A novel smart grid architecture that facilitates high RES penetration through innovative markets towards efficient interaction between advanced electricity grid management and intelligent stakeholders

H2020-GA-863876

First version of FLEXGRID S/W prototype

Deliverable D6.2

Document Information

Scheduled delivery	30.09.2021
Actual delivery	30.09.2021
Version	Final
Responsible Partner	ETRA

Dissemination Level

PU Public

Contributors

Lars Herre (DTU), Rahul Nellikkath (DTU), Eléa Prat (DTU), Spyros Chatzivasileiadis (DTU), Elena Leal (ETRA), Germán Martínez (ETRA), Prodromos Makris (ICCS), Dimitrios Vergados (ICCS), Konstantinos Steriotis (ICCS), Nikolaos Efthymiopoulos (ICCS), Maria-Iro Baka (UCY), Marios Kynigos (UCY), Andreas Kyprianou (UCY), Stylianos Loizidis (UCY), Christina Papadimitriou (UCY), Spyros Theocharidis (UCY), Domagoj Badanjak (UNIZG-FER), Vesna Županović (UNIZG-FER)

Internal Reviewers

Gesa Milzer (NODES), Prodromos Makris (ICCS)

Acknowledgements

The research leading to these results has received funding from the EC Framework Programme HORIZON2020/2014-2020 under grant agreement n° 863876.

Glossary of Acronyms

Project management terminology

Acronym	Definition
D	Deliverable
HLUC	High Level Use Case
MS	Milestone
WP	Work Package
UCS	Use Case Scenario

Technical terminology

Acronym	Definition
AC	Alternating current
AFAT	Automated Flexibility Aggregation Toolkit
API	Application Programming Interface
ATP	Automated Trading Platform
B2B/B2C	Business to Business / Business to Consumer
DFMCT	Distribution Flexibility Market Clearing Toolkit
DSO	Distribution System Operator
ES	Energy Service
ESP	Energy Service Provider
FMCT	Flexibility Market Clearing Toolkit
FMO	Flexibility Market Operator
FSP	Flexibility Service Provider
FST	FlexSupplier's Toolkit
GUI	Graphical User Interface
MTU	Market Time Unit
PV	Photovoltaic
REST	REpresentational State Transfer
TSO	Transmission System Operator

Specific terminology definition

Word	Definition
Baseline	The baseline shows the scheduled demand during the selected date
Cost	Amount of money an actor will have to pay for something
Price	Monetary value of something
Revenue	Money generated from participating in a market or a service
Benefit	Difference between costs and revenues

Table of Contents

List of Figures

List of Tables

Document History

This prototype deliverable includes the first version of the S/W integration and validation results of FLEXGRID platform. An initial version was demonstrated during the 1st official review meeting (i.e. Month 20), while the release of the first integrated FLEXGRID system prototype took place in Month 24.

Table 1: Document History Summary

Revision Date	File version	Summary of Changes
26/11/2020	v0.1	Draft ToC circulated among all partners
08/01/2021	v0.2	Updated draft ToC shared with main contributing partners
13/01/2021	v0.3	Final ToC presented to the whole consortium with the agreement of all involved partners
01/03/2021	v0.4	All partners contributed their 1st round inputs
16/03/2021	v0.5	ETRA integrated text and provided comments. First review provided by NODES
24/03/2021	v0.7	NODES reviewed the pre-final version and provided comments for changes/enhancements
26/03/2021	v0.8	ETRA integrated all comments and forwarded the final version to the coordinator
31/03/2021	v1.0	Coordinator made some final enhancements and submitted the final version (M18) in ECAS.
20/09/2021	v1.5	All partners made amendments to the GUIs and APIs according to the updates from the research WPs 3-5.
30/09/2021	v2.0	Coordinator made some final enhancements and submitted the final version (M24) in ECAS.

Executive Summary

This report is an official deliverable of H2020-GA-863876 FLEXGRID project dealing with the first version of FLEXGRID S/W prototype. It includes the outcomes of task 6.2 “Design of APIs and S/W Development” and task 6.3 “GUIs and integration activities”. Along with this document, as the main outcome of task 6.2 in M18 is also delivered an initial draft of the API that will be exposed by UCS 2.3 “Maximize ESP’s stacked revenues”, and as the main outcome of task 6.3 (also in M18) the section of the GUI implementing it.

For this first version of both tasks, it has been selected the UCS 2.3 to be demonstrated during the Period 1 review meeting that will take place on 22nd June 2021. Parts of this document D6.2 are based on the results from task 6.1 as the outcome is closely related (or relevant) for the API and GUI implementation.

It has to be highlighted that, as part of the work that has been performed within T6.2 and T6.3 until now, all the APIs 1st draft definitions (T6.2) and mock-ups (T6.3) presented here (and also the initial version of UCS 2.3) will be modified during the second half of the project as there are still some ongoing tasks within WP3, WP4 and WP5 that have a direct impact on these two tasks. The final work will be reflected in D6.3 “Final version of FLEXGRID S/W prototype” delivered in M33.

To address all relevant aspects to achieve the scope of both tasks, the deliverable is structured in 5 different chapters.

The first chapter deals with the executive summary of the deliverable contents, the description and definition of the FLEXGRID APIs and the toolkit of the GUIs that are being developed within the project, as well as the definition of the methodology used for the implementation of the task here executed.

The second chapter sum up the developed Use Cases Scenarios showing the relation between them and the main ATP configuration available for each one with the objective of being an introduction for understanding the API and GUI used for the main platform of the project.

The third chapter includes the developments regarding the GUI presentation, in particular considering the different functionalities detailed in D6.1 “Data Model of FLEXGRID architecture” and all the mock-ups and their descriptions. A separation per UCS is done in order to show the different GUIs to run the algorithms developed within the WP3, WP4 and WP5. As for the API the version of the GUI here described may be different from the one delivered at the end of the project and documented in D6.3.

Finally, some conclusions are presented, containing a summary of the main results of the work performed in M18 and presented in the current deliverable.

The fourth chapter describes the FLEXGRID software architecture to correctly implement the required APIs in T6.2. Furthermore, this chapter explains the integration of all selected UCS within the ATP and the current status of each API based on the methodology defined based

on the work done by research partners in WP3 “Automated flexibility aggregation energy market development and management as a service”, WP4 “Innovative ESS aware Business Modelling for ESPs and interaction with advanced RES & Market Forecasters” and WP5 “Optimal Power Flow and interaction between network operators and markets”. As it has been highlighted, the version of the API described here may be slightly different from the one delivered at the end of the project and documented in D6.3.

Note: Although the information here presented is showing the 1st version of the ATP integration and the GUIs to be followed in next developments, slightly modification can be applied during the last stage of the project as some of the main tasks that provide inputs are still ongoing in M24.

1 Introduction

1.1 Purpose of the document

The main objective of this report is to define the first version of the interface for the FLEXGRID ATP solution through the development of the needed APIs and GUI. With the term of API, we are referring to an interface or communication protocol to allow interaction and data exchange over the entire software platform and between the different modules/ algorithms developed in the different WP of the project.

The principal connection between the stakeholders and the FLEXGRD platform will be made via the ATP. The Platform is able to provide as a service the composition and the operation of energy markets in order to interact with each other, for offering competitive ESs by means of the advance flexibility trading.

Considering the current different market actors and their present and potential future responsibilities we made a categorization between the FMO, DSO, ESP and aggregator user aiming at providing an interface that facilitates the interaction between the aforementioned users and the FLEXGRID platforms. We will provide a detailed description of the GUIs that will be developed and the APIs that are required to achieve the principal goals of the project: i) easily and effectively provide access advanced Energy Services (ESs), ii) to facilitate a dynamic and efficient interaction with the electricity grid and the stakeholders, and iii) to automate and optimize the planning and the operation of their ESs. For each actor a different main interface is developed and will be integrated in the Automated Trading Platform and connected to the main API. The aggregator is supposed to use the GUI that facilitates the interaction of the AFAT (Automatic Flexibility Aggregation Tool) with the different algorithms. The FMO and DSO users will mainly interact through the FMCT, and the ESP user through the FST designed GUIs.

1.2 Scope of the document

This document presents the first version of the S/W integration and validation results of the FLEXGRID ATP into the context of the tasks 6.2 and 6.3. Both tasks will officially start on M18 and M25 respectively, but for achieving the main objective of this deliverable some preliminary definitions and developments are made. Based in the preliminary assumptions an initial version of the platform will be demonstrated during the 1st official review meeting (i.e. Month 20).

The outcomes of tasks 6.2 and 6.3 that are described in this deliverable are based on previous work that was conducted during the first twelve months of the project as represented in the following deliverables:

- D2.1 – “FLEXGRID use case scenarios, requirements’ analysis and correlation with innovative models”: Detailed description of the FLEXGRID’s use cases scenarios and list of the major stakeholders/users that will interact with FLEXGRID Automated Trading Platform (ATP).
- D2.2 – “The overall FLEXGRID architecture design, high-level model and system specifications “: Initial description of the FLEXGRID’s software architecture and of the

internal architecture for the different modules conforming the platform (i.e. ATP, AFAT, FST, FMCT).

- D6.1 – "Data Model of FLEXGRID architecture": Detailed data models for all the mathematical models and algorithmic solutions that have been defined in the first phase of research work within WP3, WP4 and WP5.
- WP3 – "Automated flexibility aggregation energy market development and management as a service": Developments done in WP3 and collected in D3.1 and D3.2, in particular the algorithms that will be implemented in a S/W toolkit (AFAT) and connect to the core FLEXGRID ATP
- WP4 – "Innovative ESS aware Business Modelling for ESPs and interaction with advanced RES & Market Forecasters": Developments done in WP4 and collected in D4.1 and D4.2, in particular the algorithms that will be implemented in a S/W toolkit (FST), and connect to the core FLEXGRID ATP
- WP5 – "Optimal Power Flow and interaction between network operators and markets": Developments done in WP5 and collected in D5.1 and D5.2 in in particular the algorithms that will be implemented in a S/W toolkit (FCMT), and connect to the core FLEXGRID ATP
- Work done by all research partners (i.e. ICCS, UCY, DTU, UNIZG) to identify the most important UCS functionalities that will be integrated in the FLEXGRID ATP.
- Work done by industrial partner based on their expertise and business models of the wholesale market and ETRA as industrial software developer defines the first version of the API and GUIs.

Considering the work already done and outlined in the different deliverables, in the first version of the SW development we will focus on the development and integration of the needed APIs and GUIs for one selected UCS to demonstrate and prove the main functions of the FLEXGRID platform. This first integration will be the base for the implementation and development of the other UCSs.

After the first version of the S/W development (i.e. in M18) and demonstration of the UCS 2.3[1] (i.e. in M21), a set of specific activities will take place in order to elaborate on data modeling work's results as follows:

[1] UCS 2.3 deals with the research problem of ESP's stacked revenue maximization via its co-optimized participation in 4 different markets. We have selected this UCS as it is the most mature in terms of implementation. We have defined a specific S/W integration process for UCS 2.3, which will also be followed by all other UCS during Period 2.

- Once all data models (task 6.1) are translated in json format and respective swagger files are ready to use, the deployment of all APIs and GUIs will continue.
- Another major task will be the integration of all algorithms from the selected UCS in the FLEXGRID ATP as well as the testing and validation activities that will take place during Period 2.
- Based on the mock-ups defined in section 3 of this deliverable all GUIs will be integrated in the ATP.
- Design and development of the central Data Base, where all results of the backend modules will be stored, as well as some general data (this still has to be defined) required for the developed algorithms or market design.
- For this specific Data Base, an additional API will be designed and developed to allow the basic CRUD (create, read, update, delete) operations.

1.3 Implementation Methodology

In order to define the API and GUIs, a simple methodology was selected so that all involved partners could define the required information for further system integration by ETRA.

For the implementation of the APIs the information was requested in separate excel files per UCS with all detailed inputs and outputs and then converted into a .yaml folder that is needed in swagger. A more detailed explanation of the format of the excel file is defined in section 2 where the API implementation is detailed. Nevertheless, as an overview of the required information, on each one of those spreadsheets (1 file for each service), each module owner has to fill in the following information: i) General info, ii) Services to be delivered, iii) Inputs, iv) Outputs.

For the definition of the GUIs and the APIs the information coming from the data model is essential. A more detailed definition of the data model information is defined on D6.1. For the definition of the data model, the APIs and the GUIs series of iterations with the research partners were needed. A first version of the GUIs and APIs is a still ongoing activity to be updated in following deliverables. Similar steps as the ones describe in D6.1 were followed for developing the S/W of the project. So far, a first version of the S/W development the first three steps were achieved for all UCS:

- **Step 1:** For every UCS that has been short-listed to be integrated in FLEXGRID ATP, a first analysis of the information of the data model was made.
- **Step 2:** Collaboration with research partners to design the FLEXGRID ATP frontend (GUI) services in a user-friendly manner considering the initial definition of inputs and outputs.
- **Step 3:** The first mock-ups were defined in accordance with the research partners and the specifications per UCS detailed in task 6.1.
- **Step 4:** After the initial GUI designs and user views have been agreed, the final version of the GUIs to be integrated in the ATP will be defined.

2 Use Cases Scenarios

2.1 Use Case Scenarios definition

Following, a brief description of the UCS is included to better understand how the API configuration is related with the functionalities needed for the different users of the ATP. A more detailed definition of each UCS can be found in D2.1 and in the respective technical WP.

