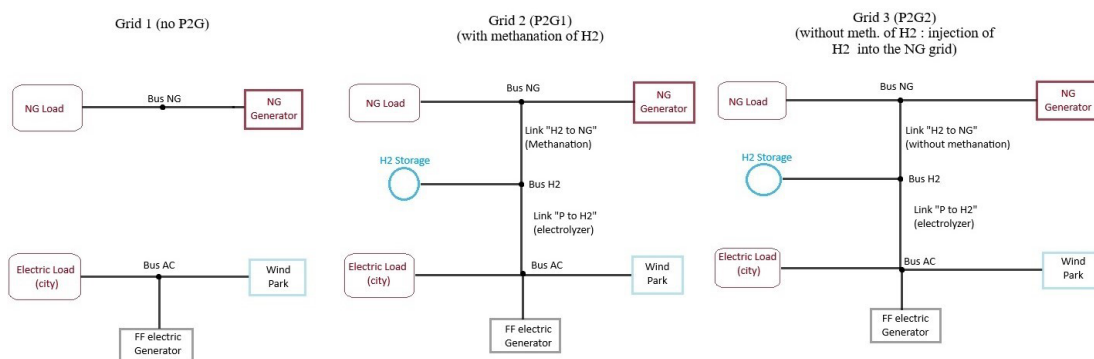
³⁴ <https://epu-ntua.github.io/enershare-twinp2g-db/>



6.2.2.1.1 Datasets

Electricity and natural gas demand timeseries in the new simulation scenarios concerned a hypothetical “average” city in Greece, of 50 thousand inhabitants, which served as a statistical representative of the seasonal behaviour of current electricity and natural gas demands in Greece. For this purpose, a randomized sample was drawn from a representative group of households, from the Kaggle “Smart meters in London” dataset³⁵. The aggregate daily electricity demand was then scaled to match the total yearly demand of 50k Greek citizens in 2021, assuming roughly similar seasonal patterns of demand. Natural gas demand was taken and respectively scaled, from real natural gas historic import data provided by DEPA. Both data timeseries covered the timespan of one year, and were then replicated ten times, to create simulated timeseries of ten years of electricity and natural gas demands. The two daily energy demand files created, were used as feedstock, for the respective energy demands to be met, in the TwinP2G model that was built.

The PyPSA³⁶ library in Python was used to configure the network, and then optimize it using linear programming techniques of Linopy³⁷ python library. Three different configurations were as mentioned earlier utilized, in order to investigate possible P2G system economic feasibilities, assess their economic potential and profiles, as well as compare their advantages and disadvantages. The three configurations are depicted in Figure 109. The experiments were executed on a personal computer with an Intel Core i7-11800H CPU @ 2.30GHz and 16 GB RAM and took 2-7 seconds to compute. Two scenarios were considered in the simulations. Scenario 1 was the current 2023 market and technology situation, and Scenario 2 was a projection for 2032, based on recent scientific and research literature on predictions of the prices and efficiencies of the components utilized.



³⁵ <https://www.kaggle.com/datasets/jeanmidev/smart-meters-in-london>

³⁶ <https://pypsa.readthedocs.io/en/latest/index.html#>

³⁷ <https://linopy.readthedocs.io/en/latest/>



Figure 109: Different P2G configurations

6.2.2.1.2 *TwinP2G Configurations*

6.2.2.1.2.1 Configuration 1: No P2G (Baseline)

In “Configuration 1: No P2G”, also called “Baseline” configuration, there is no P2G component, meaning no electricity and natural gas grids coupling. Electricity can either be generated by a “Fossil Fuel generator” (“FF electric Generator”), which is a statistical representation of fossil fuel powered electricity generation in Greece, based on the 2021 fossil fuel energy mix of Greece, connected to the electric “Bus AC” bus, or by a wind turbines farm (“Wind Park”), also connected to the same bus, which is a simulated wind farm, configured with the “Windpowerlib”³⁸ python library. Natural gas enters the national natural gas grid at “Bus NG”, and it is normally produced in the simulation by an artificial “natural gas generator” (“NG Generator”), which simulates natural gas imports or real production. The electric and natural gas loads of the virtual “typical” city are connected to their respective buses. Wind farms have been selected in this version of the deliverable to extend the capabilities of TwinP2G beyond the assessment of investments related to solar panels.

In all our simulations, the capex of the fossil fuel generator was assumed to be zero since these power plants are already present in reality. O&M generator costs in the simulator are split in a less rigid operational costs part, and a CO₂ emissions cost part, however both can easily be customized by the user or the simulator.

A customized objective function to be optimized was introduced in the simulation, representing the profit of the P2G facility administrator, or its income minus its expenses. The income of the administrator comes in the form of income from selling renewable wind power, and income from any P2G products, such as SNG and H₂ on the respective markets. The expenses of the administrator are any capital expenses and O&M expenses incurred from running the administrator’s power facilities and P2G facilities. All results concern ten-year-investment simulations.

6.2.2.1.2.2 Configuration 2: P2G1

In “Configuration 2: P2G1”, electricity can be converted to hydrogen through electrolysis at an electrolyser plant. The produced hydrogen can then be either directly converted further to SNG at a methanation plant, based on the Sabatier chemical process, taking captured CO₂ as feedstock, or stored at hydrogen storage facilities for future usage, when commodity prices of electricity, fossil fuels used in power generation, natural gas, and hydrogen itself, and weather conditions are more favourable. The produced SNG can then be injected to the national natural

³⁸ <https://windpowerlib.readthedocs.io/en/stable/>



gas grid. There are, however, specific natural gas methane content per volume constraints that need to be met, for the gas grid to function properly. More specifically, an upper content of methane in the gas grid was assumed, and taken at 95% v/v, however this parameter can be customized by the user, as can the initial content of methane in imported natural gas, here assumed at 85% v/v. Other customizable parameters are the heating values (LHV) of raw natural gas and methane/SNG and hydrogen, and all P2G components costs.

6.2.2.1.2.3 Configuration 3: P2G2

In the last “Configuration 3: P2G2”, hydrogen produced by the same electrolysis process can be directly injected into the natural gas grid for usage. Like the previous “Configuration 2”, there are H₂ content per volume constraints that need to be met, to ensure hydrogen and gas grid compatibility and safety. An upper content of 20% v/v in H₂ was assumed in the simulations, however this parameter can be customized by the user, as can the heating value of H₂ and all P2G components costs. A thorough overview about the maximum hydrogen content allowed in already existing natural gas grids, and its compatibility with common natural gas burning appliances, along with the challenges it presents can be found in (Erdener et al. 2022).

6.2.2.1.3 Selection of simulation parameters

Two different scenarios were considered, namely “Scenario 1: 2023”, and “Scenario 2: 2033”, to reflect how changes in simulations parameters can alter P2G economic profitability forecasts. The parameters used were based on recent research literature and for the future 2033 Scenario, on assumptions specified here, where applicable. Where multiple data sources were used, an average of them was taken to be used in the simulations. In particular, wind generators capital costs and O&M costs were based on (Stehly et al., 2020), adjusting capital costs and fixed operational costs for the duration of the 10-year simulation. O&M costs of wind production were also based on more detailed recent measurements by (Fraunhofer ISE, 2021). Fossil fuelled generators O&M costs were based on (Hafner, 2022). Electricity generation with fossil fuels in the simulations reflected the actual electricity production with the energy sources mix of Greece in 2021, namely natural gas, coal, and oil, in shares of 65.8%, 18.3% and 15.9% respectively, in the Greek electricity production mix of that year.

Electrolysis costs and specifications, such as PEM stack lifetime, as well as the future projections of them used, and other P2G components costs were based on the study by (Schiebahn et al., 2015), and also research such as (Mongird K. et. al., 2020), (Christensen A., 2020) and (Reksten, 2022). The respective values for the methanation facility were based on (v. Leeuwen et. al, 2018).

Hydrogen storage capital costs and storage facilities lifetime were assumed to be a statistical average consisting of 80% geological storage and 20% artificial storage, such as industrial



hydrogen containers, measured per total volume capacity. The costs were based on recent work gathered in (v. Leeuwen et. al, 2018) and (Papadias et al., 2021). O&M hydrogen storage costs were based on the work of (Forndal et al., 2022) and were negligible compared to the other simulation costs.

Natural gas pipelines compatibility with hydrogen content per volume, as well as hydrogen compatibility in natural gas consumer boilers and other currently widespread equipment using natural gas as a fuel is extensively discussed in works such as (Erdener et al. 2022).

Table 16 contains P2G and electricity generators' parameters used in both scenarios considered. It is noted that due to PyPSA library's structure, fixed O&M costs of any components had to be incorporated into the "capital costs" input parameters of the components, as they are incurred in a production-independent manner.

It is noted that a significant increase in CO₂ emissions costs between the current 2023 situation and 2033 was assumed, reflecting the greenhouse gas emissions reduction policies of the EU and of individual countries, and Greece in particular, to reduce greenhouse gas emissions. An increase from roughly 120 €/tone of CO₂ equivalent to 230 €/tone in 2033 was assumed in the simulations, comprising an increasing fraction of fossil-fuelled generation O&M costs, which can however be customized by the user of the simulator.

Selling prices of wind farm electricity produced, SNG and green hydrogen were assumed to be the averages of their respective markets for the Scenario 1 reflecting the current 2023 situation. Their respective selling prices for Scenario 2: 2033 were assumed to change depending on the other costs changes that comprise the production marginal costs for these products. Namely, wind farm produced electricity was assumed to be sold at a lower price than electricity from fossil fuelled power plants, reflecting its cheaper mode of production due to renewable wind power being cheaper than fossil fuelled electricity generation, and the fact that competition will be present among wind farm electricity producers. SNG selling price for Scenario 2 was assumed to be on average the same as in Scenario 1, plus CO₂ emissions costs of the SNG sold, since we have assumed an inclusion of natural gas products in the CO₂ pricing policy active, such as the EU's CO₂ allowances trading system. Green hydrogen selling price of Scenario 1 was assumed to be the average hydrogen price in 2023 in the EU market. Since in P2G2, green hydrogen is assumed to be used as an ingredient of natural gas that can be consumed just like natural gas without any hydrogen admixture, in other words, a competitor fuel to natural gas and SNG, its selling price for Scenario 2 was assumed to be the same as that of natural gas and SNG.



Table 16: P2G scenarios parameters

Simulation parameter	Scenario 1	Scenario 2	Sources
Generators			
EG Capex (€/MW)	0	0	Assumed to be zero
EG marginal costs (€/MWh)	236	290	(Hafner 2022)
Wind F.Capex(€/MW)	781000	726330	(Stehly et al., 2020)
WF marginal costs (€/MWh)	62	63	(Stehly et al., 2020), (Fraunhofer ISE, 2021)
NGG Capex (€/MW)	0	0	Assumed to be zero
NG marginal costs (€/MWh)	55	159	Greek regulatory authority for energy, waste, and water ³⁹
CO ₂ emissions costs (€/tone of CO ₂)	130	230	EU carbon permits prices ⁴⁰
P2G components			
Electrolyzer Capex (€/MW)	1636000	1145200	(Schiebahn et al., 2015), (Mongird, 2020), (Christensen 2020)
Electrolyzer marginal costs (€/MWh)	0	0	(Schiebahn et al., 2015), (Mongird, 2020), (Christensen 2020)
H ₂ storage Capex (€/MWh)	938	844.2	(v. Leeuwen et al. 2018), (Papadias, 2021)
H ₂ storage marginal cost (€/MWh)	0	0	(Forndal et al., 2022)
Methanation plant Capex (€/MWh)	800000	744000	(v. Leeuwen et al. 2018)
Methanation plant marginal cost (€/MWh)	4.46	4.24	(v. Leeuwen et al. 2018)
Fuel cell Capex (€/MWh)	2461000	1845000	(Forndal et al., 2022)
Electrolyzer efficiency	0.67	0.7	(Schiebahn et al., 2015), (Mongird, 2020), (Christensen 2020)
Methanation efficiency	0.75	0.77	(v. Leeuwen et al. 2018)
Fuel cell efficiency	0.63	0.68	(Forndal et al., 2022)
Selling prices (€/MWh)			
Electricity selling price	110	130	(Stehly et al., 2020), (Hafner 2022). Assumed to be on the lower end of electricity prices offered on the energy market, as compared to fossil generated electricity.
SNG selling price	55	159	Greek regulatory authority for energy, waste, and water
H ₂ selling price	120	159	CRU green hydrogen prices 2023 research ⁴¹

³⁹ <https://www.rae.gr/genika-nea/68983/>
⁴⁰ <https://tradingeconomics.com/commodity/carbon>
⁴¹ <https://sustainability.crugroup.com/article/energy-from-green-hydrogen-will-be-expensive-even-in-2050>


6.2.2.1.4 Results

In the following Table 17, key investment profitability indicators of the simulations can be seen. The Capex and O&M costs appearing on this table, concern only the P2G investor’s investments, namely only costs for wind generation installation, P2G components purchases, and the corresponding O&M costs thereof, for the entirety of each 10-year simulation. Fossil fuel generators’ Capex and O&M costs are not included in the table. The return on investment (“ROI”) index was computed, defined as the ratio of total simulation monetary return, that is, net income of the investment, divided by the investment’s Capex, assuming no inflationary effects. We observe that, even though P2G2 was the most profitable of the configurations under Scenario 2, the highest return on investment was observed when there is no P2G components installed, although with a very small difference from the P2G2 configuration.

Table 17: Simulations investment results

Scenario & configuration	Capex (in mil.€)	O&M costs (in mil.€)	Return (net income in m. €)	ROI
Scenario 1:2023, No P2G	52	295	33.8	65%
Scenario 1:2023, P2G1	52	295	33.8	65%
Scenario 1:2023, P2G2	52	295	33.8	65%
Scenario 2: 2033, No P2G	60	658	73.9	123.2%
Scenario 2: 2033, P2G1	60	658	73.9	123.2%
Scenario 2: 2033, P2G2	66	655	75.8	114.8%

6.2.2.1.4.1 Scenario 1: 2023 results and conclusions

Under current 2023 parameter values, the optimal profitability solution found by the simulations of the three configurations for Scenario 1 was in fact the same for all three different configurations. This means that under current P2G component prices, energy generation prices, wind farm capital expenses, and natural gas and hydrogen prices, P2G facilities development seems still too expensive. However, this can mostly be attributed to specific costs. These are the PEM electrolyzer stack capital expenses and its comparatively short lifetime, relatively low electrolyser and methanation energy conversion efficiencies, and the currently comparably low price of imported natural gas. The capital costs of wind power generation development, capital costs of the methanation facility, the average load factor of wind farms, and the capital costs of hydrogen storage development also seem to play a role. Further sensitivity analyses could shed light on the dependence of the profitability on these factors and other uncertainties.

As expected, we observe that wind power generation is mostly profitable and more prevalent in the energy mix for windy days, which are for the same reason those with the lowest CO₂ emissions as a result. Natural gas production/imports are highest during winter months, when demand is at its highest, and so is power generation. However, since the wind farm load factor timeseries distribution is more evenly spread throughout the year, we observe wind produced



energy more prevalent in the energy mix during milder months, and during the summer, where it can reach virtually 100% of aggregate electricity demand. Then, fossil fuel produced electricity plays an auxiliary role during peak demand usually in the winter. This can be attributed to the much lower electricity production costs of wind farms, compared to fossil fuel powered conventional production.

Total capital costs for Scenario 1 were 52 million €, which are by 100% spent on wind farm installation. O&M expenses were 295 million €, mostly for natural gas imports and wind power generation costs. Administrator horizon profit was 33.8 million €. Estimated CO₂ emissions were 1.993 million tons. The timeseries of the power production, energy mix, natural gas imports/production and CO₂ emissions can be seen in from Figure 110 to Figure 113.

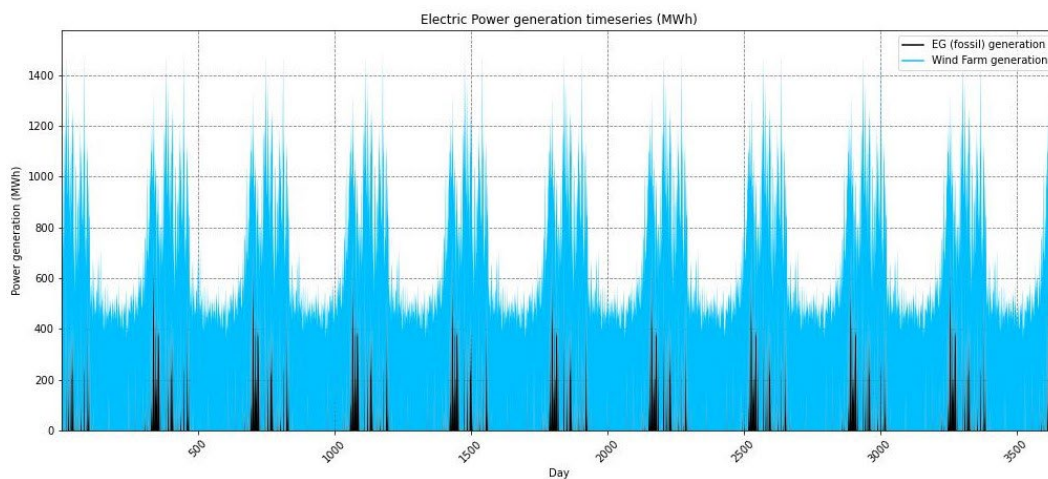


Figure 110: Electric power generation timeseries for Scenario



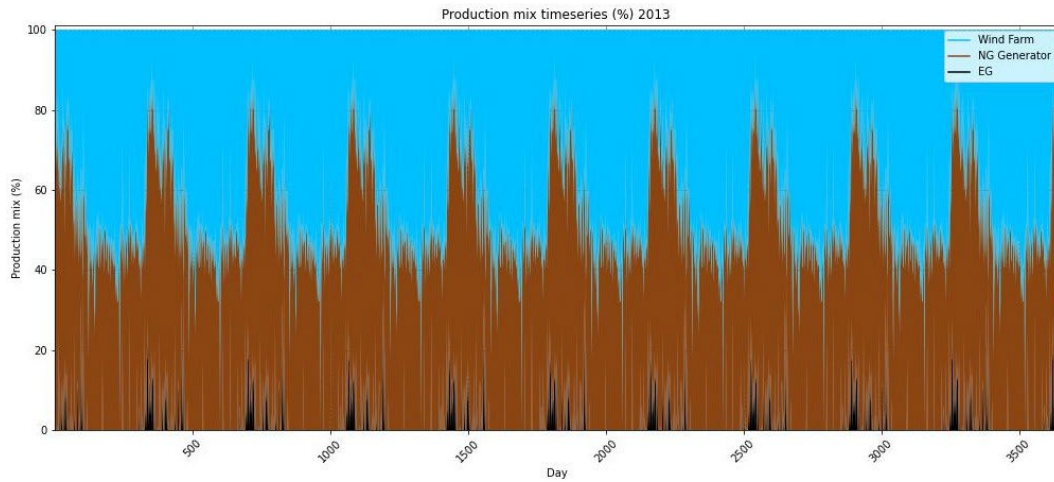


Figure 111: Energy mix timeseries for Scenario 1

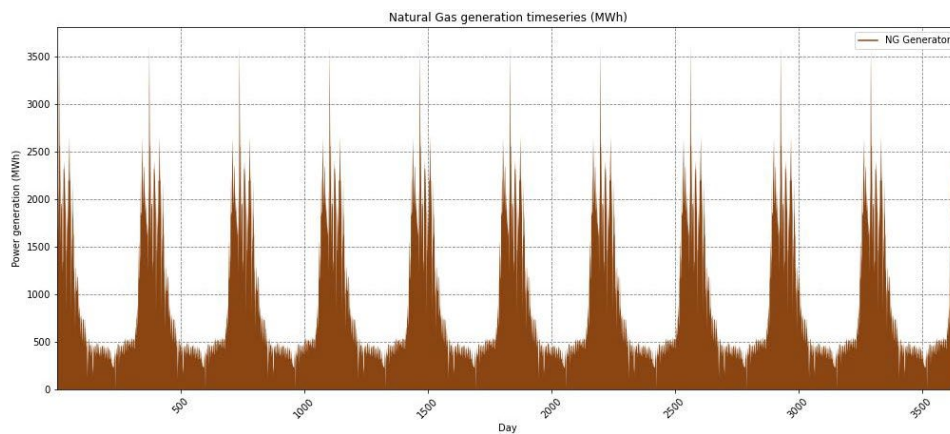


Figure 112: Natural gas imports or production for Scenario 1

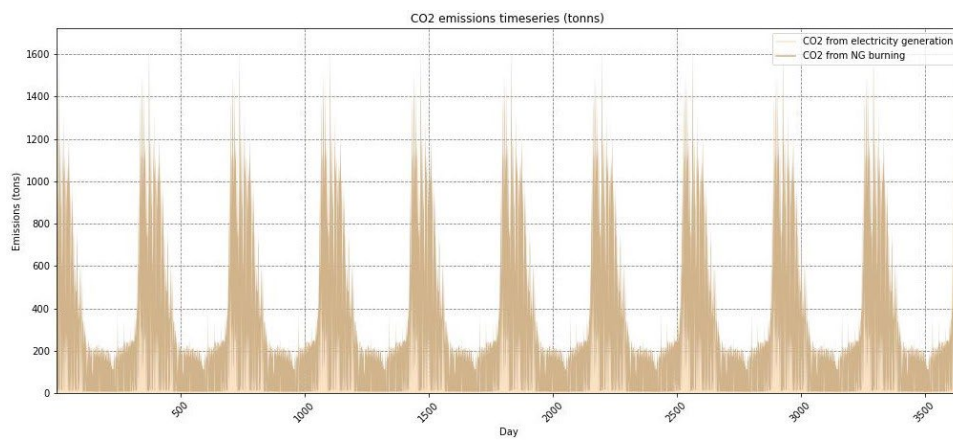


Figure 113: CO2 emissions timeseries for Scenario 1



ENERSHARE has received funding from [European Union's Horizon Europe Research and Innovation programme](#) under the Grant Agreement No 101069831

6.2.2.1.4.2 Scenario 2: 2033 projection results and conclusions

After price changes predicted by current scientific literature such as the literature cited in the previous simulation parameters chapter have taken place under Scenario 2 however, energy production shifts even more to renewable wind generation. In this case, the optimal power flow results were the same for the Baseline and P2G1 configurations, namely synthetic natural gas is generally still not profitable, for the same reasons as before, and also because now natural gas prices, and therefore methane burning as well have increased, as a result of the assumed inclusion of natural gas and methane burning into the CO₂ emissions costs policy scheme, which renders both imported or produced natural gas and synthetic natural gas or synthetic methane more expensive. In particular, synthetic methane requires the use of both electrolysis and methanation, whose lower than 1 efficiencies render this production method more expensive in general, than imported natural gas, even at the 2023 CO₂ emissions costs level of 230€ per ton of CO₂ equivalent. However, if imported natural gas prices rise even more, it is possible that this production method becomes profitable.

The results for “No P2G” and P2G1 configurations were the same, meaning that synthetic natural gas production is still not profitable under 2033 costs and other specifications forecasts. Concerning both Baseline and P2G1 configurations, the total capital costs were 60 million € (100% for wind farm development). O&M expenses were 658 million €, mostly for natural gas imports and wind power generation costs. Administrator horizon profit was 73.9 million €. Estimated CO₂ emissions were 1.861 million tons.

The most profitable configuration was that of green hydrogen production in P2G2 configuration. Hydrogen production seems to become more profitable during the windiest of days of the year, where all electricity demand is satisfied, and there is still an excess of the wind farm production to be stored to the electrolyser for later usage. This is highly expected since the profit margin of satisfying the electricity demand with wind power is on average higher than the profit margin to satisfy hydrogen demand, under the parameters and prices assumed. Therefore, wind generation is always preferred with higher priority as an investment, compared to P2G investment development.