UCS 1.1: Distribution network aware flexibility market clearing via FLEXGRID ATP
Though this UCS, a market considering the distribution networks, their constraints, and the location of the sources that could provide flexibility to decrease the occurrences of line congestions and voltage deviations is created and optimized. Instead of a market clearing considering all bids and clearing once and for all, the FLEXGRID concept would imply having continuously matching bids. (Detailed information in D5.1 and D5.2)

UCS 1.2: Market-based local congestion management using FLEXGRID ATP
Using an AC-OPF model makes it possible to anticipate/estimate the flow in each line of the distribution network and thus to identify/forecast the line congestions. (Detailed information in D5.1 and D5.2)

UCS 1.3: Market-based local voltage control in distribution network operation
Using an AC-OPF model makes it possible to anticipate/estimate the voltage level at every node of the distribution network and thus to identify/forecast voltage deviations. (Detailed information in D5.1 and D5.2)

UCS 2.1: ESP minimizes its OPEX by optimally scheduling
Through contractual arrangements with various potential providers of flexibility and services provided to the DSO/TSO and BRPs, they can be considered as a coupling point between the retail and the wholesale market. The heterogeneous mixture of services they can provide and acquire has resulted (in the past) in non-negligible operating costs (OPEX). (Detailed information in D4.1 and D4.2)

UCS2.2: ESP minimizes CAPEX by making optimal investments on RES and FlexAssets
An analysis identifies the most attractive electricity markets to participate in, considering technical constraints and CAPEX-to-profit ratio. Furthermore, due to possible future techno-economic trends, it considers a multi-stage investment plan in order to intelligently acquire the assets with the lowest CAPEX possible while obeying all of the constraints and requirements on the optimal siting and sizing of the relevant assets. (Detailed information in D4.1 and D4.2)

UCS 2.3: ESP maximization of stacked revenues
It is proposed a bi-level model in order to formulate the ESP's problem to calculate its optimal bidding strategy and the charging/discharging schedule of the Battery Storage Units (BSUs). The main novelty of FLEXGRID algorithm is that it co-optimizes the operation and the bidding strategy of the BSUs' owner (i.e. ESP) in both transmission-level and distribution-level markets. (Detailed information in D4.1 and D4.2)

UCS 4.1: Manage a FlexRequest

ESP/aggregator efficiently responds to FlexRequests made by TSO/DSO/BRP by optimally orchestrating its aggregated flexibility portfolio of end energy prosumers. (Detailed information in D3.1 and D3.2)

UCS 4.2: Manage a novel B2C flexibility market

An aggregator/retailer operates an ad-hoc B2C flexibility market with its end energy prosumers by employing advanced pricing models and auction-based mechanisms. Through FLEXGRID ATP, the aggregator user will be able to run various "what-if" simulation scenarios in order to determine better ways (via retail pricing schemes) to operate a novel B2C flexibility market, in which end energy prosumers compete with each other. In other words, the aggregator will run a retail pricing algorithm to test and evaluate the impact that new FlexContracts (with its end users) would have on several KPIs such as: i) aggregator's revenues, ii) aggregated end users' welfare, iii) quantity of flexibility offered to the system, iv) individual end user's welfare. (Detailed information in D3.1 and D3.2)

UCS 4.3: Create a FlexOffer

Aggregator/ESP maximizes its profits by dynamically orchestrating distributed FlexAssets from its end users in order to optimally participate in several energy markets. In FLEXGRID ATP, the aggregator user will be able to utilize the Automated Flexibility Aggregation Toolkit (AFAT) to make efficient FlexOffers in near-real-time balancing markets (TSO) and Distribution Level Flexibility Market (DLFM). (Detailed information in D3.1 and D3.2)

UCS 4.4: Forecasting services

ESP exploits FLEXGRID's advanced forecasting services to forecast market prices and FlexAssets' state and curves in the future
(Detailed information in D4.1 and D4.2)

2.2 Real Business Applicability of FLEXGRID research

H2020 FLEXGRID is a RIA project focusing on low Technology Readiness Levels (TRLs). This means that some of the FLEXGRID research threads (cf. pure research WPs 3-5) assume future energy market architectures and new market setups, which are mainly based on the introduction of a novel Distribution Level Flexibility Market (DLFM). Within WP6, we have short-listed the FLEXGRID UCS that are closer (or else compatible) to the existing energy market architecture and standards in the EU area. We have chosen this short-list of FLEXGRID functionalities to be integrated in the FLEXGRID ATP in order to boost the platform's exploitation potential by serving the real business needs of today's energy market stakeholders.

For example, an FMO user will be able to run a distribution network-aware market clearing process for three main flexibility products, namely: 1) energy (UCS 1.1), 2) active power reserve (UCS 1.2), and 3) reactive power reserve (UCS 1.3). A Reactive Distribution Level Flexibility Market architecture is assumed because this is compatible with today's EU regulatory framework. More specifically, the new DLFM that is introduced follows the existing day-ahead energy (MO) and reserve markets (TSO), while it precedes the existing

near-real time balancing market. Especially for UCS 1.1, this could also be applied in a Proactive DLFM (P-DLFM) setup in the future.

Regarding the ESP user, s/he will be able to get four FLEXGRID services via the FlexSupplier's Toolkit (FST), namely: 1) minimize OPEX (UCS 2.1), 2) minimize CAPEX (UCS 2.2), 3) maximize stacked revenues (UCS 2.3), and 4) forecast PV generation and market prices. In all these UCS, the R-DLFM architecture is assumed. However, it should be noted that all the FST services are applicable in the no-DLFM architecture (i.e. there is no DLFM but only the existing markets), which is the current regulatory situation in Europe.

Finally, as of the aggregator user, s/he will able to get three FLEXGRID services via the Automated Flexibility Aggregation Toolkit (AFAT), namely: 1) maximize its revenues by optimally responding to FlexRequests and minimize its associated payments to the end users (UCS 4.1), 2) propose personalized FlexContracts to end users that are economically beneficial for both them and the aggregator (UCS 4.2), 3) automatically and dynamically create a FlexOffer that best represents the current status of its portfolio in order to participate in either TSO's reserve market or DLFM or both of them (UCS 4.3).

Table 2 Relation with the existing regulatory framework and real business needs of the involved market stakeholders

Use Case Scenario	Partner	Scope	Programming language
UCS 1.1	DTU	Assume a P-DLFM architecture. The FMO wants to clear an energy market, i.e., DLEM, with Offers and Requests from different ESPs, while ensuring that the resulting power flows are feasible for the network.	Python
UCS 1.2	DTU	Assume a R-DLFM architecture. The FMO wants to clear an active power reserve market, i.e., DLFM, with FlexOffers from the DSO and FlexRequests from different ESPs, while ensuring that the resulting power flows are feasible for the network.	Python
UCS 1.3	DTU	Assume a R-DLFM architecture. The FMO wants to clear a reactive power reserve market, i.e., DLFM, with FlexOffers from the DSO and FlexRequests from different ESPs, while ensuring that the resulting power flows are feasible for the network.	Python
UCS 2.1	UNIZG	Assume a R-DLFM architecture. The ESP wants to minimize its OPEX by optimally scheduling its FlexAssets to respond to the FlexRequests without paying stiff penalties in the balancing market.	Python
UCS 2.2	UNIZG	Assume a R-DLFM architecture. The ESP wants to minimize its CAPEX by optimally investing in new FlexAssets in the future.	Python
UCS 2.3	ICCS	Assume a R-DLFM architecture. The ESP wants to maximize its stacked revenues by co-optimizing its participation in various markets (including DLFM or not)	Python

		instead of simply participating in each one of them individually in a sequential manner.	
UCS 4.1	UCY	Assume a R-DLFM architecture. The aggregator wants to maximize its profits by optimally responding to FlexRequests. This translates to maximization of its revenues and minimization of the associated payments to the end users.	Python
UCS 4.2	ICCS	Assume a novel B2C flexibility market which uses a FlexRequest as input. The aggregator user wants to determine better ways (via retail pricing schemes) to operate a novel B2C flexibility market, in which the end energy prosumers compete with each other. It also wants to evaluate the impact that new FlexContracts (with its end users) would have on several KPIs such as: aggregator's revenues, aggregated end users' welfare, quantity of flexibility offered to the system, individual end user's welfare.	Python
UCS 4.3	ICCS	Assume a R-DLFM architecture. The aggregator wants to determine/create a FlexOffer that best represents the current status of its portfolio and submits it to the FLEXGRID ATP. This FlexOffer may be used either in the: i) TSO's reserve market (cf. "no-DLFM" architecture), or ii) proposed DLFM market operated by the FMO to solve DN-level problems.	Python
UCS 4.4 - PV	UCY	The ESP/aggregator wants to forecast the PV generation of its portfolio in a day-ahead and intra-day context. This service is offered on top of all the other FST and AFAT services described above.	R Language
UCS 4.4 - Price	UCY	The ESP/aggregator wants to forecast the market prices (only applicable auction-based markets) in a day-ahead and intra-day context. This service is offered on top of all the other FST and AFAT services described above.	Python

3 Graphical User Interface

3.1 Introduction

This section is related to Task 6.3 “GUIs and integration activities”. As depicted in section 4.3.1 of D2.2 “The overall FLEXGRID architecture design, high-level model and system specifications”, the toolkit of GUIs is the frontend that the different users will use for interacting with the functionalities offered by the modules developed within WP3, WP4 and WP5

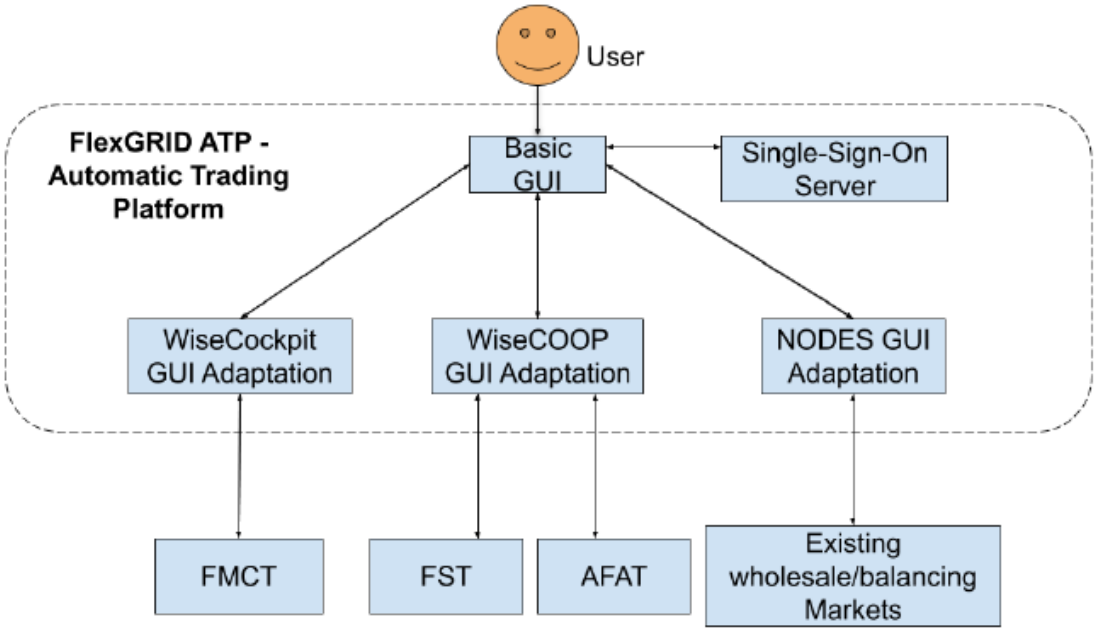


Figure 1: The Automated Trading Platform (ATP) internal architecture

Before the developing the GUIs some common agreements are required how input and output data of the different modules shall be transmitted via the GUI. Therefore, some mockups of each UCS are defined and documented in D6.1 “Data Model of FLEXGRID architecture”

It has to be highlighted that **this GUI design process is still ongoing**, so the final version of the GUIs may be different from the mockups presented in this deliverable, and thus this will be reflected on D6.3 “Final version of FLEXGRID S/W prototype” delivered in M33. This is an iterative process that will continue during the following months, but the work performed until now and documented here, will be the basis for the following discussions about the final version of the FLEXGRID GUIs.

In Section 3.3 the actual version (in M18) of the designed mockups pfor all the UCS and some descriptions about them have been included, except for UCS2.3 which is the one selected to be implemented before M18 and demonstrated during the 1st review of the project. Therefore some real screenshots and description are provided in Section **Error! Reference source not found.**, although it can still be modified during the second half of the project.

3.2 Functionalities general overview

The ATP front-end panel is divided in three different sections or categories to facilitate ethe use of the application and make it easier for the market actors. In the following sections a brief explanation in general terms of the possibilities or the ATP are explained in general. All the views will be available only for the allowed market actors (Table 6 GUIs functionalities). A more detailed explanation related the front-end with the different UCS is explained in section 3.3.

Historical data view

For the market actors (DSO, TSO, FMO, aggregator…) allowed to use the ATP functionalities in this view it is possible to see all the historical data available for previous operations with a series of useful information to better understand the different operation carried out in a different period. In this view it is possible to search for a specific value, delete any operation or see a detailed explanation of each activity. In a simple way, this view allows the user to see what happened in past periods of time.

To have this functionality available it is needed to store the information the Data Base developed in the context of FLEXGRID project and linked via a different API with the modules' information.

Configuration

The configuration is the main part of the ATP as it is the link between the application user and the algorithms. As it is defined, at the configuration view it is possible to indicate all the inputs needed for the algorithm calculation and optimization of the possibilities considering the market, the gird, the assets, the final user and more.

Although some configuration items can be similar, it will depend on the different algorithms to run and the different UCS to prove. In the following sections a deep description of all the possibilities for the different UCS selected will be defined with the objective to understand how it works and how each user (DSO, TSO, ESP, FMO…) can interact with the ATP and the modules developed in the FLEXGRID project.

Results

The "Results" view allows the user to see the results of the operations made according to his role registered on the ATP, such as. the revenues from the different markets available and the energy use in the market, relevant output from the algorithm calculations.

The general functionalities defined have a specific structure and information according to the UCS and the associated module as shown in Table 3-5:

Table 3 HLUC 1 general functionalities

General	UCS1.1	UCS1.2	UCS1.3
Module	DFMCT	DFMCT	DFMCT
Historical view	**Flexibility market clearing** historical view	Market-based local **congestion management** historical view	Market-based local **voltage control** historical view

Configuration	**Flexibility market clearing** configuration	Market-based local **congestion management** configuration	Market-based local **voltage control** configuration
Result	**Flexibility market clearing** results	Market-based local **congestion management** results	Market-based local **voltage control** results

Table 4 HLUC 2 general functionalities

General	UCS2.1	UCS2.2	UCS2.3
Module	FST	FST	FST
Historical view	**OPEX** optimization historical view (with revenues/without revenues)	**CAPEX** optimization historical view	**Profits** optimization historical view (with revenues/without revenues)
Configuration	**OPEX** optimization configuration	**CAPEX** optimization configuration	**Profits** optimization configuration
Result	**OPEX** optimization results (with revenues/without revenues)	**CAPEX** optimization results	**Profits** optimization results (with revenues/without revenues)

Table 5 HLUC 4 general functionalities

General	UCS4.1	UCS4.2	UCS4.2	UCS4.4 (price)	UCS4.4 (PV)
Module	AFAT	AFAT	AFAT	AFAT	AFAT
Historical view	**FlexRequest** dispatch optimization historical view	**Real pricing** optimization historical view	**Flexibility offer** optimization historical view (with revenues/without revenues)	Market **price forecasting** historical view	**PV** generation **forecasting** historical view
Configuration	**FlexRequest** dispatch optimization configuration	**Real pricing** optimization configuration	**Flexibility offer** optimization configuration	Market **price forecasting** configuration	**PV** generation **forecasting** configuration
Result	**FlexRequest** dispatch optimization results	**Real pricing** optimization results	**Flexibility offer** optimization results (with revenues/without revenues)	Market **price forecasting** results	**PV** generation **forecasting** results

A quick overview of all the functionalities offered by each UCS to each type of user is presented on the following table:

Table 6 GUIs functionalities

Module	UCS	Functionality	User			
			Aggr	DSO	ESP	FMO
FMCT	UCS1.1	Flexibility market clearing historical view	✓	✓	✓	✓
		Flexibility market clearing configuration	✗	✗	✗	✓
		Flexibility market clearing results	✓	✓	✓	✓
	UCS1.2	Market-based local congestion management historical view	✓	✓	✓	✓
		Market-based local congestion management configuration	✗	✓	✗	✓
		Market-based local congestion management results	✗	✓	✓	✓
	UCS1.3	Market-based local voltage control historical view	✗	✓	✓	✓
		Market-based local voltage control configuration	✗	✓	✓	✓
		Market-based local voltage control results	✗	✓	✓	✓
FST	UCS2.1	OPEX optimizations historical view (with price)	✗	✗	✓	✗
		OPEX optimizations historical view (without price)	✗	✓	✗	✓
		OPEX optimization configuration	✗	✗	✓	✗
		OPEX optimization results (with price)	✗	✗	✓	✗
		OPEX optimization results (without price)	✗	✓	✗	✓
	UCS2.2	CAPEX optimizations historical view	✗	✗	✓	✗
		CAPEX optimization configuration	✗	✗	✓	✗
		CAPEX optimization results	✗	✗	✓	✗
	UCS2.3	Profits optimizations historical view (with price)	✗	✗	✓	✗
		Profits optimizations historical view (without price)	✗	✓	✗	✓
		Profits optimizations configuration	✗	✗	✓	✗
		Profits optimization results (with price)	✗	✗	✓	✗
		Profits optimization results (without price)	✗	✓	✗	✓
AFAT	UCS4.1	FlexRequest dispatch optimizations historical view	✓	✓	✗	✓
		FlexRequest dispatch optimization configuration	✓	✗	✗	✗
		FlexRequest dispatch optimization results	✓	✓	✗	✓
	UCS4.2	Real pricing optimization historical view	✓	✗	✗	✗
		Real pricing optimization configuration	✓	✗	✗	✗
		Real pricing optimization results	✓	✗	✗	✗
	UCS4.3	Flexibility offer optimizations historical view (with revenues)	✓	✗	✗	✗
		Flexibility offer optimizations historical view (without revenues)	✗	✓	✗	✓
		Flexibility offer optimization configuration	✓	✗	✗	✗
		Flexibility offer optimization results (with revenues)	✓	✗	✗	✗
		Flexibility offer optimization results (without revenues)	✗	✓	✗	✓
	UCS4.4 (price)	Market price forecasting historical view	✓	✗	✓	✗
		Market price forecasting configuration	✓	✗	✓	✗
		Market price forecasting results	✓	✗	✓	✗
	UCS4.4 (PV)	PV generation forecasting historical view	✓	✗	✓	✗
		PV generation forecasting configuration	✓	✗	✓	✗
		PV generation forecasting results	✓	✗	✓	✗

More information about the FLEXGRID use by each user is available in the section 3.3, but following some minor clarifications are made to better understand the Table 6:

- The aggregator user is able to see all the historical market results in the FMCT module because it participates in DLFM.
- The DSO user should be able to configure and submit a FlexRequest and view all historical data about its FlexRequests.