For P2G2, the total capital costs were 66 million €, from which 2.84 million € for P2G components capital expenditure, and the rest for wind farm installation. O&M expenses were 655 million €, again mostly natural gas imports and wind power generation costs. Administrator horizon profit was 75.8 million €. Estimated CO₂ emissions were 1.807 million tons.

Another important factor to consider, is that under the simulation’s synthetic methane and hydrogen production methods, synthetic methane and hydrogen marginal production costs per unit can be calculated as follows:



$$P_{SNG} = \frac{P_{wind\ power}}{eff.electrolysis \cdot eff.methanation} + P_{methanation\ marginal\ cost}$$

$$P_{H2} = \frac{P_{wind\ power}}{eff.electrolysis}$$

where $P_{wind\ power}$ the marginal production cost of wind power, $eff.electrolysis$ and $eff.methanation$ the efficiencies of electrolysis and methanation processes, and $P_{methanation\ marginal\ cost}$ the marginal production cost of SNG in the methanation process.

Marginal selling prices of both synthetic methane and hydrogen must be higher than the above marginal production costs, in order for the P2G system to be economically feasible, and to cover the capital expenses of P2G components.

With respect to power production, energy mix, green hydrogen production, injection to the national gas grid and hydrogen storage level, as well as CO₂ emissions time series simulations results, they are illustrated in from Figure 114 to Figure 120.

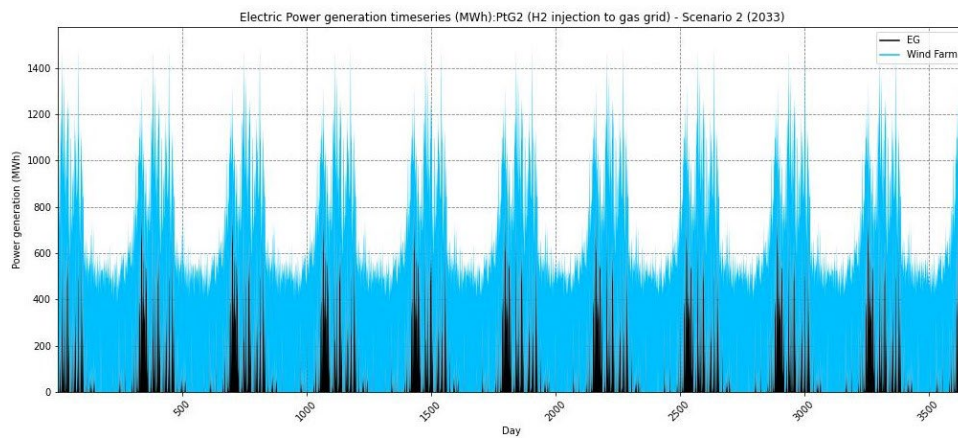


Figure 114: Electricity generation Scenario 2: 2033



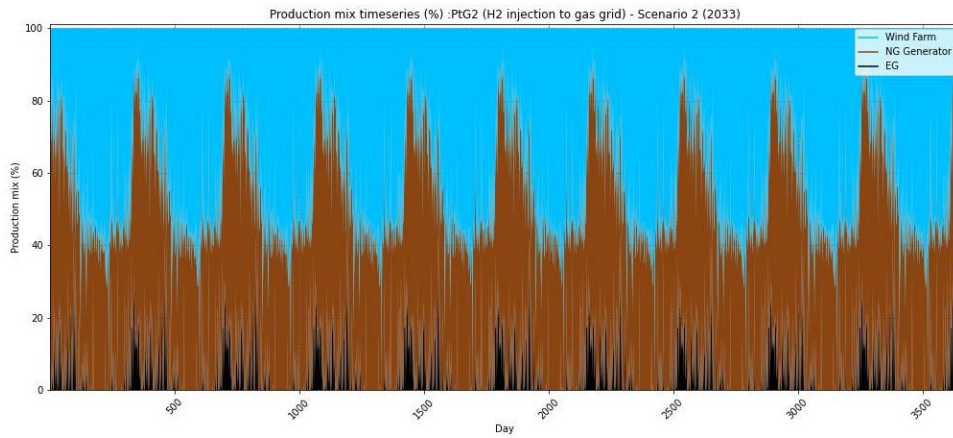


Figure 115. Energy mix timeseries for Scenario 2

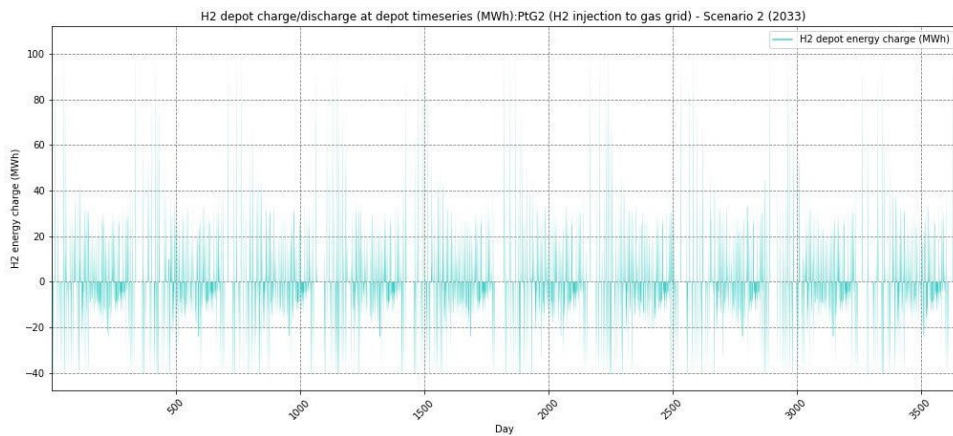


Figure 116. Green hydrogen storage charge and discharge timeseries for Scenario 2

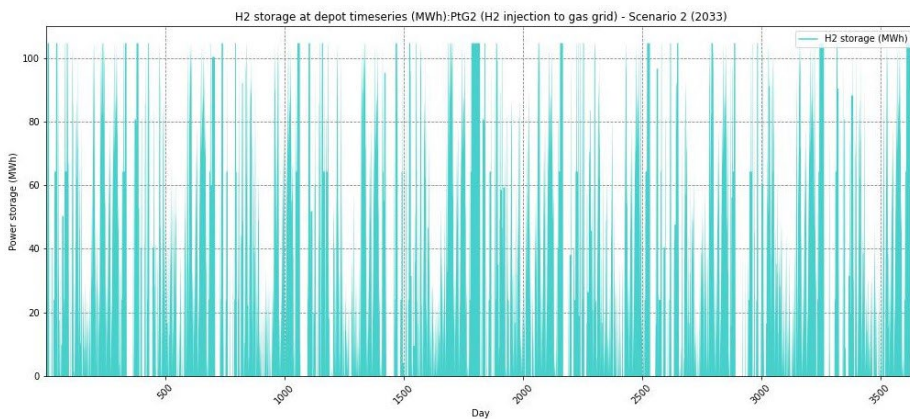


Figure 117. Green hydrogen storage level timeseries for Scenario 2



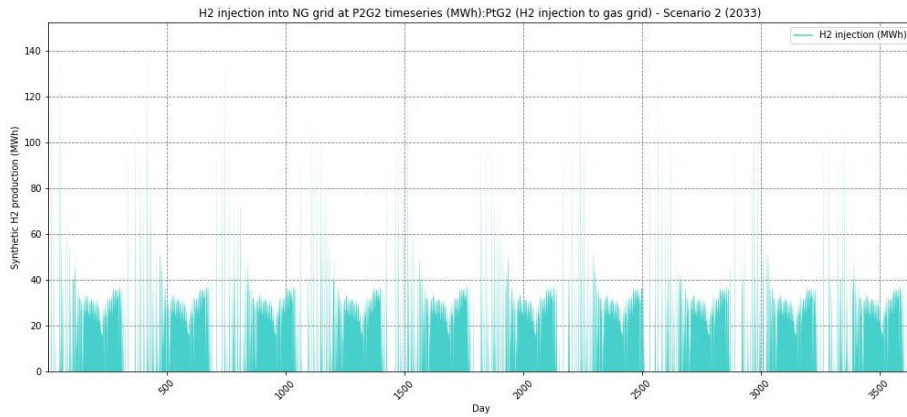


Figure 118. Green hydrogen injection into natural gas grid timeseries for Scenario 2

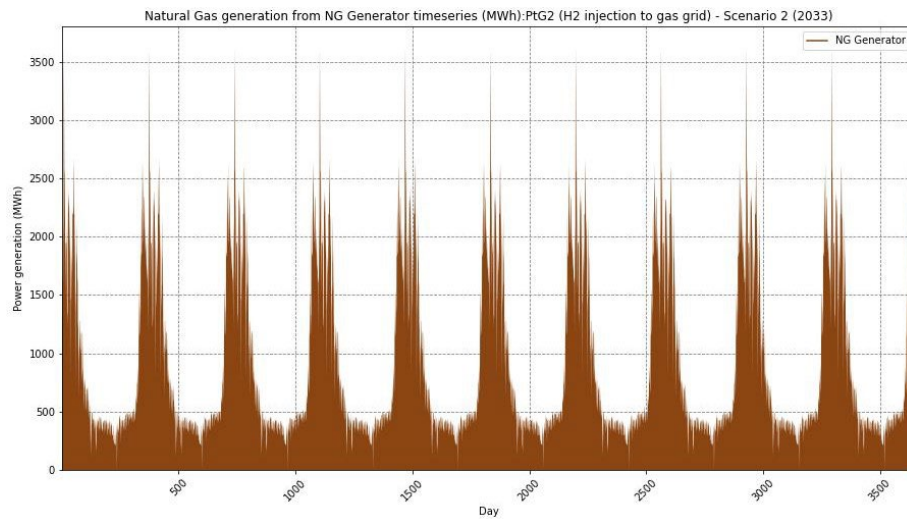


Figure 119. Natural gas imports or generation timeseries for Scenario 2

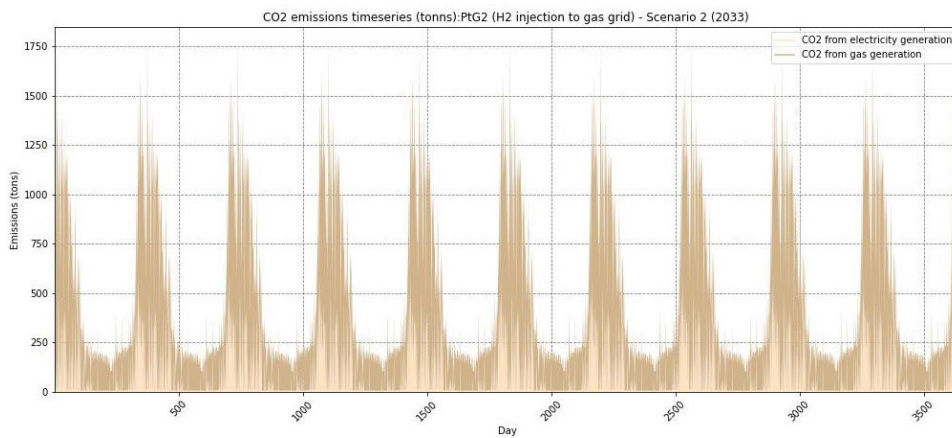


Figure 120. CO2 emissions timeseries for Scenario 2

6.2.3 Interface specifications

Please refer to section 6.2.1.1.4.

6.2.4 Front-end

Streamlit⁴² is the technology used to develop the TwinP2G front-end, where users can interact with the TwinP2G models, by specifying and customizing parameters and other inputs they wish, and also receive the results of the model as visualisations, or if needed as extracted files.

6.2.4.1 User roles

The TwinP2G front-end has been designed to achieve usability by two main user roles: “Data Scientist” and an “Energy Engineer”. The “Data Scientist” is a role who is an experienced user with a scientific background and coding skills, as well as modelling capabilities in P2G use cases. This user can access TwinP2G's "Data Science Platform" to interact with the "Simulation and Optimization" component. They can develop P2G experiments and visualize the results, including simulation and optimization outcomes, forecasting accuracies, and more. Initially, Jupyter notebooks formed the core of the data science platform, however the user experience has been extended through the usage of Streamlit.

The second role is the "Energy Engineer", who is an end-user with knowledge and understanding of energy systems but possesses limited coding and modelling skills. Such users will only be interested in observing the behaviour and the current status, and at most having some high-level information to gain insights for decision support concerning future P2G investments. This persona needs to monitor simulation and forecast results, as well as relevant metrics. The latter component also relies on Streamlit, alongside a strong interconnection of TWINP2G with the Enershare visualisation engine developed within T6.4.

6.2.4.2 Demonstration

Typical representations of the TwinP2G front-end can be seen below. The user can input their customized power network components and parameters and data time series directly when prompted by the front-end, but they can also store customized “use-cases” for future automated use. Network components, such as buses, generators, power lines, loads and P2G components can be specified, customized, and added, creating customized energy networks

⁴² <https://streamlit.io/>



containing conventional and renewable power generators, and power demands timeseries (electric, natural gas and hydrogen), as demonstrated from Figure 121 to Figure 125.

Twin P2G

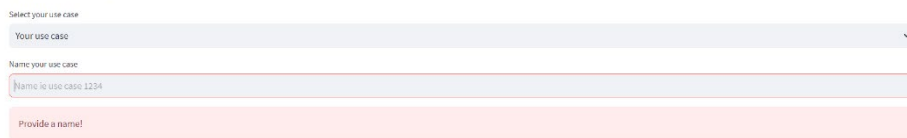


Figure 121: Definition of customizable use cases via Streamlit front end.

Network

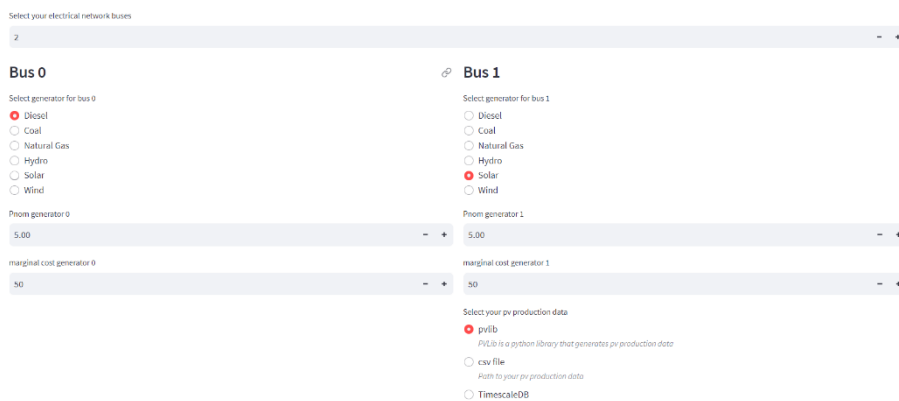


Figure 122. Adding buses, connect generators, and import generation timeseries.

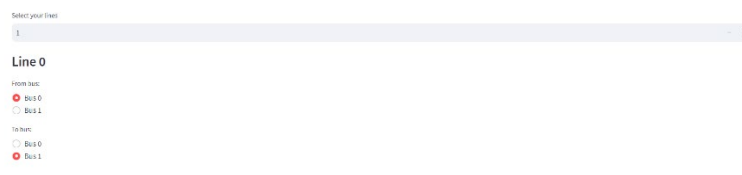


Figure 123. Addition of power lines

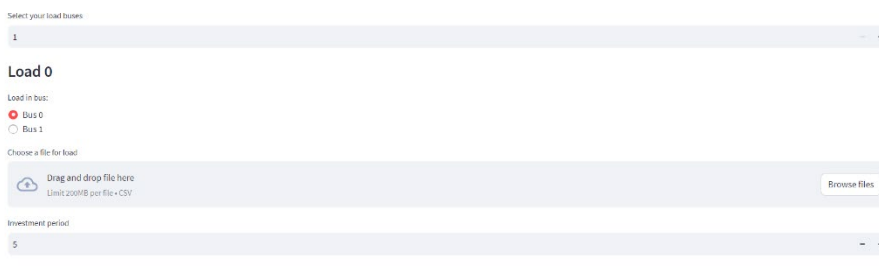


Figure 124. Connection of power loads to buses and insertion loads timeseries



Electrolyzer

Electrolyzer link from bus:
 Bus 0
 Bus 1

Electrolyzer link to bus:
 Hydrogen Bus

Nominal electrolyzer
 - +

electrolyzer cost per MW
 - +

marginal cost el
 - +

Fuel Cell

Fuel Cell link from bus:
 Hydrogen Bus

Fuel Cell link to bus:
 Bus 0
 Bus 1

Nominal fuel cell
 - +

fuel cell cost
 - +

marginal cost fc
 - +

Hydrogen Storage

Hydrogen Storage bus:
 Hydrogen Bus

Nominal H2 buffer
 - +

H2 buffer cost
 - +

marginal cost H2
 - +

Figure 125. Insertion of P2G components and customization of their features

After the user has received the simulation results based on their network inputs and customizations, the UI also provides the option to select customized timeslot visualisations of the simulation, as well as the timeseries specified, such as any generator's energy production timeseries, loads timeseries, any P2G components such as electrolyzers, methanation units output timeseries, or hydrogen storage level and injections to the gas grid timeseries. An example is shown in Figure 126.

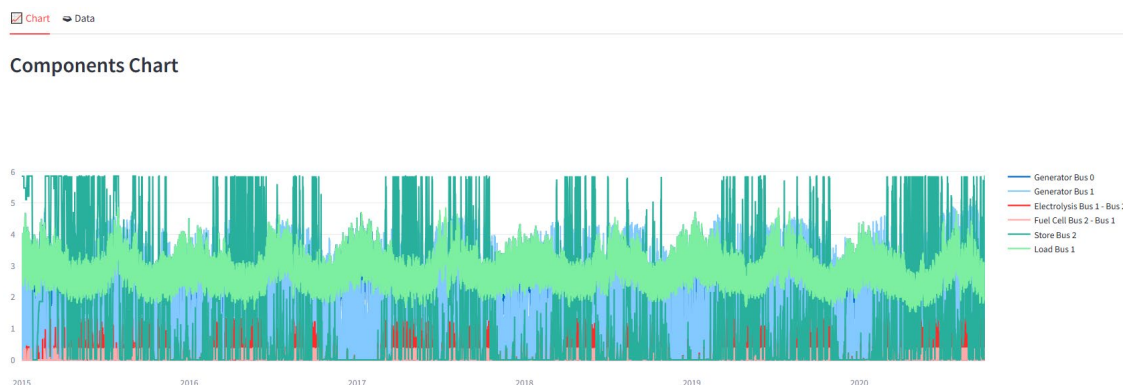


Figure 126. Simulation results timeseries selection

6.2.5 Requirements and deployment strategy for pilot feedback

NTUA already possesses at its premises a virtual machine dedicated to Enershare project (currently 8 cores, 16GB ram and further extendable in the future) that also provides a public IP that can be accessible through VPN.

Given that the service is already at a quite mature development stage, the next steps from NTUA developers will be to create a unified docker deployment of all its subcomponents therefore



allowing for replicability, scalability, and public deployment at NTUA’s premises meant for extensive pilot testing, evaluation and feedback. This process will lead to the continuous evaluation of TwinP2G leading in turn to continuous refinement and improvements of the software until its latest development stages.

6.2.6 Integration with ENERSHARE Data Space and Services

The use case is expected to contribute to the Data Space by providing access to the aforementioned datasets and/or a summary of the inputs (simulation parametrisation) / outputs (simulation results) of the DT. Figure 127 describes the integration plan with the Data Space for the current use case. The diagram shows the envisioned structure for the integration of various services into the Enershare Data Space, highlighting the strategic use of the IDS Connector for data interoperability and security. The central component of this framework is the IDS Connector, a key element that facilitates the integration of different services, including TwinP2G and most importantly its main database which allows for storage of datasets and simulation results, while offering an OpenAPI for their provision to the Data Space. These services will be designed to connect seamlessly with data sources and sinks within the Data Space, using the TSG IDS Connector to exchange data and metadata. Some of the components of the ENERSHARE IDS framework could be integrated at a later stage depending on demand and availability. Also note here that the integration plan might be reduced to one of the two connections presented in the schema.

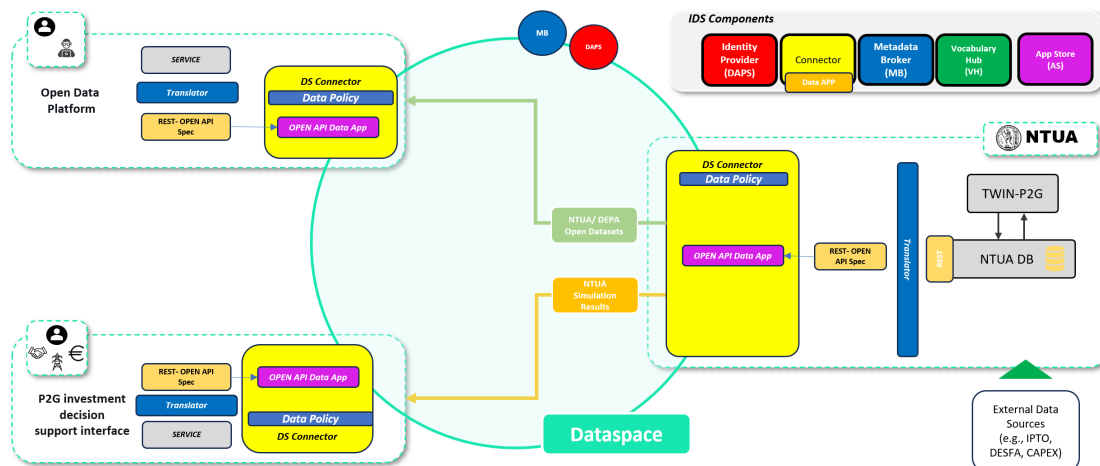


Figure 127: Data Space integration plan for TwinP2G and pilot 4

The International Data Spaces specifies the interactions taking place between the different components in an IDS ecosystem:



5. **Onboarding**, i.e., what to do to be granted access to the International Data Spaces as a Data Provider or Data Consumer. Pilot 4 will use the Identity provider of the Data Space to obtain the certificates as Participant and for the connectors.
6. **Data Offering**, i.e., offering data or searching for a suitable dataset. Through the Open Api the different services will be exposed. The pilot 4 connector provider will register in the Metadata Broker of the Data Space and in this way their data, access conditions and how to call it will be part of the Enershare catalogue and the rest of the participants could search and use this information.
7. **Contract Negotiation**, i.e., accept data offers by negotiating the usage policies. Pilot 4 will define and check data usage control.
8. **Exchanging Data**, i.e., transfer data between IDS Participants. Pilot 4 will use an IDS connector to exchange the data.

The process will be as follows:

- A participant of the Data Space will query the Metadata broker to see the available services/datasets and their restrictions.
- A participant of the Data Space wants to use one service of pilot 4.
- Through their connector consumer, they will call to pilot 4 connector provider.
- Then a contract negotiation will be held and if a contract agreement is reached, the data will be exchanged. All the interchange messages must be logged in the Clearing House.
- Sometimes a service will not be able to obtain the expected result immediately because some algorithms must be called and the processing time of them could be huge. In this case, the connector of Pilot 4 will send a notification message to the connector of the Participant when the data are available.

6.2.6.1.1 MVP deployment

For the next version of the MVP, a REST service is being developed. This first version of the REST API contains a call that lets the user obtain the results of a simulation given a set of input parameters. The output of the REST service will be a JSON object with the output parameters. The REST endpoint will be exposed to the Data Space through a TSG connector.

The next steps for the subsequent releases will include the following tasks:

- Evolve the REST API to include all the required methods and services.
- Register the Connector in the Metadata Broker.
- Define the Contract conditions of the different services and data.