- The FMO users should be able to see all the FlexRequests made by the DSO and use them as an input for UCS1.2 and UCS 1.3.
- For the FST and AFAT modules only the ESP or aggregator are able to configurate inputs and the DSO and FMO can see historical data and results without considering the revenues.

More detailed information about FLEXGRID functionalities can be read in Section 6 of D2.1 "FLEXGRID use case scenarios, requirements' analysis and correlation with innovative models".

3.3 Functionalities for Design

In this section the actual status of all the designed mockups for all the UCS is presented, with the exception of the aforementioned UCS2.3 which will be documented in the next section.

Each UCS described in the following will contain:

- A definition of the users of the specific GUI.
- Mockups for the different possible displays that have been identified for covering all the needs for the purpose of the UCS.
- Detailed description of the actions per user to be performed on the interface and associated functionalities defined in 3.2.

3.3.1 UCS 1.1– DLFM clearing for the active power (energy) product

Flexibility market clearing historical view

Users: DSO, ESP, FMO, Aggregator

Main front-end view:

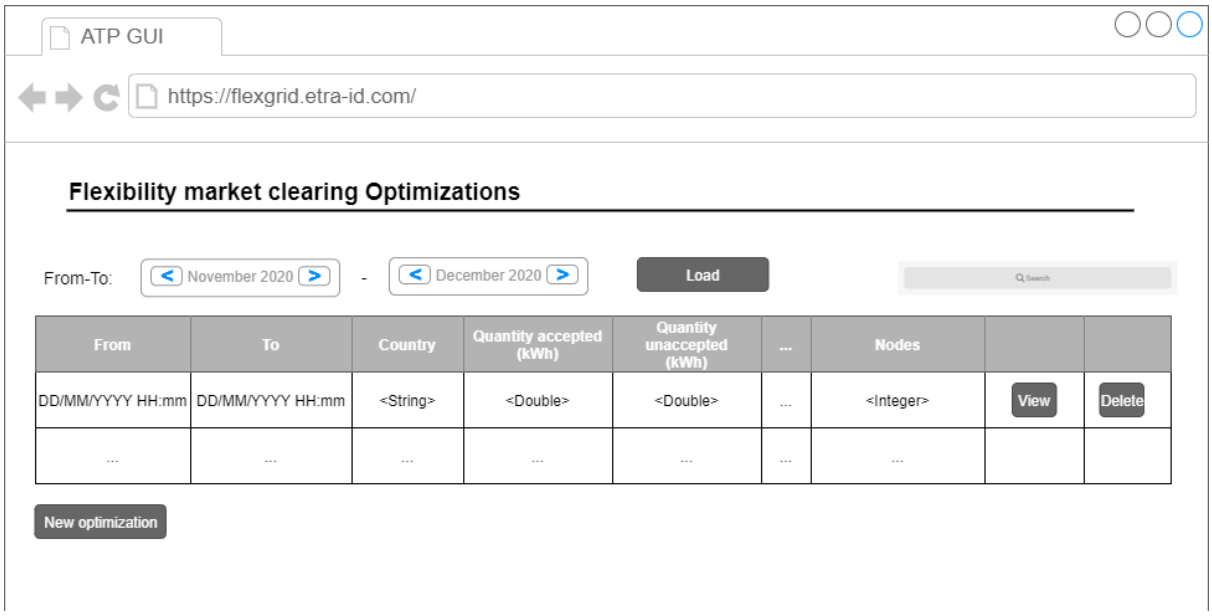


Figure 2: UCS1.1 - Market clearing historical view

Description:

- All the results available from previous flexibility market clearing optimizations stored in the DB will be listed (Figure 2) with the following information available:
  - The duration of the optimization (From, To).

- The country the optimization was run for (**Country**).
- The total amount of energy accepted for the market after running the algorithms (**Quantity accepted**); measured in kWh.
- The total amount of energy unaccepted (**Quantity unaccepted**); measured in kWh. It will be presented only if this value exists for this optimization.
- The number of nodes affected by the optimization (**Nodes**).

- It will be possible to search for a concrete value by using the text box above the table.
- By clicking on the "delete" button the selected optimization will be removed.
- By clicking on the "view" button a screen will be opened with the details of the selected optimization.
- By clicking in the "Load" button it will be possible to retrieve the stored information by selecting a range of dates
- By clicking on the "New optimization" button a new screen will be opened to allow a new optimization configuration.

Flexibility market clearing algorithm configuration

Users: FMO

Main front-end view:

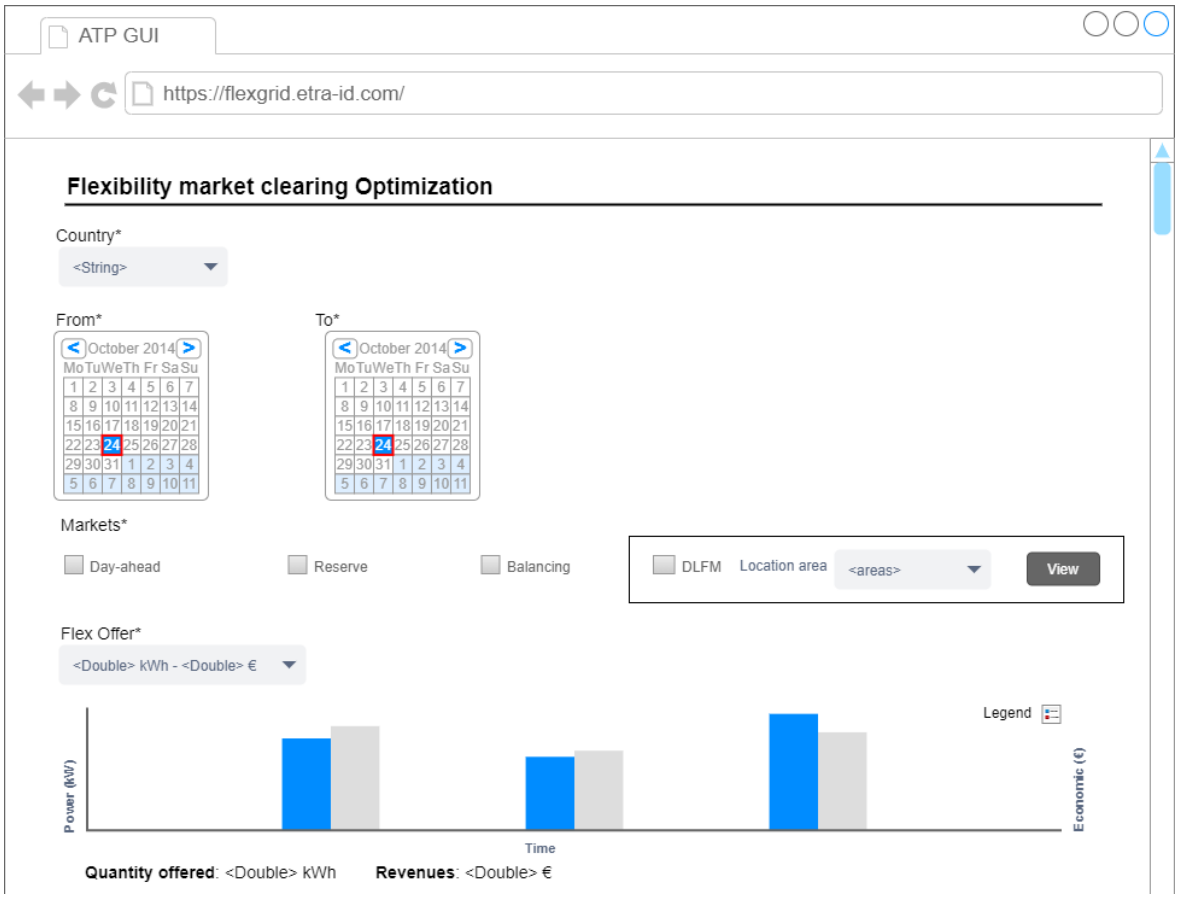


Figure 3: UCS1.1 - Market clearing algorithm configuration 1/3

Description: The user will configure the settings to run a flexibility market clearing operation following the next steps:

Firstly, it has to be selected (Figure 3):

- Select the country by clicking on *"Country"* button: This field is mandatory and will indicate the country/area for which this optimization is being performed. As the

clearing will take place for one specific DSO area not only the country is enough information.

- Select the dates for the optimization by clicking on "*From*, *To"* calendars: This field is mandatory and should include the hours for the optimisation.

For being able to run the algorithm:

- Indicate at least 1 *market* to be considered. It will be possible to select several of them by checking the checkbox.
- Two ways for the selection: by selecting some of the areas already defined and stored in the DB or by clicking on the "View" button (Figure 3).

Selection of the zone by "view" button:

- A map (Figure 4) will be presented with all the possible areas/zones contained inside the selected country.
- By selecting the area, it will be considered for running the flexibility market clearing algorithm to be run (and stored in the DB).

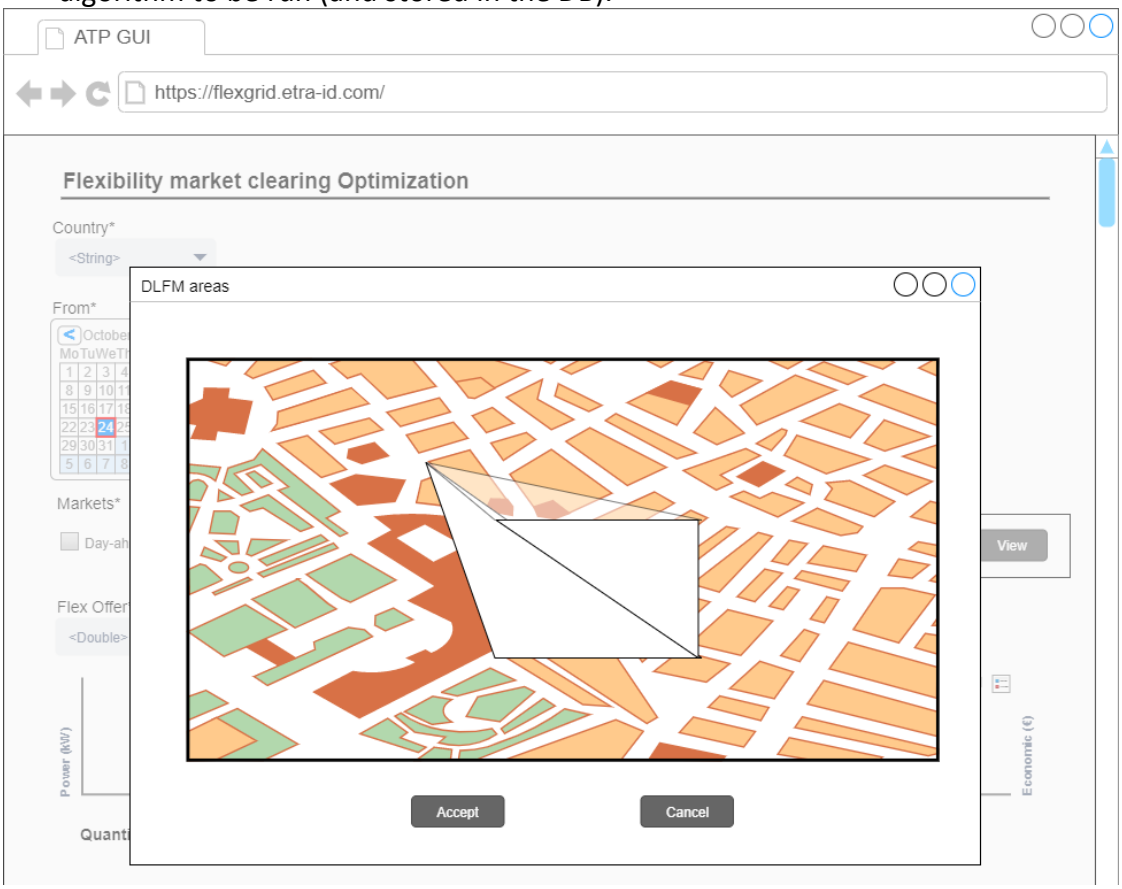


Figure 4: UCS1.1 - Market clearing algorithm configuration 2/3

Finally, by using the two last dropdowns (Figure 5) it can be selected:

- The type of clearing algorithm to be used: continuous, auction.
- The type of optimal power flow: Second order cone relaxation of AC-OPF, DC-OPF with approximations of losses and voltages.

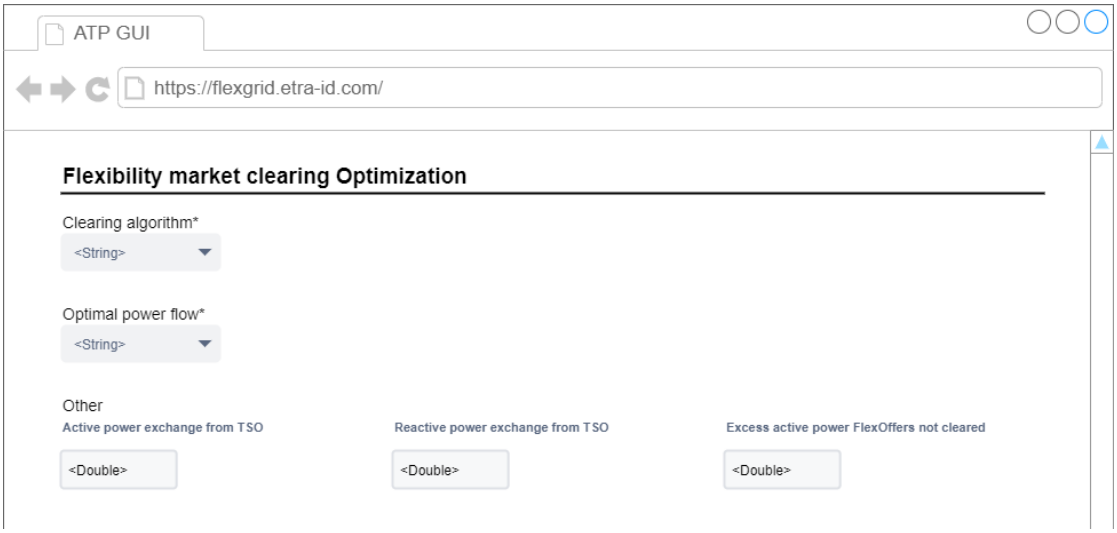


Figure 5: UCS1.1 - Market clearing algorithm configuration 3/3

On the same interface the cleaning and optimization algorithm is selected, it is possible to configurate the following optional fields

- Active power exchange from TSO
- Reactive power exchange from TSO
- Excess active power FlexOffers not cleared in the FM

Once all the settings have been configured, by clicking on the "Optimize" button the flexibility market clearing process will be triggered. Once it is finished, the results will be presented on the window *Flexibility market clearing results*.