6.2.6.1.2 Internal Deployment

TwinP2G development

TwinP2G is being developed as a Digital Twin to allow simulations for power2gas integration in the conventional power grid. The development details can be found in D6.2. These open datasets will probably be provided by the Data Space.

Database development

A Postgres database has been set for storing the datasets to be used by TwinP2G alongside the simulation results to enable their exploitation by P2G stakeholders and investors. These results will be shared within the Data Space.

Development of the Open API

The REST service will be developed according to the Open API and data model defined in WP3. The developed REST API will be exposed to the Data Space through an IDS connector to ensure interoperability and secure access to the data.

6.3 Digital Twin based O&M algorithms and generation of synthetic failures data

6.3.1 Changes since deliverable D6.1

Following to that exposed in deliverable D6.1, this DT will be used for synthetic data generation, both for electric generator as for the hydraulic pitch system of a wind turbine. This synthetic data is useful for feeding the services described in section 3.4, since sometimes the availability of labelled failure data is hard to achieve.

As first step, a Simulink model, depicted in Figure 128, has been created (both for PMSG and hydraulic pitch system). Electric failures such as short-circuits in the stator winding of the PMSG (inter-coil, phase-phase, ground-phase) can be represented, as well as different working conditions in the hydraulic pitch system (pump condition ON/OFF, cylinder extension/retraction, ...).



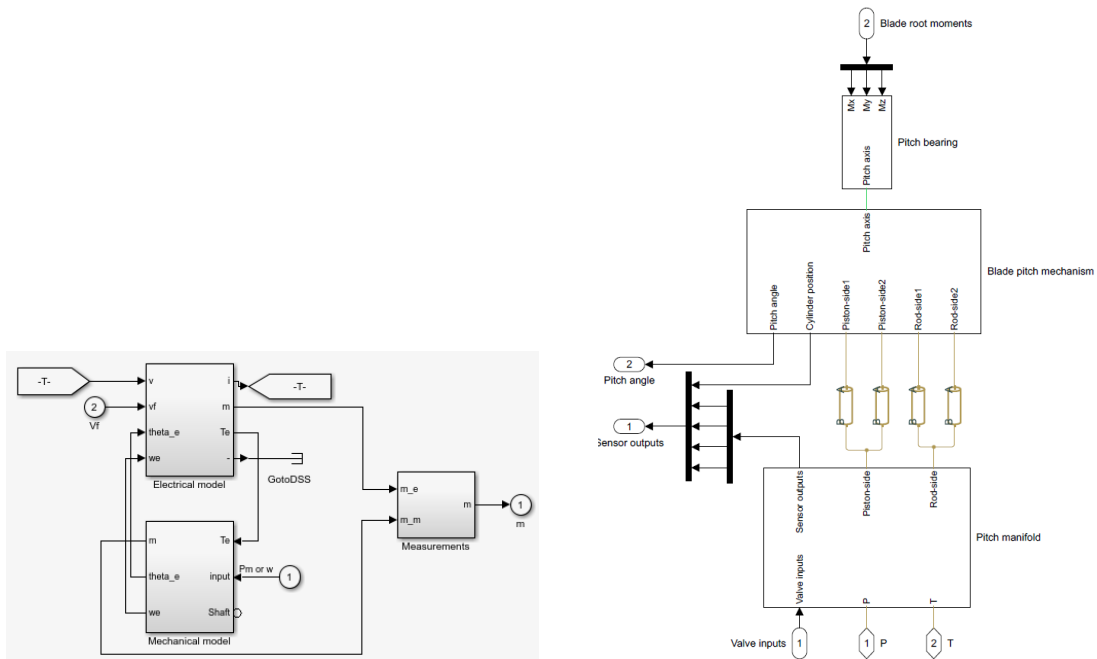


Figure 128: Simulink model for PMSG and hydraulic pitch system

As second step, a script has been created and deployed using Simulink Compiler. Finally, a Microservice docker image is created by MATLAB® Compiler SDK™, providing an http/https endpoint to access MATLAB code, for running the Simulink model previously developed. In this way, a MATLAB function is compiled into a deployable archive, and then creating a Docker image that contains the archive and a minimal MATLAB Runtime package. The image can be run in Docker and make calls to the service using any of the MATLAB Production Server™ client APIs.

There has not been progress in TRL terms, only software progress and improvement. A final TRL of 7 is expected at the end of Task 6.5 (in month M28).

6.3.2 Interface specifications

REST API is being implemented to give access to the above algorithms. The endpoints of the REST Api will be the following:

- **Root endpoint:** <https://digitaltwin.enershare.urban.tecnalia.dev/api-digitalTwin/v1/ui/>
- **Generator Digital twin:** /generator/digital_twin
 - Description: Retrieve synthetic data of the generator of a wind turbine, given some specific wind turbine input data.
 - HTTP Method: POST



- Input data: The input will be a mixture of generator, wind turbine and weather data.
- Output data: The output will be the synthetic data.
-
- **Hydraulic pitch system Digital twin:** /hydraulic/digital_twin
 - Description: Retrieve synthetic data of the hydraulic pitch system of a wind turbine.
 - HTTP Method: POST
 - Input data: The input will be the wind speed
 - Output data: The output will be the synthetic data of the hydraulic pitch system.

All the input and output json can be transformed into a json-ld align with Enershare ontology using the transformation service developed in WP3. This service is explained in D3.2.

6.3.3 Requirements and deployment strategy for pilot feedback

TECNALIA already possesses at its premises a virtual machine dedicated to ENERSHARE project (16GB RAM and further extendable in the future) that also provides a public IP (150.241.10.28) that can be accessible through VPN.

TECNALIA developers will create a unified docker deployment of all its subcomponents/services for extensive pilot testing, evaluation, and feedback.

6.3.4 Integration with ENERSHARE Data Space and Services

The integration of the wind turbine digital twin for synthetic data generation in the ENERSHARE dataspace, will be done using IDS connectors – as presented in Figure 129. We will use the TNO TSG connector for secure data exchange and to facilitate the interoperability between our services and the data sources within the Data space. Each digital twin will have their own REST API to access their functionalities. The Data App will encapsulate the REST calls and will be responsible for preparing and parsing the input data and redirecting the entering calls in a suitable way. Also, to wrap the output synthetic data and provide it in JSON format according to the common ENERSHARE Data Models.



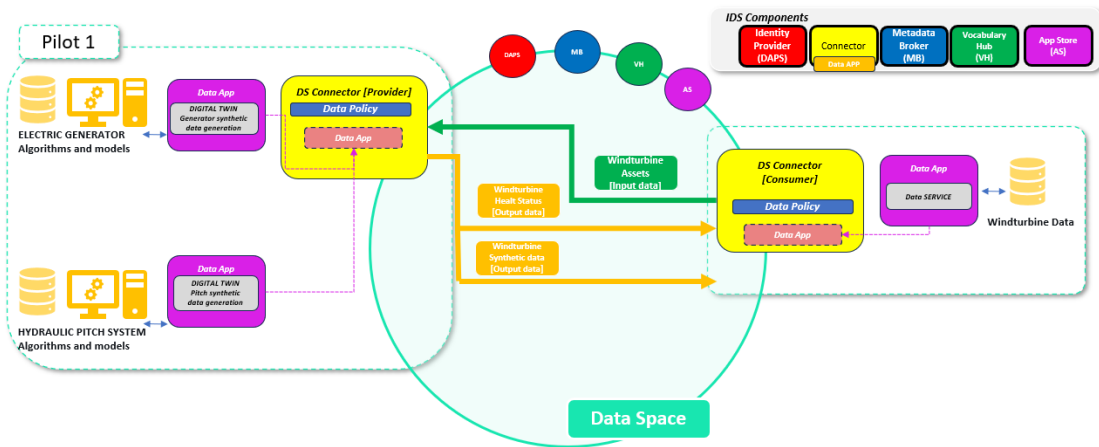


Figure 129: High level overview for the DT for wind energy O&M integration with the Data Space

6.4 Digital Twin for flexible energy networks

The DT concept for electrical networks is based on the simulation tool [DPSim](#) and the acquisition of measurement points from a real electrical network.

However, we are aiming now for a paradigm change, more microservice oriented and Data Space friendly. This strategic decision leads us to a solution where a dedicated data-app plays together with DPSim a central role. In the following sections of the current deliverable, we proceed to document the new concept and the implementation efforts of the application that is built iteratively on the high-level architecture presented in D6.1 (as shown in Figure 130). The current paradigm evolved into a more concrete software architecture for the implementation of the service, with a data-app that relies in the adaptation and adoption of the [VILLAS](#) framework components. This is explained in more detail in the subsequent sections.



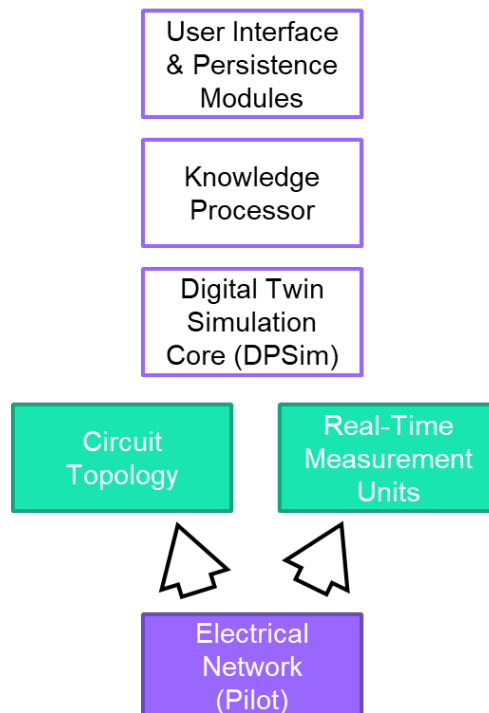


Figure 130: High-level architecture of the DT for flexibility in electrical networks

6.4.1 Changes since deliverable D6.1

After analysing the functions required previously in D6.1, we decided to generalize the options available for the platform user. This gives us a concept for a dedicated data app within the concept of the proposed dataspace in WP4, in this case for DT applications, as our requirements become special due to the nature of the information and the volume and time-criticality of the communications.

Concept

This data app concept serves to the purposes of a digital twin application that requires to:

- exchange data in a real-time context or in a continuous manner
- control the digital twin configuration, parameters or properties
- modify the running state of the digital twin

For this type of applications, the mere use of solutions based on RESTful API over HTTP is not the best choice in all the cases, by the following reasons:





- the exchange of some of the data needs to be as fast and deterministic as possible (measurements, network reconfigurations, financial-related data, etc.)
- the control of the configuration and properties would benefit of a unified and standardized interface, which is extendable to another digital twin or time-critical applications

Solution

The proposed solution is a system architecture that is described in the respective integration with Data Space section.

In summary, there is a data stream using VILLASframework, that handles the real-time communication between the actors and services, and at the same time complies with the Data App specifications from the TSG implementation and the IDS-RAM specifications.

Road Ahead

As the concept is being redesigned, the changes in the TRL of the solution as a whole are brought mainly by the introduction of new apps within the dataspace concept. The old Simulation Core was repropsoed into a Digital Twin Worker App, to be controlled using the dataspace. This redesign will enable a first-of-a-kind full dataspace integration for Digital Twin applications. More information about the current and future developments can be observed in Table 18.

Table 18: Digital Twin for flexible energy networks TRL status

Aspect	Current TRL and justification	Next steps
Data App: Consumer Data App: Producer	Current standing at TRL 4 as a new development (not present in the Alpha Version), the tests with the tool VILLASnode are ongoing. The system is being tested in the internal servers of RWTH. Proof of concept using synthetic data outside the dataspace is working and was already proven in similar setups in other applications.	TRL5/6 will be reached with the full integration into Data Apps within the ENERSHARE DATASPACE, as a highly proven technology demonstrator running in varied scenarios.
Data App: Digital Twin Worker	TRL 4, with no changes in the readiness level due to significant changes, in terms of the redesign of the concept. Proof of concept using network data outside the dataspace is working as reported in the Alpha Version. Test of the OpenAPI coming from the project I-ENERGY has started, to support the Digital Twin Manager.	TRL 6-7 is expected. The full connection to the Dataspace will enable a use in operation conditions, showing readiness for real-life conditions.