Flexibility market clearing results

Users: DSO, ESP, FMO, Aggregator.
Description: It is assumed that all the mentioned users have access to the available data as all are participating in DLFM. The GUI allows two ways to access the results:

- By selecting one optimization from the list in the *Flexibility market clearing historical view*.
- From the *Flexibility market clearing results*, once the user triggers the optimization process and it has finished.

The information presented here contains (Figure 6):

- The configuration-specific data input inn this optimization.
- The country this optimization was performed for.
- If it exists, the active power exchange with TSO.
- If it exists, the reactive power exchange with TSO.
- A unique bar chart containing the results of the optimization. By clicking on a series name (the ones from both y-axis) on the legend, it will be possible to hide that series (or to put it back again):
- Information of each grid node per timestamp presented in a table format with relevant information (Figure 6):
    - Price of the active power
    - Voltage at the distribution node
- Information about the power flows across all distribution lines (**Error! Reference source not found.**)

By clicking on the “Save” button, the optimization results will be stored on the Central DB.

Main front-end view:

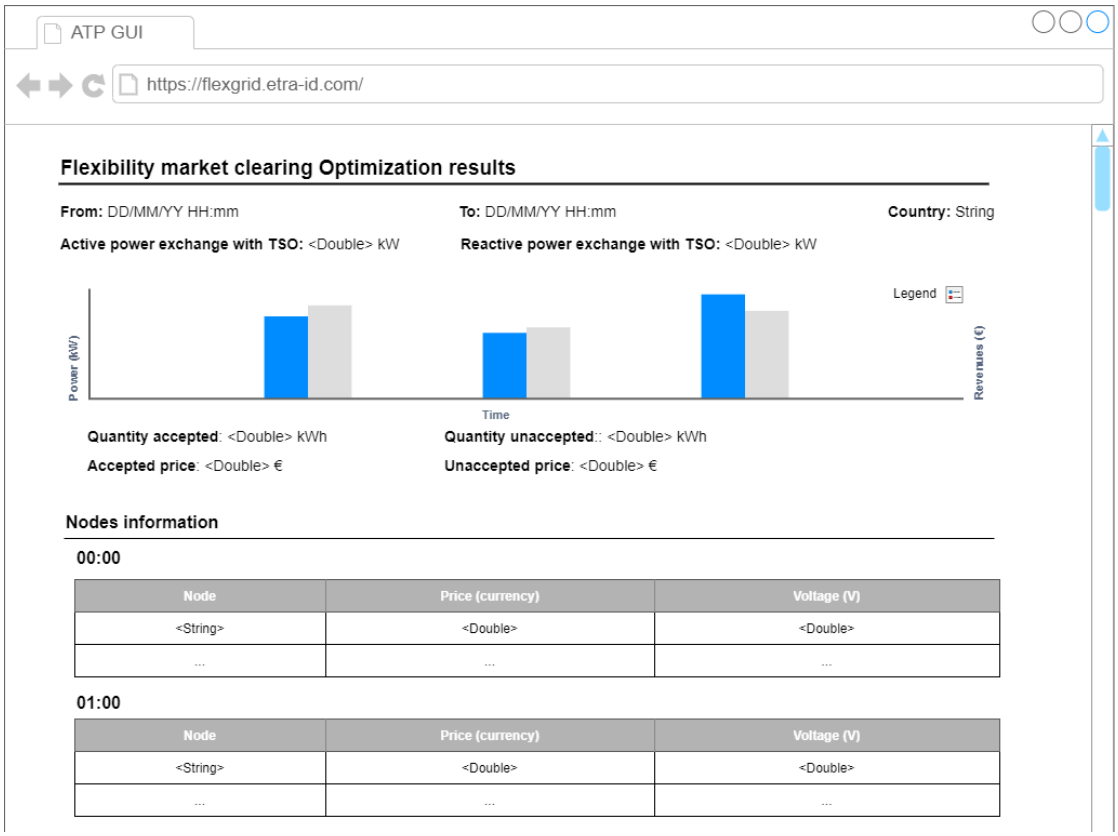


Figure 6: UCS1.1 - Market clearing results 1/2

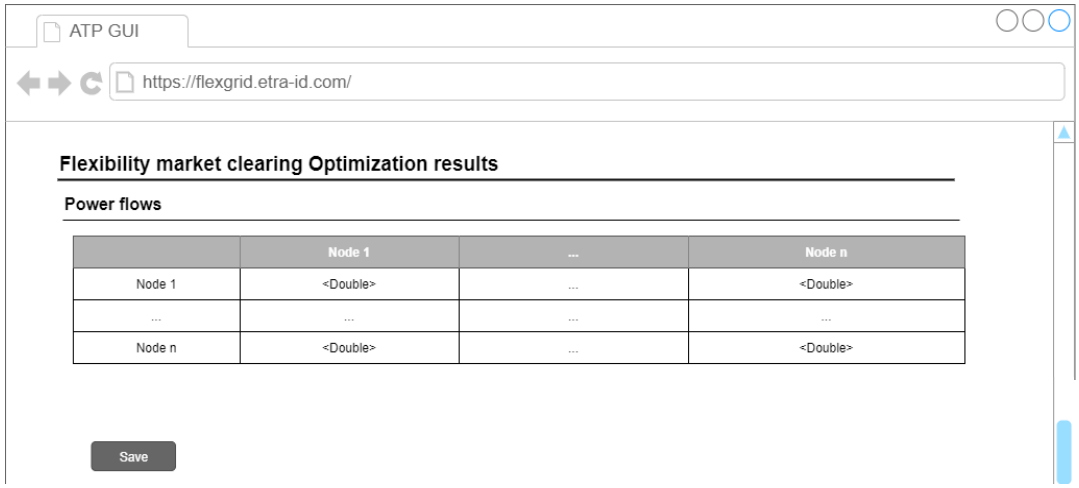


Figure 7 UCS1.1 - Market clearing results 2/2

3.3.2 UCS 1.2– DLFM clearing for the active power reserve (up/down) product

The mock-ups and descriptions are the same as in UCS 1.1– DLFM clearing for the active power (energy) product. The only difference is the name of this GUI section, which in this case is called “Market-based local congestion management”.

3.3.3 UCS 1.3– DLFM clearing for the reactive power reserve (up/down) product

The mock-ups and descriptions are the same as in UCS 1.1– DLFM clearing for the active power (energy) product. The only difference is the name of this GUI section being named "Market-based local voltage control".

3.3.4 UCS 2.1– Minimize ESP's Operational Expenditures (OPEX)

The UCS 2.1 will be shown considering the price (with revenues) and not considering the price (without revenues) depending on the type of user allowed to see the information as outlined in table 6. The front-end panels for "with revenues" and "without revenues" will be very similar with minimal differences explained in the subsections. The DSO and FMO will be only permitted to see the historical data of the configurations made without detailed information of prices as explained in the following.

OPEX optimization historical view (with revenues)

Users: ESP

Main front-end view:

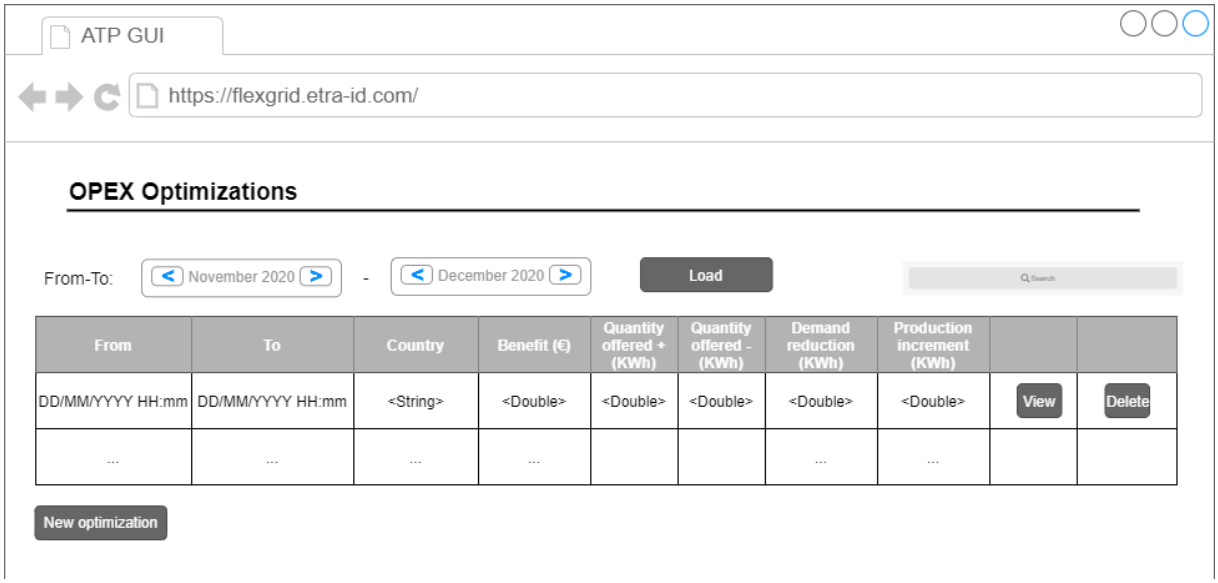


Figure 8: UCS2.1 – OPEX optimization historical view (with revenues)

Description:

All the stored optimizations for the actual month will be listed (Figure 8). It will also be possible to retrieve the stored optimizations for past operations (days, months…) by selecting a range of dates and clicking on the "Load" button.

It will also present basic information as:

- The duration of the optimization (**From, To**).
- The country the optimization was run for (**Country**).
- The total amount of benefit as a result of the optimization (**Benefit**).
- The total amount of energy resulting from the optimization both up and down (**Quantity offered +, Quantity offered -**).
- The total amount of demand reduced by all the activated flex assets on the optimization (**Demand reduction**).

- The total amount of production increase by all the activated flex assets in the optimization (**Production increment**).

Other functionalities available will be:

- Too sort the table according to one selected column.
- To search for a concrete value by using the text box above the table.

By selecting different button defined it will be possible to:

- By clicking on the "view" button the screen with the details of the selected optimization will be opened.
- By clicking on the "delete" button the selected optimization will be removed.
- By clicking on the "New optimization" button a window for configuring a new optimization will be opened.

OPEX optimization historical view (without revenues)

Users: DSO, FMO

Main front-end view:

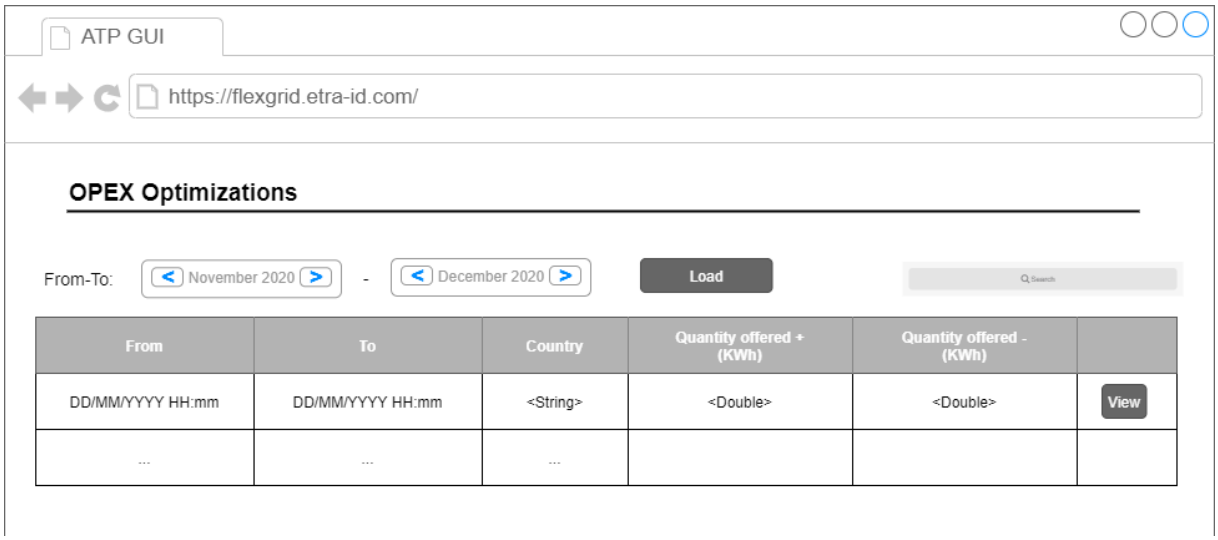


Figure 9: UCS2.1 – OPEX optimization historical view (without revenues)

Description: This screen (Figure 9) will be almost the same as the previous one (

The UCS 2.1 will be shown considering the price (with revenues) and not considering the price (without revenues) depending on the type of user allowed to see the information as outlined in table 6. The front-end panels for "with revenues" and "without revenues" will be very similar with minimal differences explained in the subsections. The DSO and FMO will be only permitted to see the historical data of the configurations made without detailed information of prices as explained in the following.

OPEX optimization historical view (with revenues)), with the following differences:

- No benefits data will be listed in the table.
- No demand reduction data will be listed.
- No production increase data will be listed.
- The "New optimization" button is not available here.
- The "Delete" button is not available here.

This view for the DSO and the FMO will be only informative view to better understand and have a control of the different bids made by the ESP.

OPEX optimization configuration

For the "optimization configuration" no price differentiation is available because only the ESP user will have the opportunity to see and modify the information defined in this front-end.
Users: ESP
Description: The user will configure the settings for running a new OPEX optimization. The ESP has to follow the steps described below to correctly introduce the required data to run the algorithm and to obtain an optimized result.