Aspect	Current TRL and justification	Next steps
Data App: Digital Twin Manager Data App: Management and Control GUI	Currently at TRL 2 as a new development (not present in the Alpha Version). The supporting tools VILLAScontroller and VILLASweb and their concept were proven in a different setting and are currently redesigned for the purpose of the Data Apps in the ENERSHARE Data Space.	TRL4-5 expected, as a technology demonstrator for controlling digital twin worker applications.

6.4.2 Experimental setup and challenges

A next-generation concept for digital twins would require then, as previously mentioned, three different data flows. Taking as a base concept the ideas on the Federated Learning Data App that is part of the TNO Secure Gateway, we can end in two implementations that will cooperatively act to provide a full set of functionalities for digital twin applications, one that deals with the data solutions and another that provides the management of the application that delivers the digital twin services:

a) Digital Twin Data Provisioning Data Apps

The digital twin uses data in a different way, and the requirement comes from the user perspectives that varies from the regular needs. The data needs to be ingested in a way that ensures availability, particularly for time critical processing. The exact values for the times are given by the service provided, and it can go from a few milliseconds to some seconds in the most challenging cases.

There are two main user roles that can be envisioned in this case. One is the Digital Twin Engineer, and the other is the Data Owner.

The first one, **Digital Twin Engineer**, represents the technical user that sets the configuration of the data streams and the routing of the data from the perspective of the service, and we can describe the user stories as follows:

As a Digital Twin Engineer, I want to receive data input into digital twin applications.

As a Digital Twin Engineer, I want to configure a data input stream channel for digital twin applications.

As a Digital Twin Engineer, I want to configure the data that sets the parameters of the digital twin service.



As a Digital Twin Engineer, I want to pre- and post-process the data stream before and after is used by digital twin applications.

On the other hand, the **Data Owner** is a way to encompass the role of the pilot partner. They might want to share the data but restrict the reach or the purposes for which is used and evaluate the results of the digital twin service. This is a potential connection point with the work of the ENERSHARE Data Visualisation service, which is also part of WP6. The main user stories are then:

As a Data Owner, I want to provide data outputs to digital twin applications.

As a Data Owner, I want to configure a data output stream channel for digital twin applications.

As a Data Owner, I want to decide who can use the data output stream channel.

As a Data Owner, I want to analyse the data results of the service.

b) Digital Twin Management Data Apps

The digital twin service provisioning is a way to expose an interface that grants privileged users the required tools to control the execution of the apps that transform and process the input data.

This configuration needs to be done using a specialized toolchain that allows to trigger events and control the flow of the simulation or data-driven application that provides the digital twin core services. This set of roles derived can be represented as the following user stories for the **Digital Twin Engineer**

As a Digital Twin Engineer, I want to select and load the configuration of the digital twin simulation.

As a Digital Twin Engineer, I want to select the running status of the digital twin simulation.

As a Digital Twin Engineer, I want to evaluate the status of the digital twin.

We decided to adopt the VILLAS framework as the solution, as by adapting it we can solve the architectural design problems that a Digital Twin application faces and integrate the services into the Data Space at the same time.

On the first side, there are time constraint integrations for applications that can be described as follows:



The **Hard Real-time Constraint Applications** encompass an integration that can work around the limitations and requirements of diverse digital twins (simulator and data-driven), hardware devices and data sources that demand a point-to-point connection to ensure real-time interactions.

The **Soft Time Constraint Applications** are about the tools that do not run in time-critical contexts. This includes the user interactions (for example via the web app) or the configuration, management, and control tools.

The adoption and adaptation of **VILLAS framework** will not only give support to our Digital Twin but will also be an enabler for System-of-system Digital Twin concepts and for any time-critical and data-intensive process that needs to deal with dataspace connector integrations.

The principal components to be adapted will be mentioned hereinafter, with the disclaimer that during the implementation phase there are possible changes. The number of services and the scope of the solution for each individual use case will be adapted to fit the solution and the ENERSHARE Dataspace MVP. Those changes will be documented and released in the D6.3, as part of the final technological release.

VILLASnode consist of a server daemon and client modules. It acts as a gateway that forwards simulation data between clients, using state-of-the-art encryption and tunnelling. Each client (an equipment or service) is implemented using a node-type as part of VILLASnode allowing it to communicate and send data while collecting statistics and ensuring quality of service. It provides both a web interface for live monitoring and a remote API to control it via HTTP. Additionally, the hook-types allow to process the data in a data stream (shift, cast, calculate averages, phasor, etc.) enabling the metering functions required for an electrical digital twin in a configurable way.

VILLAScontroller is a unified controller API for simulation hardware and software, with a multi-vendor focus. It uses AMQP communication to send commands to control the simulation. The current approach is CLI-based but is going to be rewritten into a standardised OpenAPI to be easily accessible via the Dataspace. There will likely be a client-side and a server-side implementation.

VILLASweb is a tool to configure real-time co-simulations and display simulation real-time data. It consists of a web-based application that connects to a backend and to VILLASnode instances. The simulation data is channelled to the web application using the VILLASnode websockets communications.



The VILLASframework will in turn be interfaced with the Digital Twin Simulation Core (based on DPsim), making possible to have the whole chain of services available for more complex applications. Examples of those services will be discussed with the pilot partner and added to the visualisation layer and widgets available in VILLASweb.

6.4.3 Interface specifications

The data app implementations to provide digital twin services will need to comply with the specifications defined to deploy in a core container (<https://tno-tsg.gitlab.io/docs/data-apps/build/>). The main challenges are related to the implementation decisions, and the fact that the main language for the TSG is Kotlin.

The proposed solutions for the data interfacing can be summarized by:

A **Data Provider** solution, to stream the data coming from the measurement devices and other network assets from the pilot partner. This is going to be achieved using a VILLASnode adapted implementation as a DataApp.

A **Data Consumer** solution, to receive the data stream coming from the pilot partner. Is the counterpart of the Data Provider. Similarly, the use of a VILLASnode implementation as a DataApp will provide the required capabilities.

A **Data Processing** solution, to provide additional services for data in digital twins. A good example for this is to change the measurement unit, to combine data to obtain a calculated magnitude, to do interpolations or data cleaning or harmonisation in a pipeline within the time constraints required by the digital twin solution.

Moreover, the apps that manage the data are needing the connection to the services that manage the configurations and provide results:

A **Digital Twin Worker**, which contains the simulator or data-driven app. This ensures that the data is only used for the purposes that the Data Owner has established. In the case of the Flexible Energy Networks, the implementation with DPsim will give an exemplary case that showcases solutions for real-time challenges.

A **Digital Twin Manager**, that configures and manages the digital twin lifecycle. The Digital Twin Worker needs to be not only configured, but also monitored and if needed, steps required to return to a healthy state must be taken. The data streams and connections to the Worker and its internal variables will be done here. This will become an extension of the VILLAScontroller, a solution developed to manage simulators in the electrical networks study field.



All the previous solutions will require a wrapper, as the main language is not Kotlin. This was already achieved with other languages like Python, so the expertise on how to envision, design, develop and deploy this type of solution is present in RWTH-ACS.

The documentation of the software will be done using Swagger, to ensure a process that allows to keep it aligned with the latest developments, and to automatically deploy the current version that is compliant with the OpenAPI specifications.

6.4.4 Front-end

The frontend will be adjusted to the requirements of the users. The configuration screens will include elements to control and adjust the simulation.

An example of the potential for the results viewing screen is visible in Figure 131, in this case using the graphical interface of the VILLASframework. The widgets will reproduce details of interest for the user.

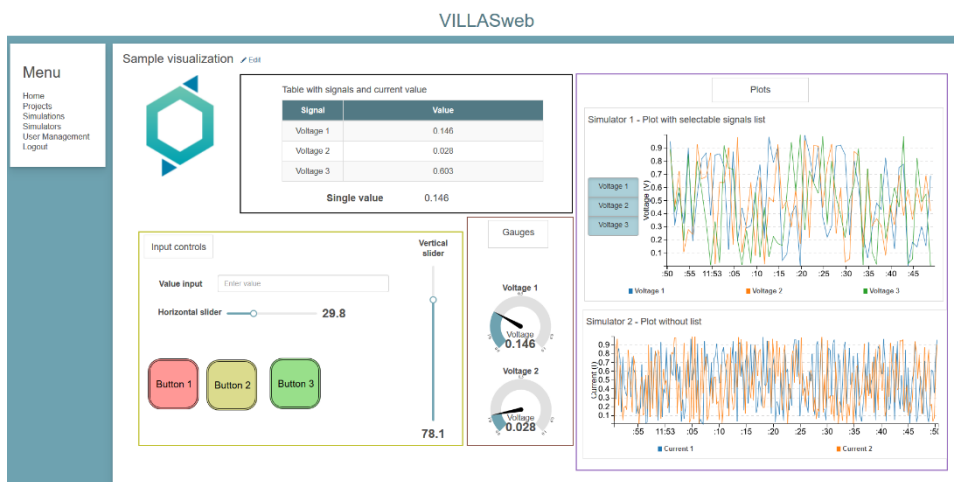


Figure 131: Example graphics using VILLASweb and VILLAScontroller

6.4.5 Requirements and deployment strategy for pilot feedback

The deployment for the service is going to be done by using a microservice approach. The configuration using Helm Charts will be the preferred option for the standardisation in the process.

However, the initial hosting and configuration of the services will be done in the premises of the RWTH-ACS servers, to ensure a controlled environment where there is access to the logs and performance metrics.



RWTH-ACS has allocated dedicated resources to the project, and in particular a set of virtual machines in the self-managed OpenStack cluster is available for the usage of the task. The machines are connected to the internet, and particularly for this task a reserved set of resources (VCPU 4 cores, RAM 8Gb) that is expandable will be available for the duration of the project as shown in Figure 132. This ensures an environment that replicates real-life conditions and challenges.

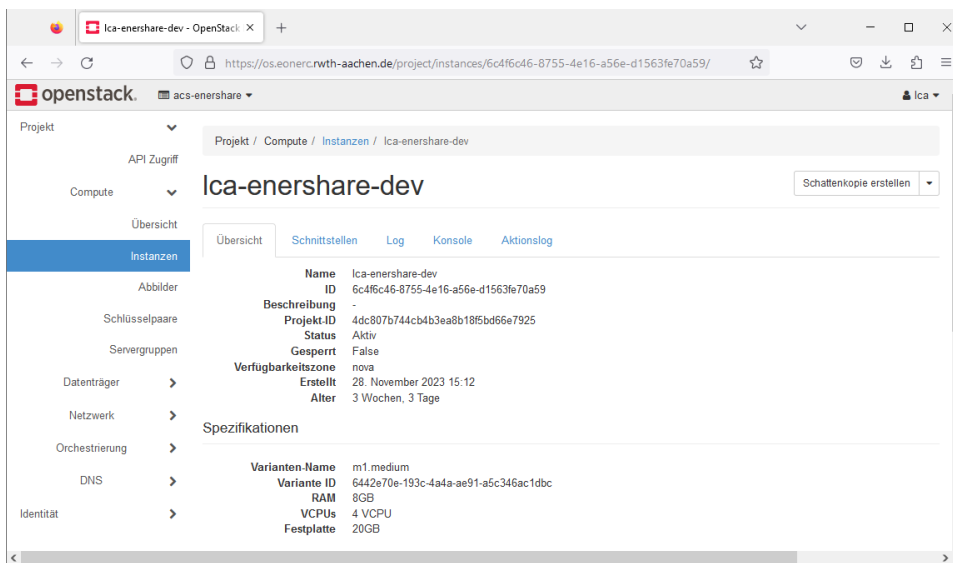


Figure 132: Screenshot of the OpenStack dashboard

6.4.6 Integration with ENERSHARE Data Space and Services

Figure 133 represents the current design for the architecture of the integration with the ENERSHARE Dataspace. The main difference compared to the standard approach of a RESTful API that is directly integrated with the Data Connector (TSG in our assumption) is that in this proposed new architecture approach, the data flows using other types of communication, for example network sockets. However, for some of the communications a RESTful approach might still be used, an example of this can be the management interface.