The first step is the selection of the country for which this optimization is being performed and the dates of the period of interest (Figure 10)

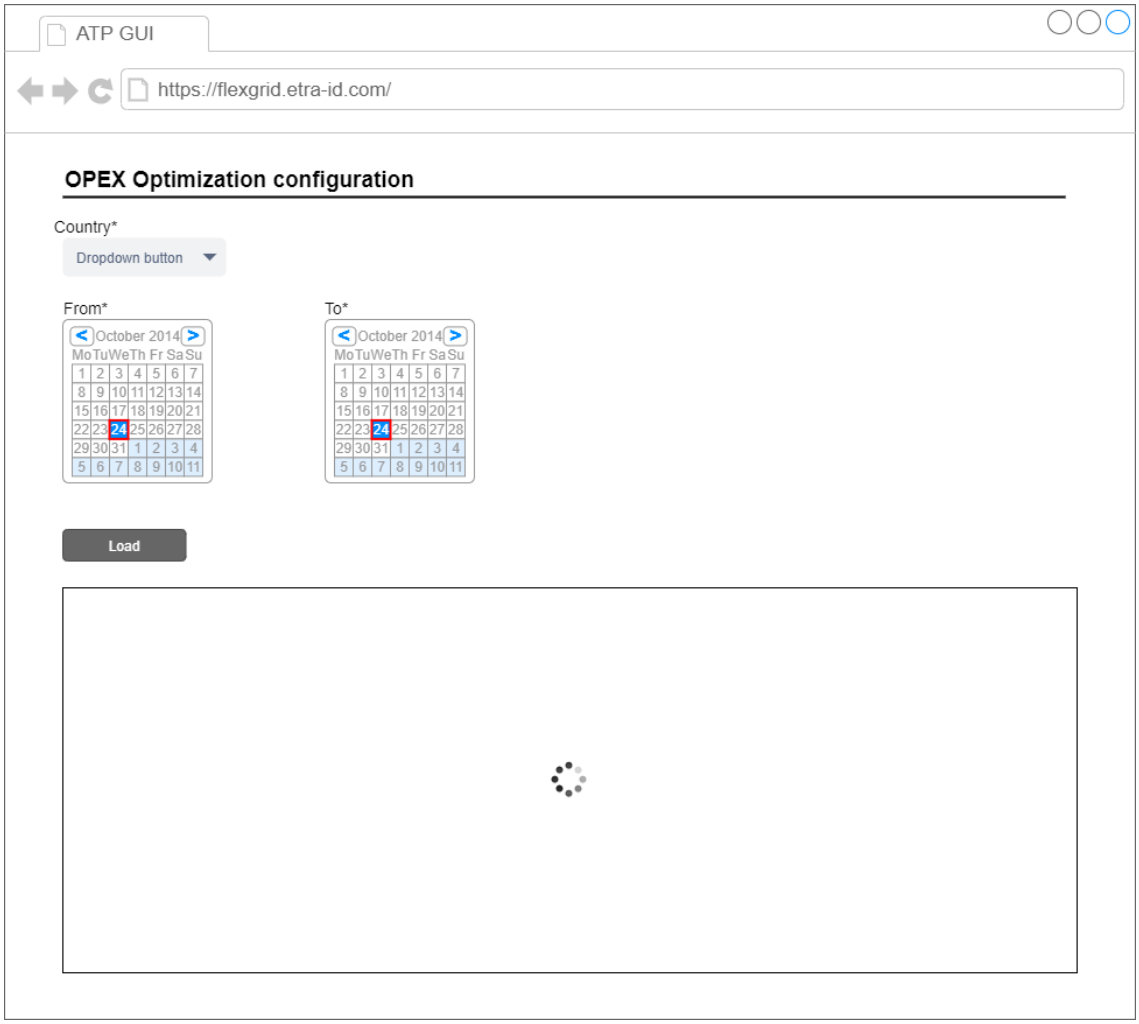


Figure 10: UCS2.1 – OPEX optimization configuration 1/8

The aforementioned field are mandatory for every configuration. Once these fields are complete, by clicking on the "Load" button (Figure 10) the *day-ahead energy schedules* and *flexibility offers* will be retrieved for the selected days. A schedule and offer will be presented in the two charts (Figure 11).

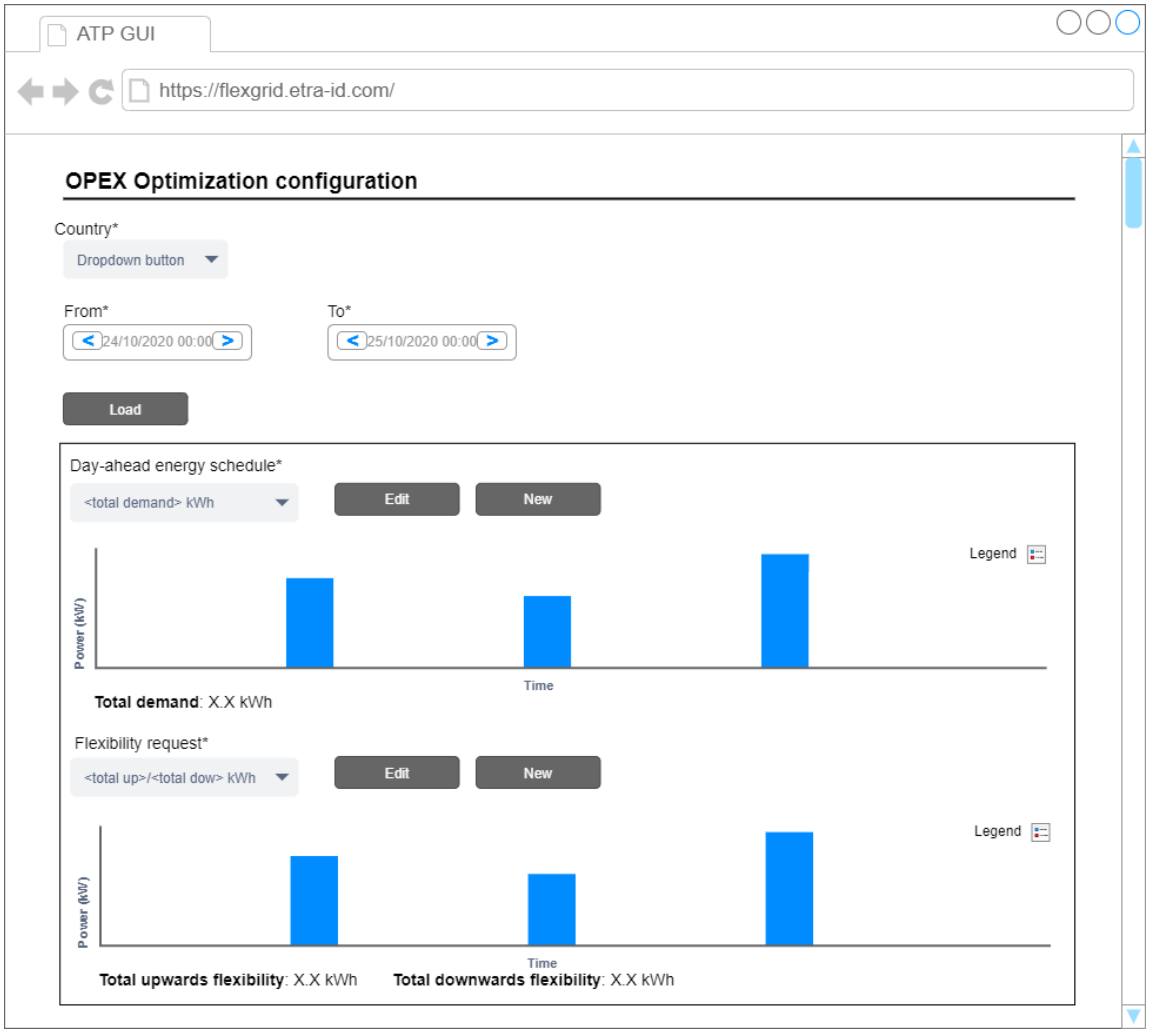


Figure 11: UCS2.1 – OPEX optimization configuration 2/8

The data to be visualized in the charts can be selected from the dropdowns, in addition the information on both graphs can be manually edited (by clicking on the "Edit" button) or created from scratch (by clicking on the "New" button). This function is available in case the user wants to run the algorithm with different data from the one retrieved from the system initially. If no *day-ahead energy schedules* and/or *flexibility offers* are found for these days, a warning message will be presented as both are mandatory information to run the algorithm.

Day-ahead energy schedules data edition (Figure 12):

- Table for customizing (edit, include, remove…) demand values
- By clicking on the "Accept" button, a new schedule will be included on the corresponding dropdown menu. With this functionality the user would check other schedules before deciding which one will be used for the optimization.
- By clicking on the "Cancel" button this window will be closed, and the schedule will remain the same one before the data edition.

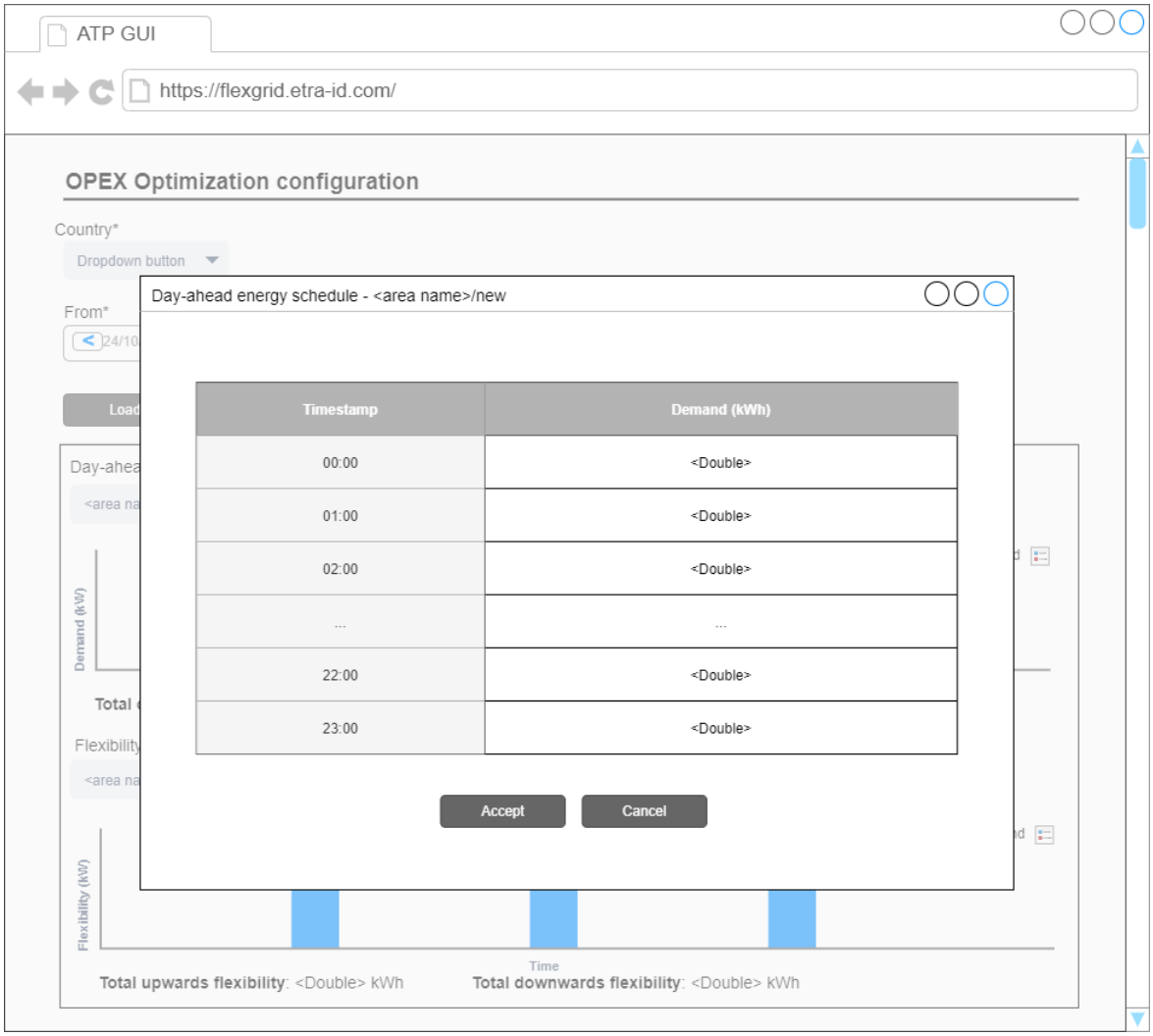


Figure 12: UCS2.1 – OPEX optimization configuration 3/8

Flexibility offers data edition (Figure 13):

- Table for customizing the flexibility in both regulation directions.
- By clicking on the “Accept” button, a new offer will be included on the corresponding dropdown menu just in case the user wants to check some other offers before deciding which one will be used for the optimization. Also, this custom offer will be presented on the chart.
- By clicking on the “Cancel” button this window will be closed, and the schedule will remain the same one before the data edition.

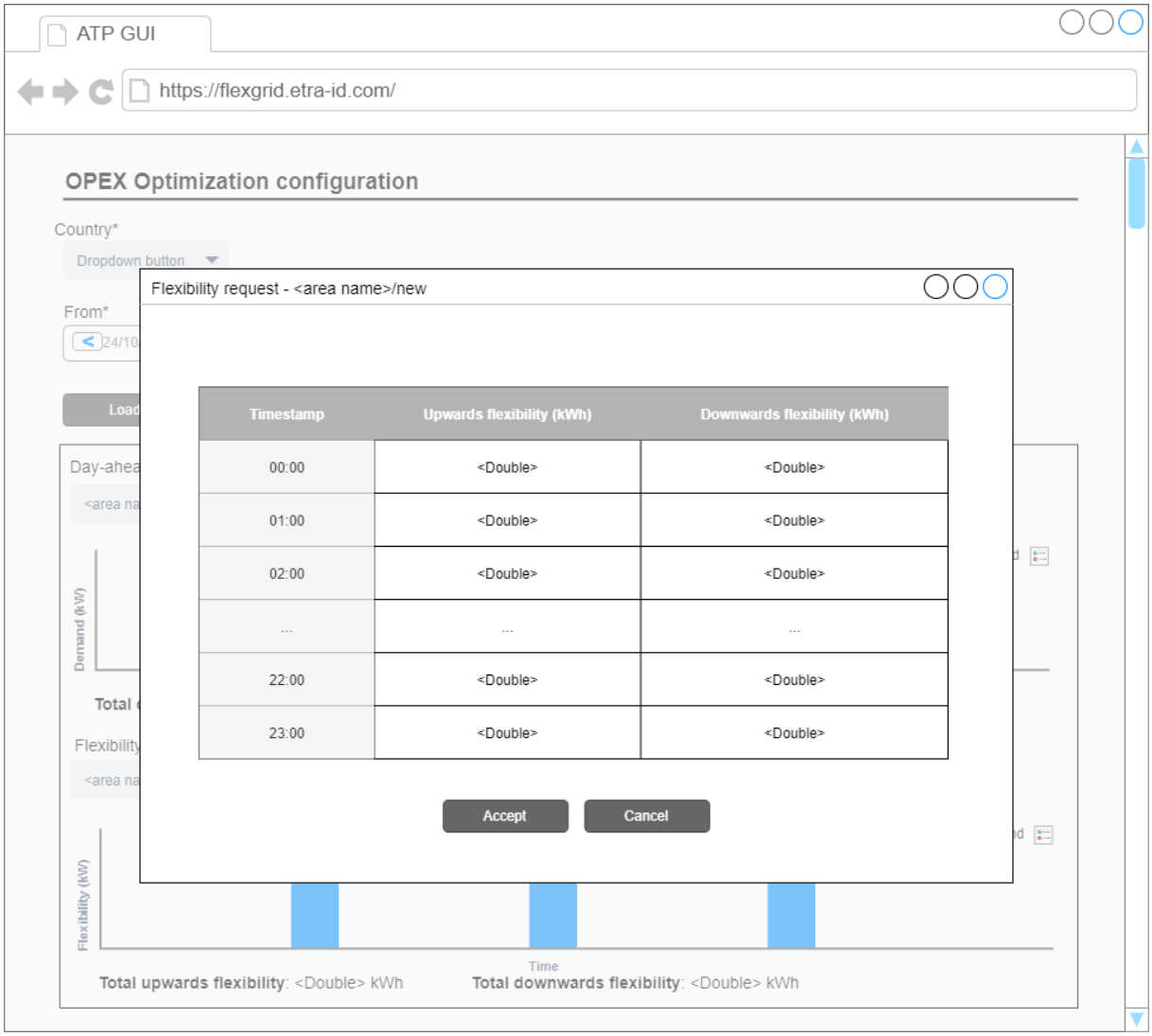


Figure 13: UCS2.1 – OPEX optimization configuration 4/8

Once the schedules are ready, the next field to add is the *FlexAssets* to be involved in the optimization. For running the OPEX optimization algorithm it is mandatory to select at least one asset. It will be possible to select more than one asset. The selectable assets belong only to the selected DSO zones in the first dropdown.