This data streams are going to be powered by VILLASframework, ensuring that the data flow from the measurement devices in the pilot partner premises can be securely and timely transferred to the Digital Twin Simulation Core.

There are then three main separate service instances that interact with each other:

The Data Provisioning is deployed in the pilot partner premises. Consist in a data app that streamlines the data ingestion and pre-processing. It includes the GUI that the Data Owner



needs to manage the configuration of the data streams ensuring that it complies with their decisions, establishing the policy for the usage of the streams.

The Digital Twin itself, which contains most of the backend-related services, namely the Simulation Core, the Simulation Manager and the VILLAScontroller. All of them allow to enable the configuration options and to run the simulation-based digital twin.

The Digital Twin Manager is a separate service focused on the user interaction. It provides the dashboard to set the simulation parameters and configurations and to explore and analyse the results.

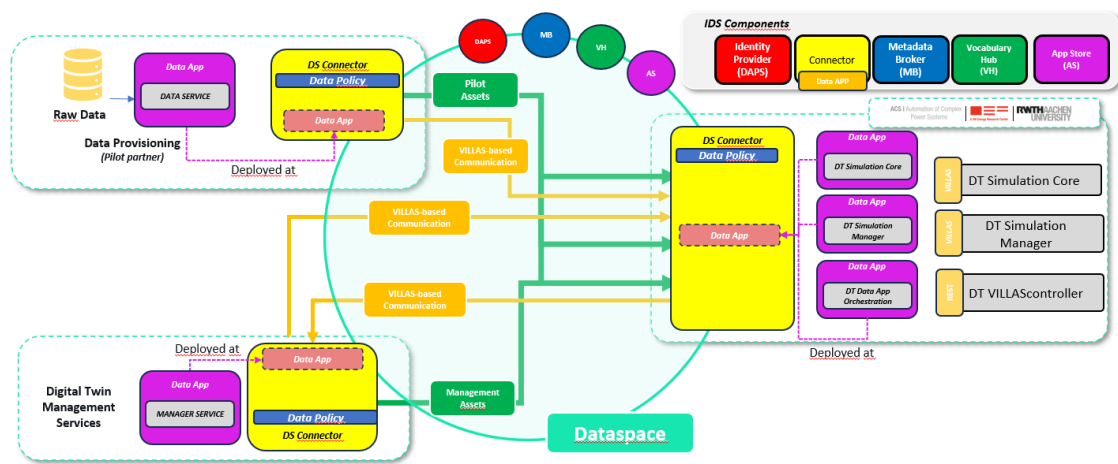


Figure 133: Data Space integration of the Digital Twin for Flexible Energy Networks

Next steps

A roadmap to integrate the minimal version of the digital twin into the project's dataspace is currently ongoing, comprising:

1. Implementation of the Data App based on VILLASnode.
2. Implementation of a Management and Control Data App based on VILLAScontroller.
3. Interfacing of the Digital Twin Simulation core based on DPsim with the Data App and the Management and Control App.
4. Testing of the solution, including the end-to-end cases.



7 Final Remarks

The ENERSHARE project pioneers the integration of the Data Space paradigm within the energy sector, fostering the development of cutting-edge data-driven services and System-of-System Digital Twin applications. This innovative ecosystem tackles various challenges, encompassing load flexibility estimation, operational dynamics of local energy communities, functions of TSOs and DSOs, as well as the optimization of RES O&M. Moreover, ENERSHARE extends its impact to cross-sector domains, introducing services such as wellness alarms, building comfort monitoring, and EV management. The overarching System-level DTs play a pivotal role in enabling sophisticated energy planning and facilitating real-time operations across diverse domains, including electrical networks, wind energy systems, and power-to-gas applications.

The following KPIs for ENERSHARE services were defined in the objectives of the Description of Action:

- ≥ 10 energy services demonstrating long-lasting data-centric business models. **In D6.2: 10 energy services were described and available in its beta version, plus the 4 federated learning implementations for energy services.**
- ≥ 6 non-energy services enabled by the Energy Data Space. **In D6.2: 7 cross-sector services were described and available in its beta version.**
- ≥ 15 digital services fully integrated with the Energy Data Space. 21 data-driven services described in this the beta version (D6.2). **For the final version (D6.3), we expect that all energy services are integrated with the Energy Data Space. Two services were already tested integrated with the Data Space (using the OneNet connector).**
- ≥ 3 Digital Twins. **In D6.2: 3 digital twins were described and available in its beta version.**

Finally, it is important to mention that all services from WP6 are in the beta version ranging between TRL 5 and 7. As these services progress towards their final versions, they are expected to achieve a TRL of 6 to 7, demonstrating increased maturity and stability. Additionally, the upcoming testing within the framework of WP9 presents an opportunity for further refinement and enhancement, ultimately contributing to the overall robustness and effectiveness of these services.



8 References

Alvarez, F. M., Troncoso, A., Riquelme, J. C., Ruiz, J. S. A. (2010). Energy time series forecasting based on pattern sequence similarity. *IEEE Transactions on Knowledge and Data Engineering*, 23(8), 1230-1243.

Christensen A. (2020). Assessment of Hydrogen Production Costs from Electrolysis: United States and Europe, ICCT.

https://theicct.org/wp-content/uploads/2021/06/final_icct2020_assessment_of_hydrogen_production_costs-v2.pdf

Dufendach, L., et al. (2017). A Randomized Trial Comparing Classical Participatory Design to VandAID, an Interactive CrowdSourcing Platform to Facilitate User-centered Design. *Methods of Information in Medicine*. <https://doi.org/10.3414/me16-01-0098>

Erdener, B.C. et al. (2022). A review of technical and regulatory limits for hydrogen blending in natural gas pipelines. [sciencedirect.com/science/article/abs/pii/S0360319922050923?via%3Dihub](https://www.sciencedirect.com/science/article/abs/pii/S0360319922050923?via%3Dihub)

Faulkner, X. (2003). Sample Sizes for Usability Studies: Additional Considerations. *Human Factors: The Journal of the Human Factors and Ergonomics Society*. <https://doi.org/10.1177/001872089403600215>

Forndal L. et al. (2022). System Study of the Techno-Economic Potential of a Hydrogen System A case study of Power to Mobility and Power to Power hydrogen systems, stand-alone or integrated with a CHP.

<https://liu.diva-portal.org/smash/record.jsf?pid=diva2%3A1666533&dswid=-796>

Fraunhofer ISE, (2021). «STROMGESTEHUNGSKOSTEN ERNEUERBARE ENERGIEN 2021». http://www.ise.fraunhofer.de/content/dam/ise/de/documents/publications/studies/DE2021_ISE_Studie_Stromgestehungskosten_Erneuerbare_Energien.pdf

Hafner M. & Luciani G. (2022). *The Palgrave Handbook of International Energy Economics*, Palgrave McMillan.

Mongird K. et al. (2020). 2020 Grid Energy Storage Technology Cost and Performance Assessment, U.S Department of Energy. www.pnnl.gov/sites/default/files/media/file/Final%20-%20ESGC%20Cost%20Performance%20Report%2012-11-2020.pdf



Papadias D. D. et al. (2021). Bulk storage of hydrogen.

<https://www.sciencedirect.com/science/article/abs/pii/S0360319921030834>

Reksten A. H. et al., 2022. Projecting the future cost of PEM and alkaline water electrolyzers; a CAPEX model including electrolyser plant size and technology development.

www.sciencedirect.com/science/article/pii/S0360319922040253?via%3Dihub

Schiebahn S. et al. (2015). Power to gas: Technological overview, systems analysis and economic assessment for a case study in Germany.

www.sciencedirect.com/science/article/abs/pii/S0360319915001913

Saputra, M., et al. (2022). Effectiveness of Wireless Network Roaming Access Point using PEAP Security System to Improve Internet Access. Jurnal Sisfotek Global. <https://doi.org/10.38101/sisfotek.v12i1.450>

Scandurra, T., et al. (2008). From User Needs to System Specifications: Multi-disciplinary Thematic Seminars as a Collaborative Design Method for Development of Health Information Systems. Journal of Biomedical Informatics. <https://doi.org/10.1016/j.jbi.2008.01.012>

Stehly, T. et al. (2020). 2019 Cost of Wind Energy Review, National Renewable Energy Laboratory ,TP-5000-7847.

Tan, Y., & Bishu, Z. (2002). Which is a Better Method of Web Evaluation? A Comparison of User Testing and Heuristic Evaluation. Proceedings of the Human Factors and Ergonomics Society Annual Meeting. <https://doi.org/10.1145/1125021.1125115>

Valente, A., et al. (2016). The Goals Approach: Enterprise Model-Driven Agile Human-Centered Software Engineering. https://doi.org/10.1007/978-3-319-44902-9_17

v. Leeuwen C. et al. (2018). D8.6: Innovative large-scale energy storage technologies and Power-to-Gas concepts after optimization, Store &Go project.

Virzi,, R. A. (1992). Skip and Scan Telephone Menus: User Performance as a Function of Experience. Proceedings of the Human Factors and Ergonomics Society Annual Meeting. <https://doi.org/10.1518/107118192786751763>

Pelekis, S., Sarmas, E., Georgiadou, A., Karakolis, E., Ntanos, C., Dimitropoulos, N., et al. (2023). TwinP2G: A digital twin architecture for optimal power-to-gas planning. 21st edition of the International Conference on e-Society.



9 Appendix

9.1 Flexibility Analytics and Register (FAR) service

9.1.1 FAR service User Interface features

This section aims to provide a brief overview on User Interface (UI) related features, as an application, that can be offered to each user (e.g., resources owner, aggregator). This application is, accordingly, integrated with the OneNet connector to allow the user to share meta-data on the resource/resource groups to the FAR service and in turn establish access on several flexibility and grid analytics. The main tabs provided in the application are:

- *Resources*: the user can access the Flexibility Service module and select “Add Resource” button, where a window with multiple tabs to be filled-in will pop-up. The registration of the Resource requires several information which are divided in three tabs (Base Info, location, and Details). (see Figure 134)
- *Resource Groups*: this tab is on orchestrating the portfolio on clusters/groups based on their technical characteristics.
- *Product definitions*: the user can access the product definitions to view the available product/services available in their area. The user might click on a specific product to view the eligibility details and minimum requirements. (see Figure 135)
- *Flexibility analytics*: is the tab where the user can retrieve from FAR service flexibility related analytics on his resource group portfolio. At this phase a basic feature implementation presents to the user information on a table form. This tab has incorporated button “Flexibility Analytics” which is integrated with the NGSI-LD API of the OneNet Connector to retrieve data from FAR (see Figure 136).
- *Grid analytics*: this tab is similar to the flexibility analytics one, yet it present grid related aggregated analytics of installed/available temporal flexibility which is retrieved also from FAR service with another service agreement as it is explained in the next section. At this point only the button “Grid Analytics” is implemented that is integrated with OneNet Connector to retrieve analytics from FAR; in the final release some visuals will be presented to the user.
- *Consent Management*: this tab records all the consents that a Flexibility Service Provider (FSP)/Aggregator from residential user to represent the resources for flexibility services. (see Figure 137)



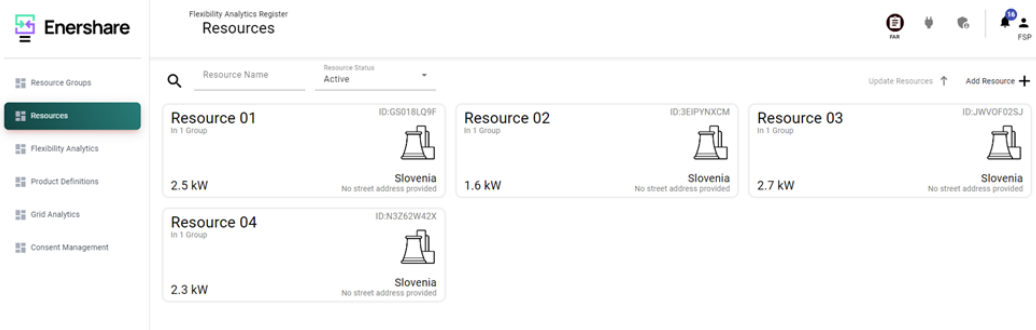


Figure 134: User Interface for storing locally (i.e., at Aggregator or User system) data on resources/resource groups