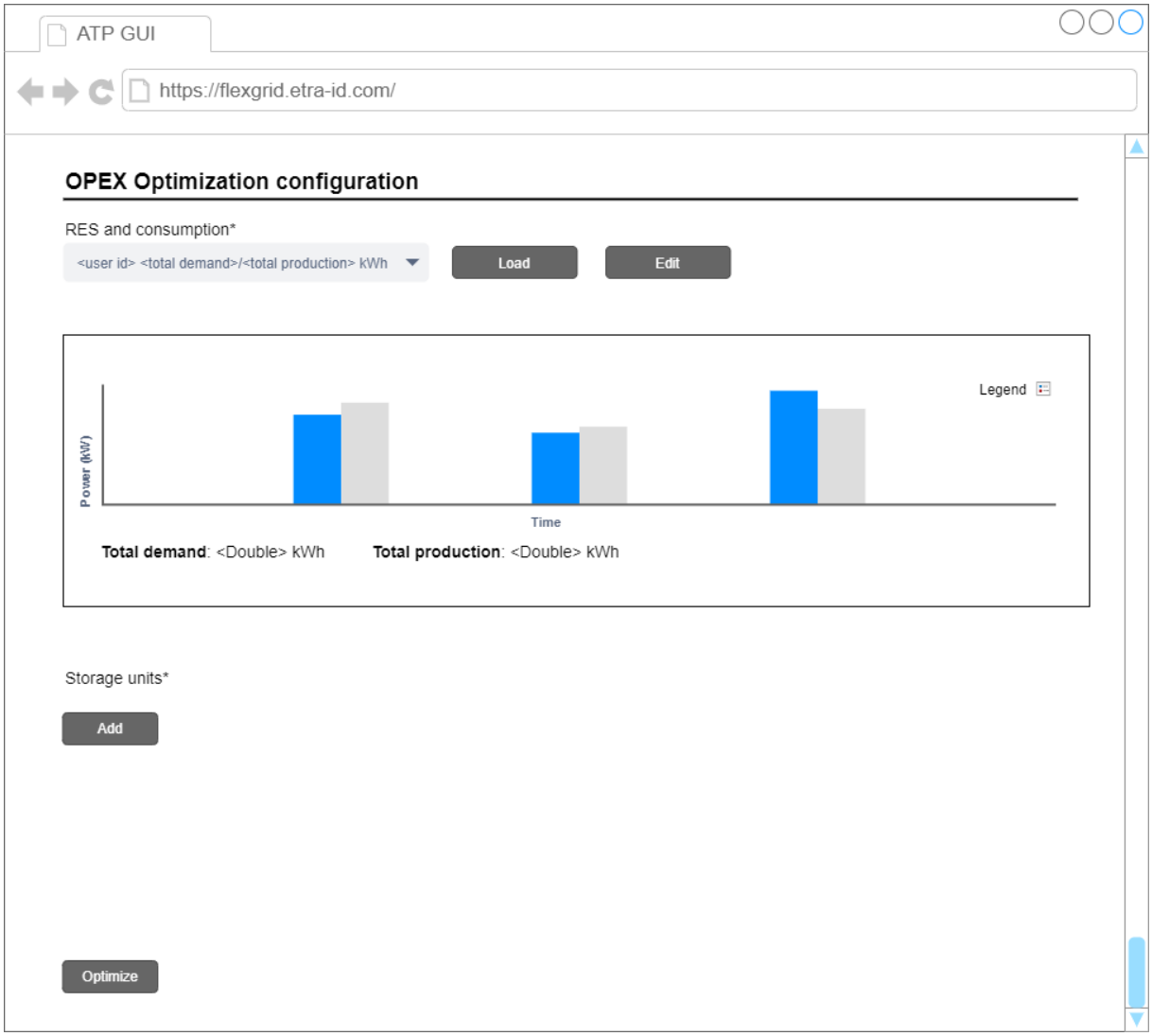


Figure 14: UCS2.1 – OPEX optimization configuration 5/8

Once the selection has been finished, by clicking on the "Load" button it will be retrieved the aggregated demand and/or production from the selected users/assets (Figure 14). When this aggregated data has been retrieved and presented, the "Edit" button will be available for modifying this data at user level (Figure 15).

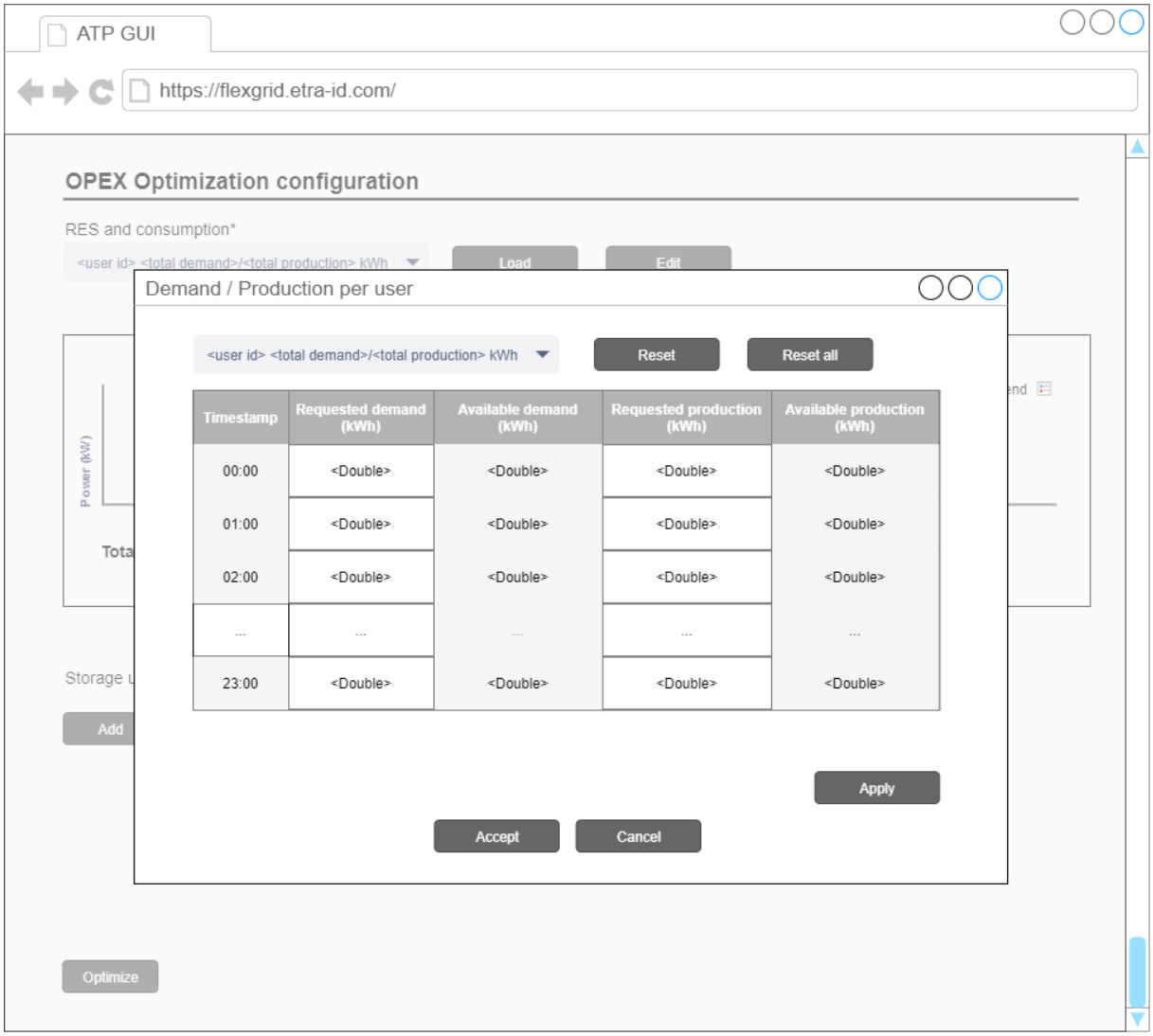


Figure 15: UCS2.1 – OPEX optimization configuration 6/8

Following a brief explanation on how each button of this view works

- Via the "Apply" button the modified values are stored locally added for this user/asset.
- The "Reset" button they will be set again to its default values (the ones retrieved from the system). If the "Reset all" button is clicked, this reset process will be applied to all the retrieved users.
- The "Accept" button allows all these changes will be considered and the aggregated chart will be updated (and also the aggregated data to be sent as an input to the optimization process).
- The "Cancel" button allows to reset the values modify to the last values "accepted"

Finally, the *Storage Units* to be considered in the optimization process have to be selected.

- The “Add” button (Figure 14) will only be available after selecting one or more users. All storage units will be presented as in Figure 16, as a map, for the selected country.

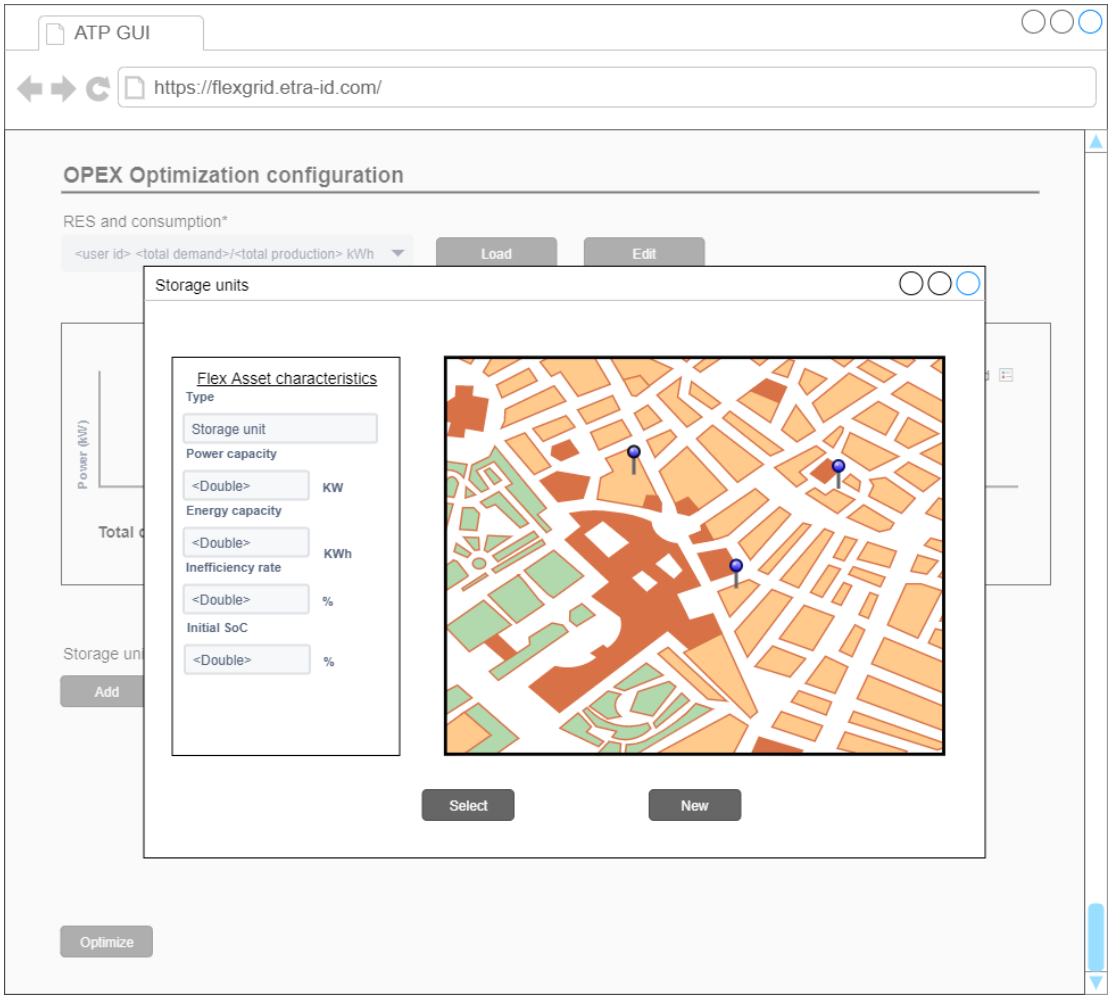


Figure 16: UCS2.1 – OPEX optimization configuration 7/8

It is possible to include new storage units from this map pop-up window by clicking on the “New” button. The new assets will only be considered for optimization purposes, they won’t be stored on the system. It will also be possible to remove the new storage units by clicking the “Delete” (Figure 17: UCS2.1 – OPEX optimization configuration 8/Figure 17).