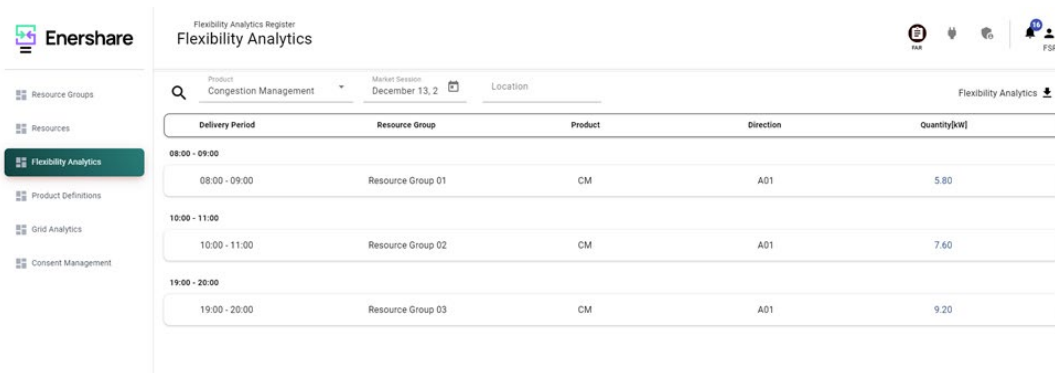
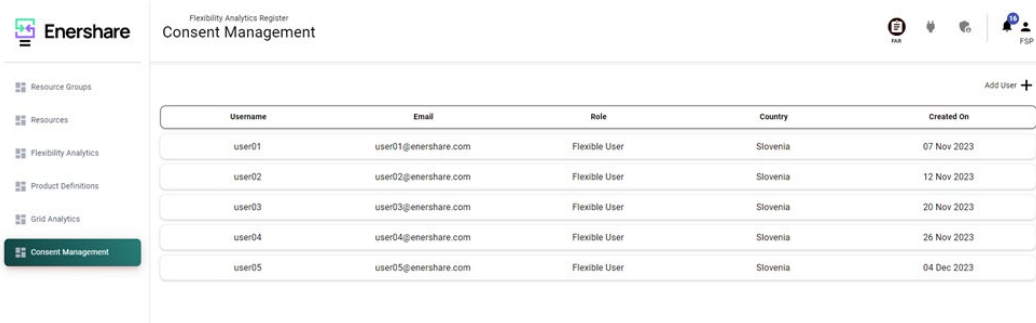


Figure 135: Flexibility Analytics tab, visual dashboard and “Flexibility Analytics” button integrated with OneNet Connector



Figure 136: Architecture of the integration of the FAR with Enershare data space using OneNet connector (NGSI-LD APIs)





Username	Email	Role	Country	Created On
user01	user01@enershare.com	Flexible User	Slovenia	07 Nov 2023
user02	user02@enershare.com	Flexible User	Slovenia	12 Nov 2023
user03	user03@enershare.com	Flexible User	Slovenia	20 Nov 2023
user04	user04@enershare.com	Flexible User	Slovenia	26 Nov 2023
user05	user05@enershare.com	Flexible User	Slovenia	04 Dec 2023

Figure 137: Consent management tab, Flexibility Service Provider representing resources

9.1.2 FAR service integration with OneNet Connector

For the integration of FAR service with a data space instance, two users have been created in the OneNet Middleware, one referring to the service provider (FAR_provider) and a FSP user (ed-new1). For the two individual users there have been installations/deployments of the OneNet connector to allow their interconnectivity with the dataspace.

From the Service Provider' point of view respective service offerings (e.g., District Heating Grid Aggregated Flexibility, Substation based temporal data forecasts, etc) for the analytics are registered in the OneNet Middleware, which in turn maintains meta-data descriptions user for the service discovery (see Figure Figure 138 & Figure 139).

From the service user point of view, one can search on the middleware for the available service offerings and submit requests accordingly. Any request has to be assessed by the service provider for acceptance or rejection (see Figure 140). Finally, once a service is agreed among service provider and service user, the user can make requests to obtain the respective analytics via the connectors (see Figure 141). The analytics retrieval is achieved in a decentralized manner calling the respective NGSI-LD API.

Available Services

Select from the Available Services

Category	Title	User Offering	Created On	Input Profile	Input Data Source	Profile Format	Profile Description
+ Generic- User defined service ▶ Generic- User defined service ▶ District Heating Grid Aggregated Flexibility for Balancing Service	Enershare_Analytics	Enershare ▶ FAR_provider	14/12/2023 15:02			json	
+ Measurements and Monitoring ▶ Grid State ▶ State Estimation Data	Substation based temporal data forecasts	Enershare ▶ COP_service_provider	13/12/2023 16:03				

Figure 138: OneNet Connector UI: Available services on the ecosystem (FAR and COP service providers)



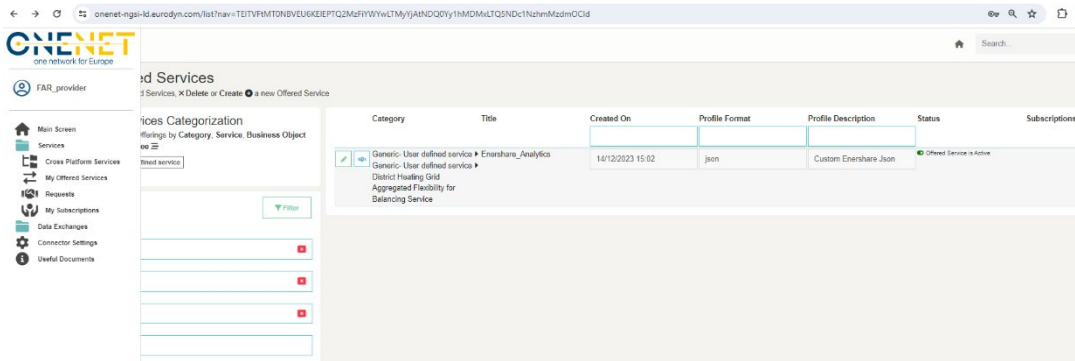


Figure 139: OneNet Connector UI: Available services on the ecosystem (FAR provider's service offering)

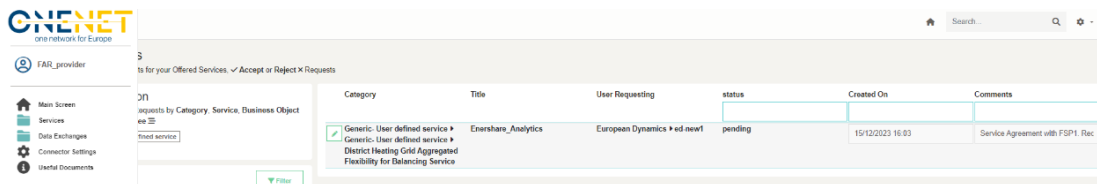


Figure 140: Service request for subscription from user ed-new1 to FAR_provider



Figure 141: Available data analytics to be retrieved from FAR service

9.2 Cross-operators' portal (COP) service

9.2.1 COP User Interface features

An application is provided to DSO and DHO to streamline the process of interacting with the COP and the data space. The UI features resemble the FAR application, as there is a "Resource Group" for viewing registered resource. It shall be noted that a separate UI/application is



ENERSHARE has received funding from [European Union's Horizon Europe Research and Innovation programme](#) under the Grant Agreement No 101069831

provide to the DHO for viewing available heating flexibility and retrieving relevant grid and shared cross-domain flexibility analytics.

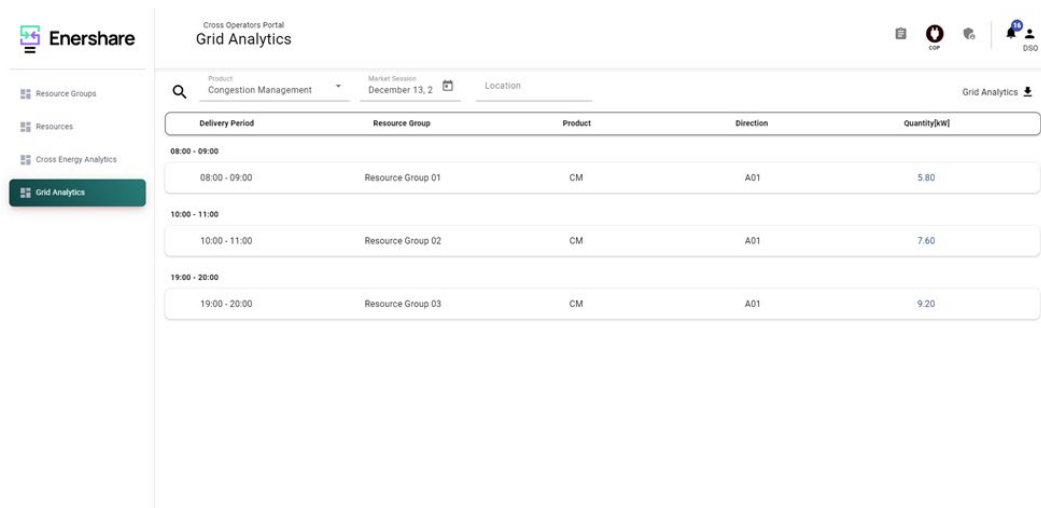


Figure 142: Grid Analytics tab on COP provided application

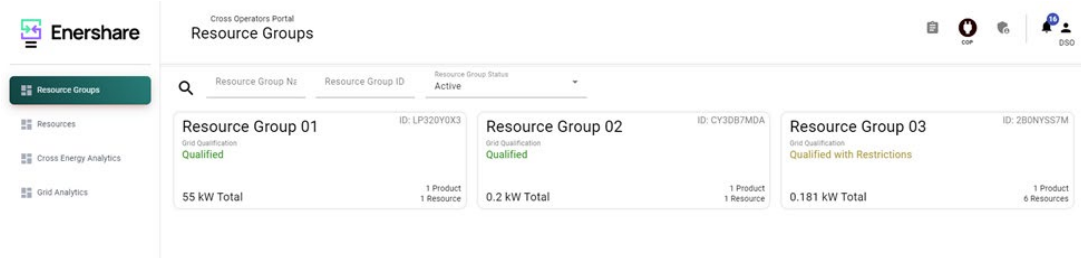


Figure 143: Resources groups tab on the electricity grid

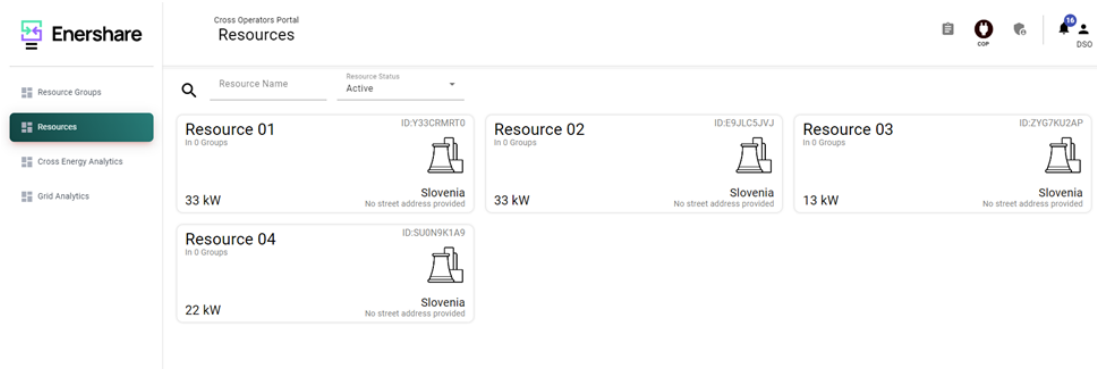


Figure 144: Available resources on the electricity grid