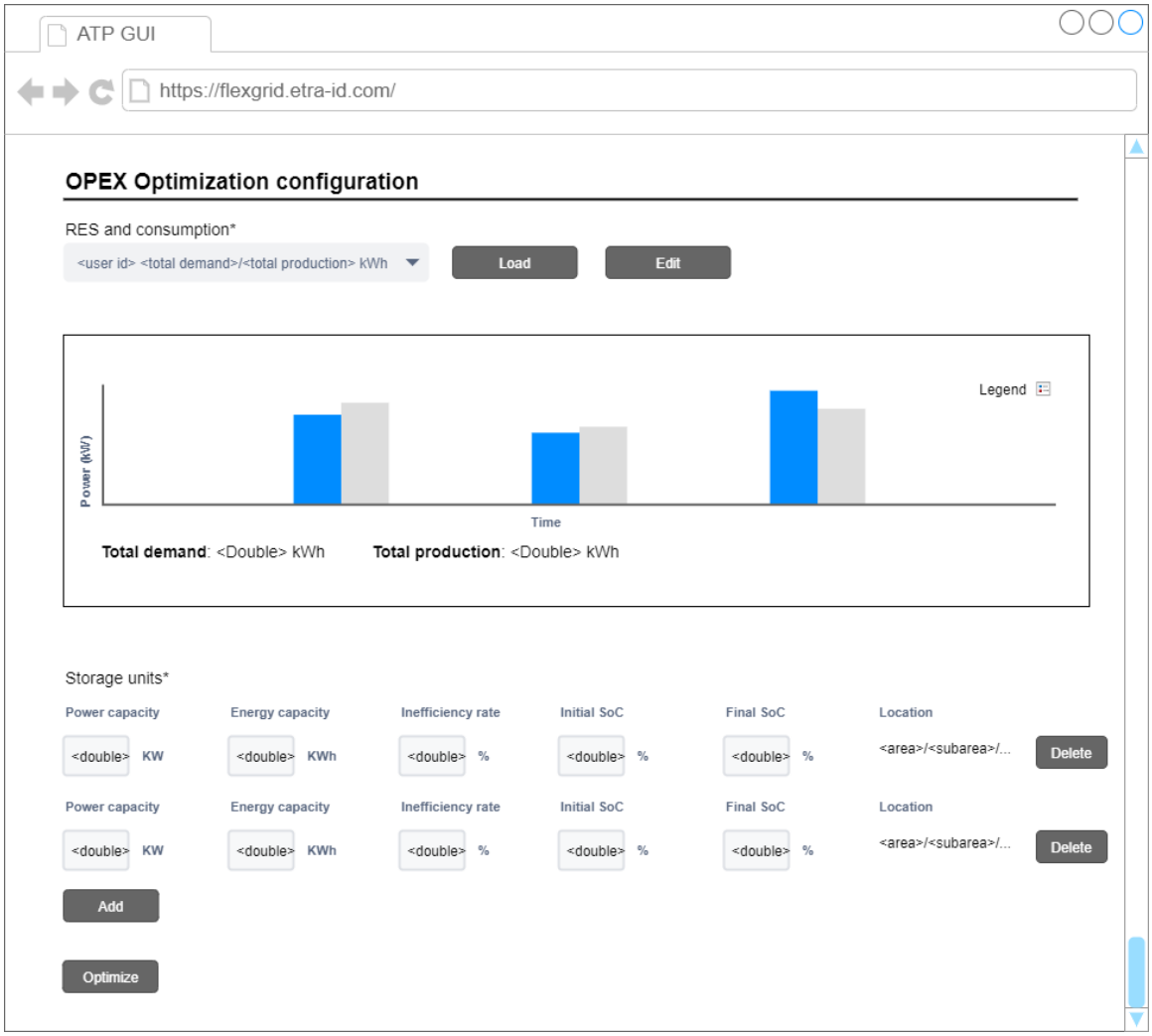


Figure 17: UCS2.1 – OPEX optimization configuration 8/8

Once all the settings have been configured, by clicking on the "Optimize" button the OPEX optimization process will be triggered. Once it has finished, the results will be presented on the screen

OPEX optimization results (with revenues).

OPEX optimization results (with revenues)

Users: ESP
Main front-end view:

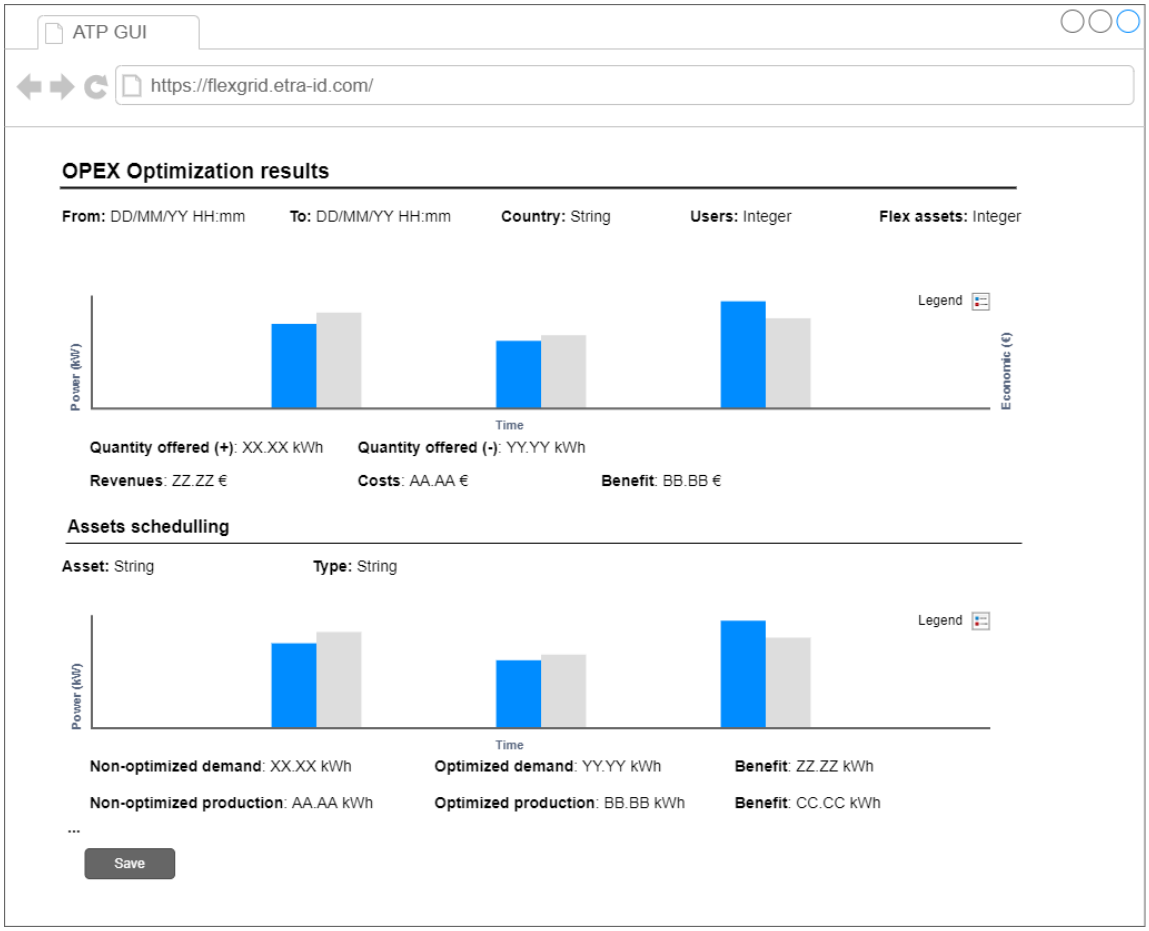


Figure 18: UCS2.1 – OPEX optimization results (with revenues)

Description: There are two possible ways the ESP user can access to this screen (Figure 18):

- By selecting one optimization from the list in the
- *The UCS 2.1 will be shown* considering the price (with revenues) and not considering the price (without revenues) depending on the type of user allowed to see the information as outlined in table 6. The front-end panels for "with revenues" and "without revenues" will be very similar with minimal differences explained in the subsections. The DSO and FMO will be only permitted to see the historical data of the configurations made without detailed information of prices as explained in the following.
- OPEX optimization historical view (with revenues)
- From the *OPEX optimization configuration* view, once the user triggers the optimization process and it has finished

The information presented here contains:

- The dates, country, number of users and number of Flex Assets considered for the optimization.
- A bar chart containing the general results of the optimization.
- Economical information:
  - Revenues
  - Costs
  - Benefit, which is the difference between costs and revenues
- Aggregate data for the entire period below the chart

By clicking on the “Save” button, the optimization results will be stored on the Central DB.

OPEX optimization results (without revenues)

The DSO and FMO users should have access to the historical results (i.e. dispatch decisions of the ESP). We call this view "without revenues", because DSO and FMO users should not be able to see the ESP's revenues, but only the ESP's dispatch decisions. This is needed for the settlement phase (i.e. how the ESP is paid through the FMO) and also for the DSO to be able to verify ex-post that the ESP is not manipulating the market!

Users: DSO/FMO
Main front-end view:

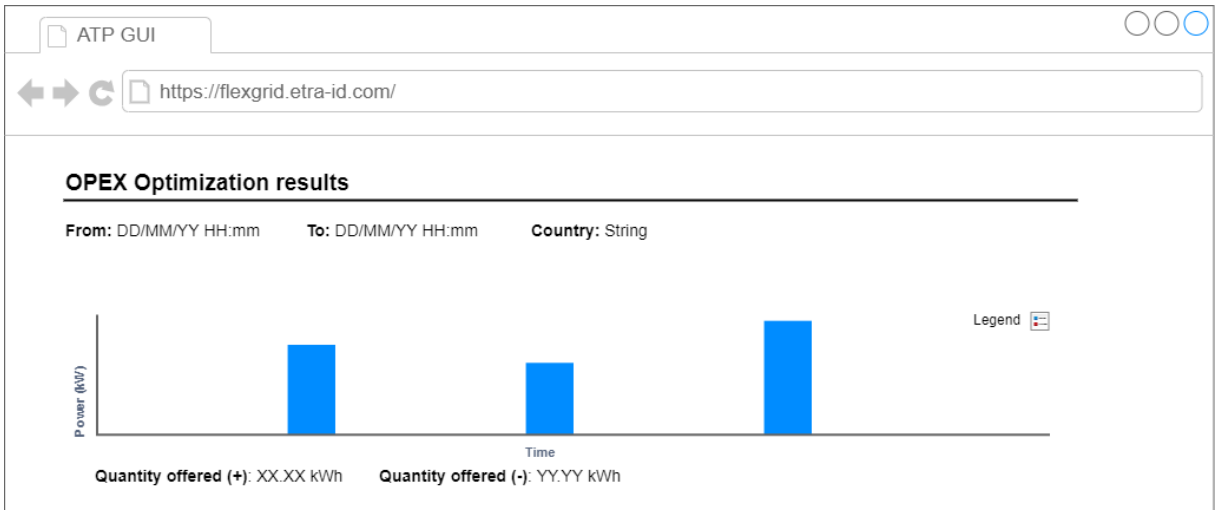


Figure 19: UCS2.1 – OPEX optimization results (without revenues)

Description: There is only one possible way the DSO/FMO user can access to this screen:

- By selecting one optimization from the list in the *OPEX optimization historical view (without revenues)*

This screen will be almost the same as the previous one, with the following differences:

- No information about the number of users considered inn the optimization is presented.
- No information about the number of flexibility assets considered in the optimization is presented.
- No economic information is presented.
- No assets scheduling is presented.
- No “Save” button is presented.

3.3.5 UCS 2.2– Minimize ESP’s Capital Expenditures (CAPEX)

CAPEX optimization historical view

Users: ESP
Main front-end view:

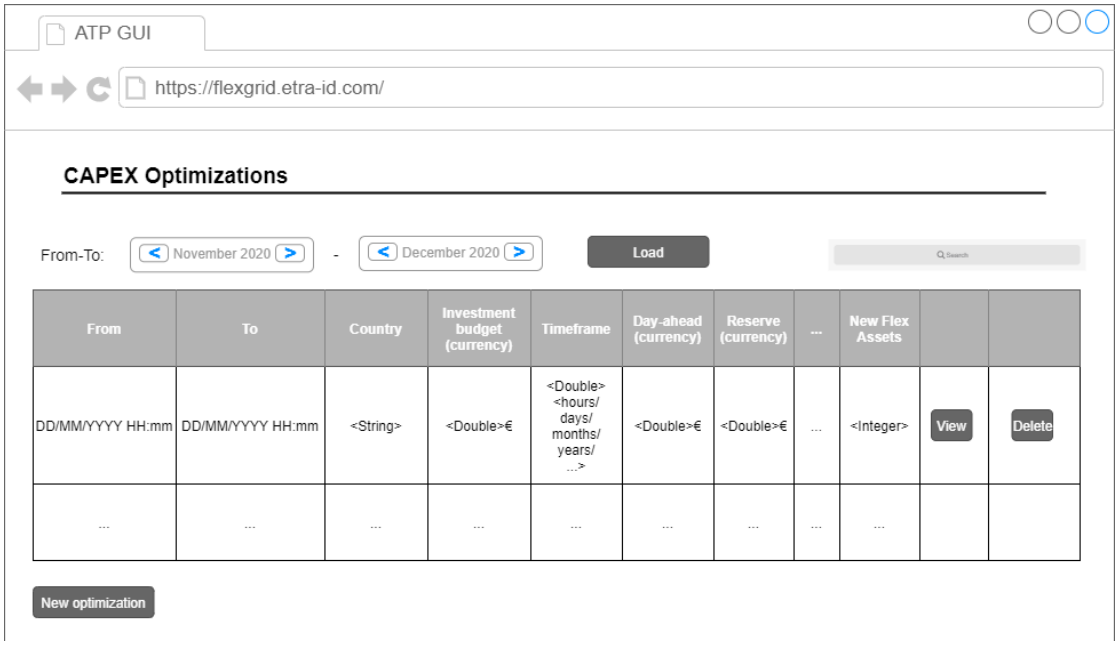


Figure 20: UCS2.1 – CAPEX optimization historical view

Description: In this view all the stored optimizations made for the actual month will be listed (Figure 20). It will also be possible to retrieve the stored optimizations for past months optimization by selecting a range of dates and clicking on the "Load" button.
For each optimization basic information is included:

- The duration of the optimization (**From**, **To**).
- The country the optimization was run for (**Country**).
- The necessary cost of the investment on the purchase of new FlexAssets in the future (**Investment budget**) - Only for the ESP -.
- The assumed timeframe for the budget investment (**Timeframe**)
- The total amount of revenues as a result of the optimization for each possible market the optimization was carried out. (**Day-ahead, Reserve, Balancing, DLFM**) - Only for the ESP-. If one market was not optimized, no information will appear.
- The minimum number of new flex assets to install (**New Flex Assets**)

It will be possible to sort the table according to one selected column. It will be possible also to search for a concrete value by using the text box above the table.
By the different buttons presented it is possible to manage the information available:

- By clicking on the "view" button it will be opened the screen with the details of the selected optimization.
- By clicking on the "delete" button it will be removed the selected optimization.
- By clicking on the "New optimization" button it will be opened the screen for configuring a new optimization.

CAPEX optimization configuration

Users: ESP
Description: The user will configure the settings for running a new CAPEX optimization. The ESP has to follow the steps described below to correctly introduce the data required to run the algorithm and obtain an optimized result.

The first step is the selection of the country for the optimization, the "FlexAssets" to be involved in the optimization and the dates for the optimization. All the information can be added in the view showed in Figure 21.

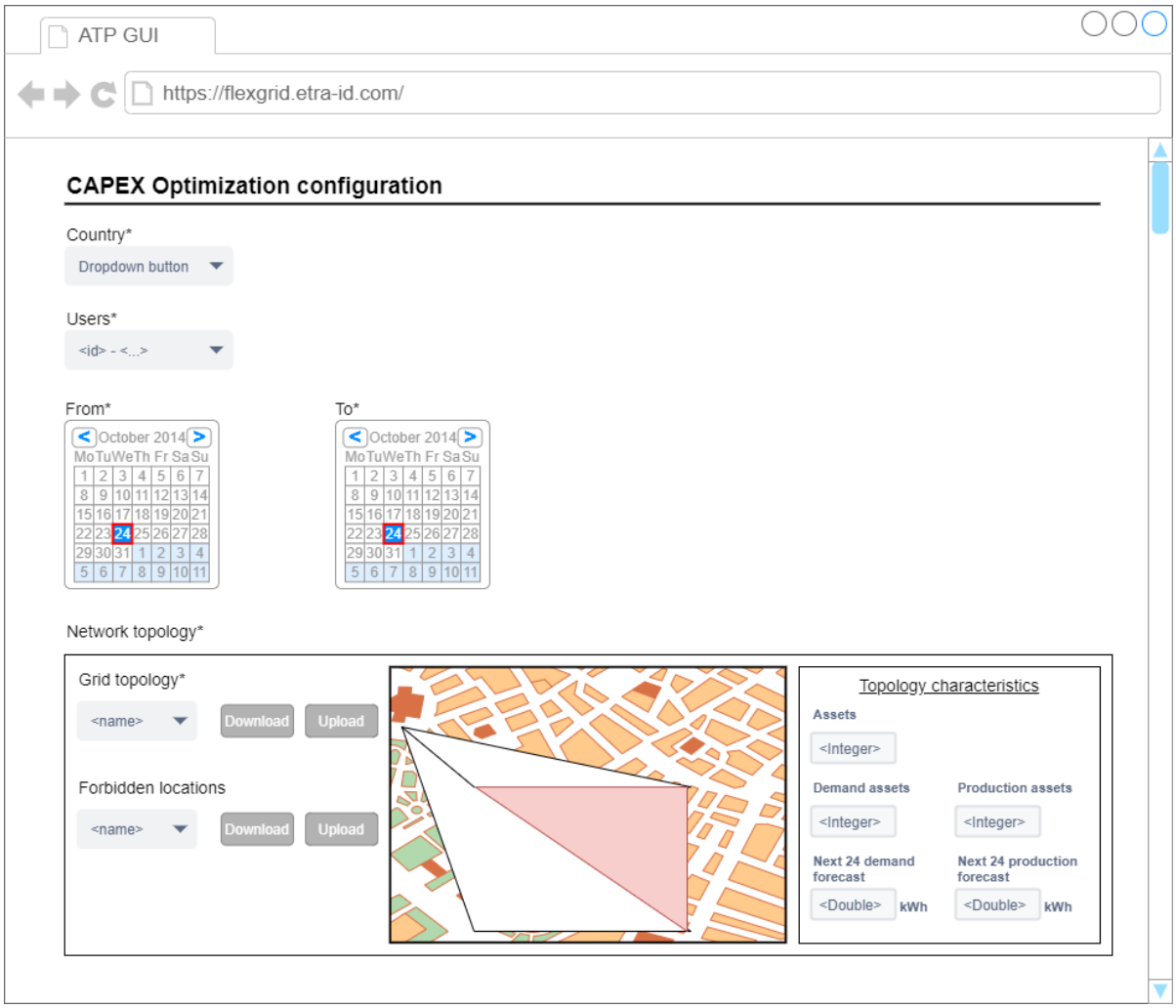


Figure 21: UCS2.2 – CAPEX optimization configuration 1/4

After filling this basic mandatory information, the ESP should define the network topology (Figure 21):

- *Grid topology*: The area to be considered for optimization. Only predefined areas within the selected country will be presented here and it is mandatory to introduce the information for the correct operation of the algorithm.
- *Forbidden locations*: If any have been restricted to act the ESP should indicate it. This can happen because for example the sub-areas belong to other market actor. This information is

The user can manually define more areas by clicking on "custom" button following the next steps for both "Grid topology" or "Forbidden locations":

- Download: A .csv template would be available (as in NODES platform) for including the coordinates of the new custom area.
- Upload: Once the information has been filled, it will be possible to upload the .csv back on the system for considering the new area in the optimization.

Once the network topology has been defined, it will be possible to select the Storage Units to be considered by the optimization by clicking on the “Add” button (Figure 22).

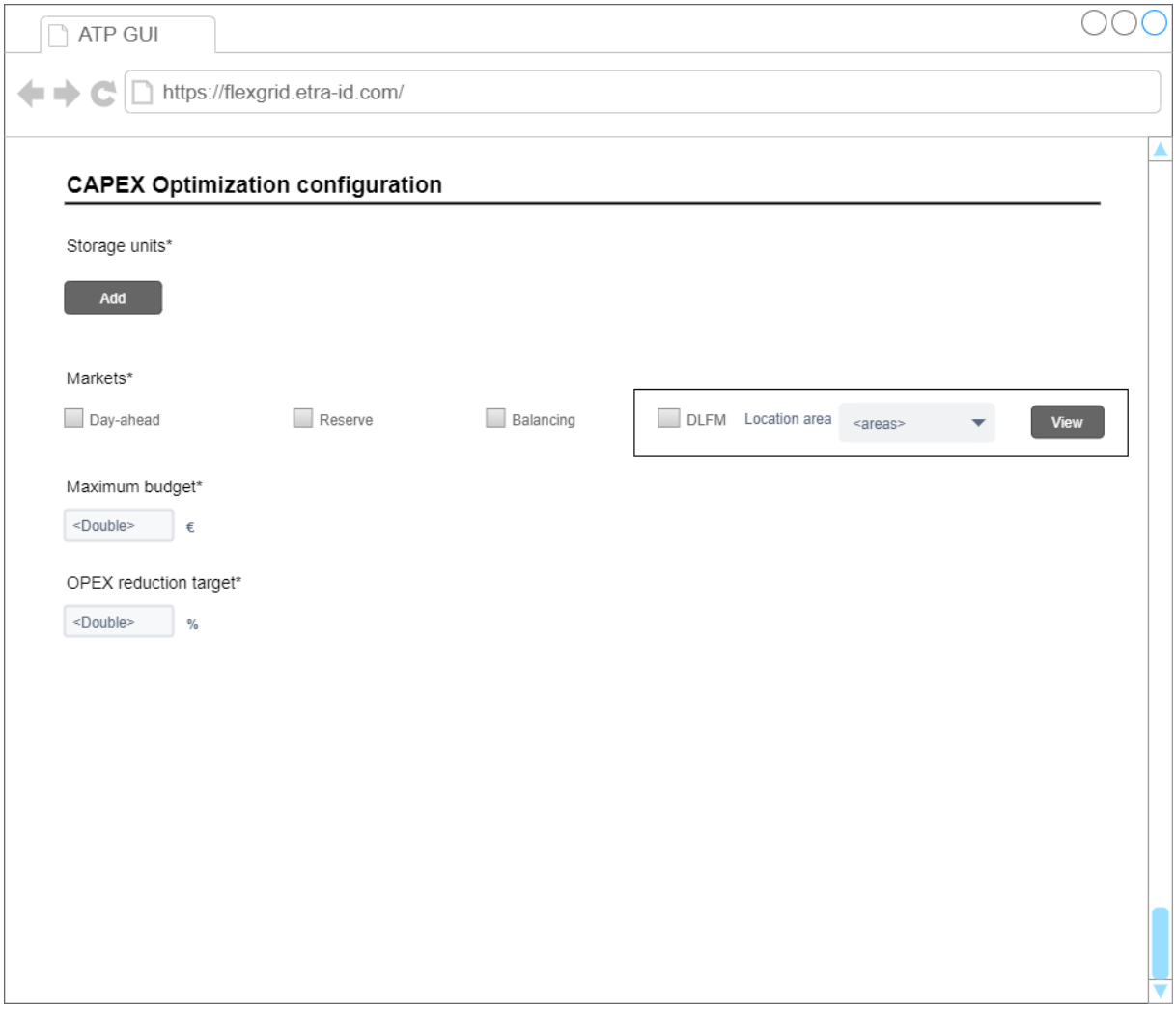


Figure 22: UCS2.2 – CAPEX optimization configuration 2/4

A map with all the storage units available within the selected areas (Figure 23). When the storage units are selected, by clicking on the “Select” button this window will be closed, and the information of the units will be presented. I will be possible to edit the retrieved data only for current optimization (any update here won’t be stored back in the system).

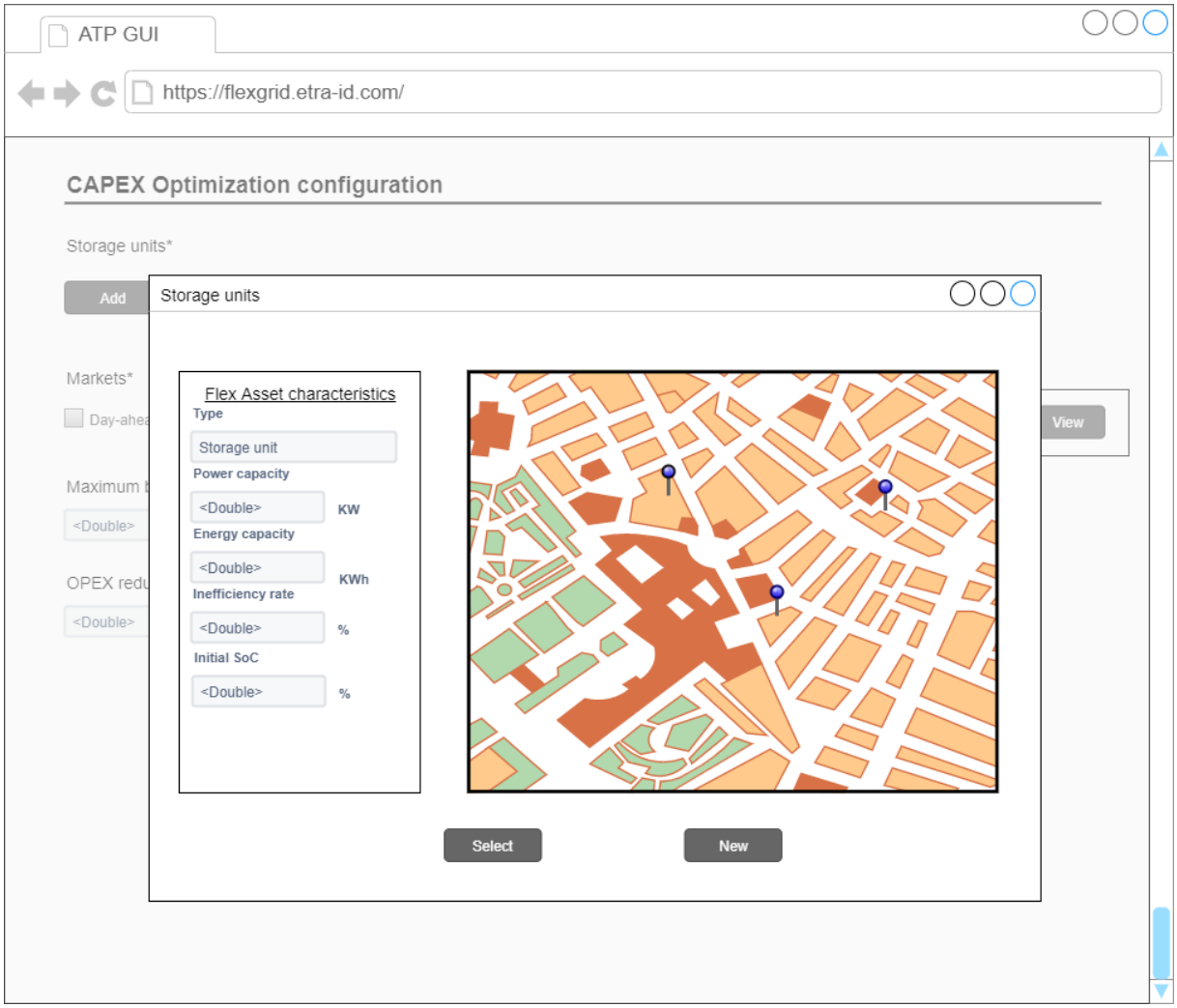


Figure 23: UCS2.2 – CAPEX optimization configuration 3/4

At the edition view (Figure 24) in addition to change the storage information, it will also be possible to include new storage units by clicking on the "New" button and introducing the data manually. In this view the ESP also is allowed to remove storage units by clicking the "Delete" button.

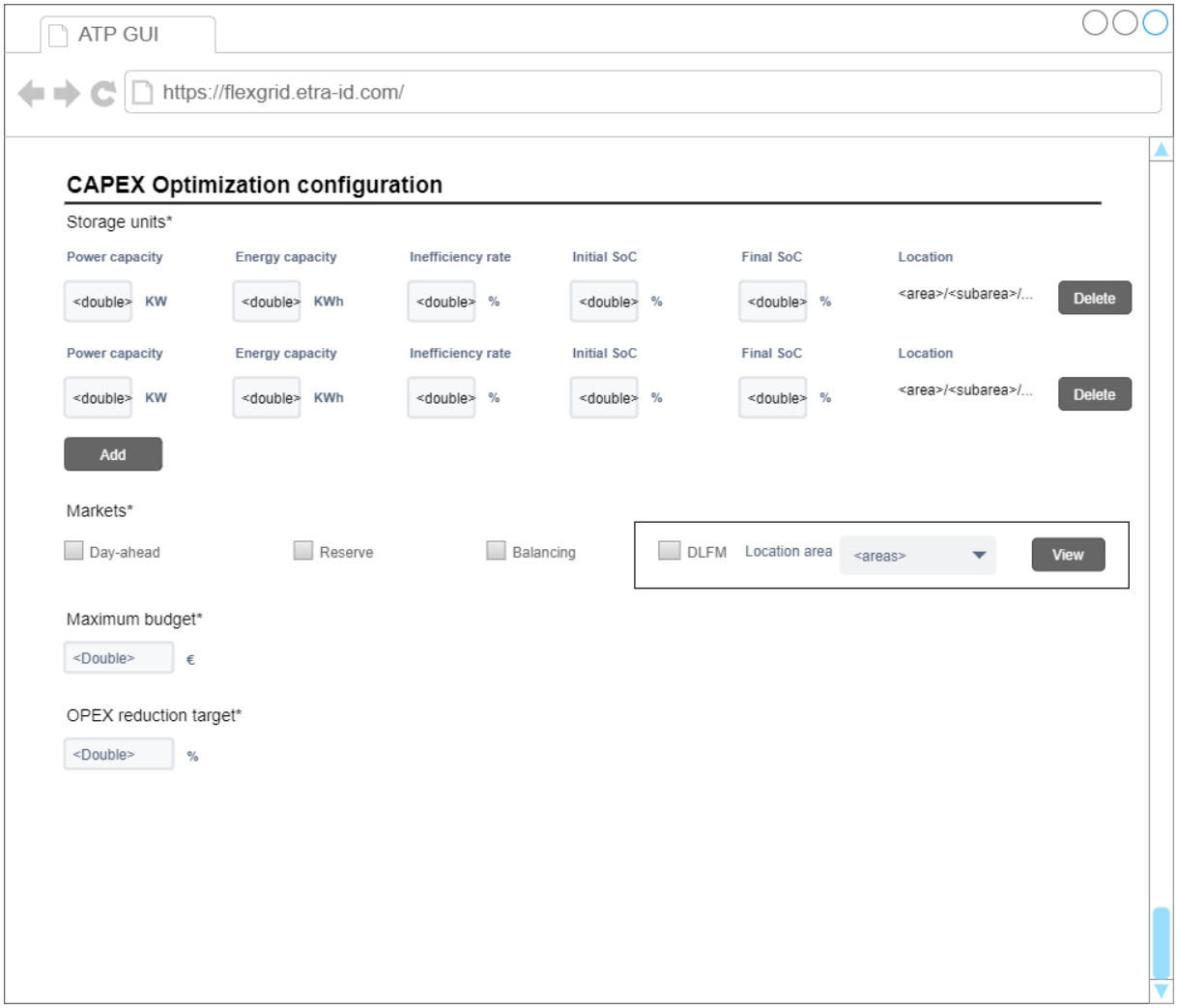


Figure 24: UCS2.2 – CAPEX optimization configuration 4/4

Finally in order to run the algorithm:

- One *market* must be selected. It will be possible to select some of them by checking the checkboxes.
- The *Maximum budget* and *OPEX reduction target* fields have to be filled in. Both are mandatory. This field implies the budget that the ESP is willing to invest on new FlexAssets.

Once all the settings have been configured, by clicking on the "Optimize" button the OPEX optimization process will be triggered. Once it is finished, the results will be presented on the screen CAPEX optimization results.

CAPEX optimization results

Users: ESP

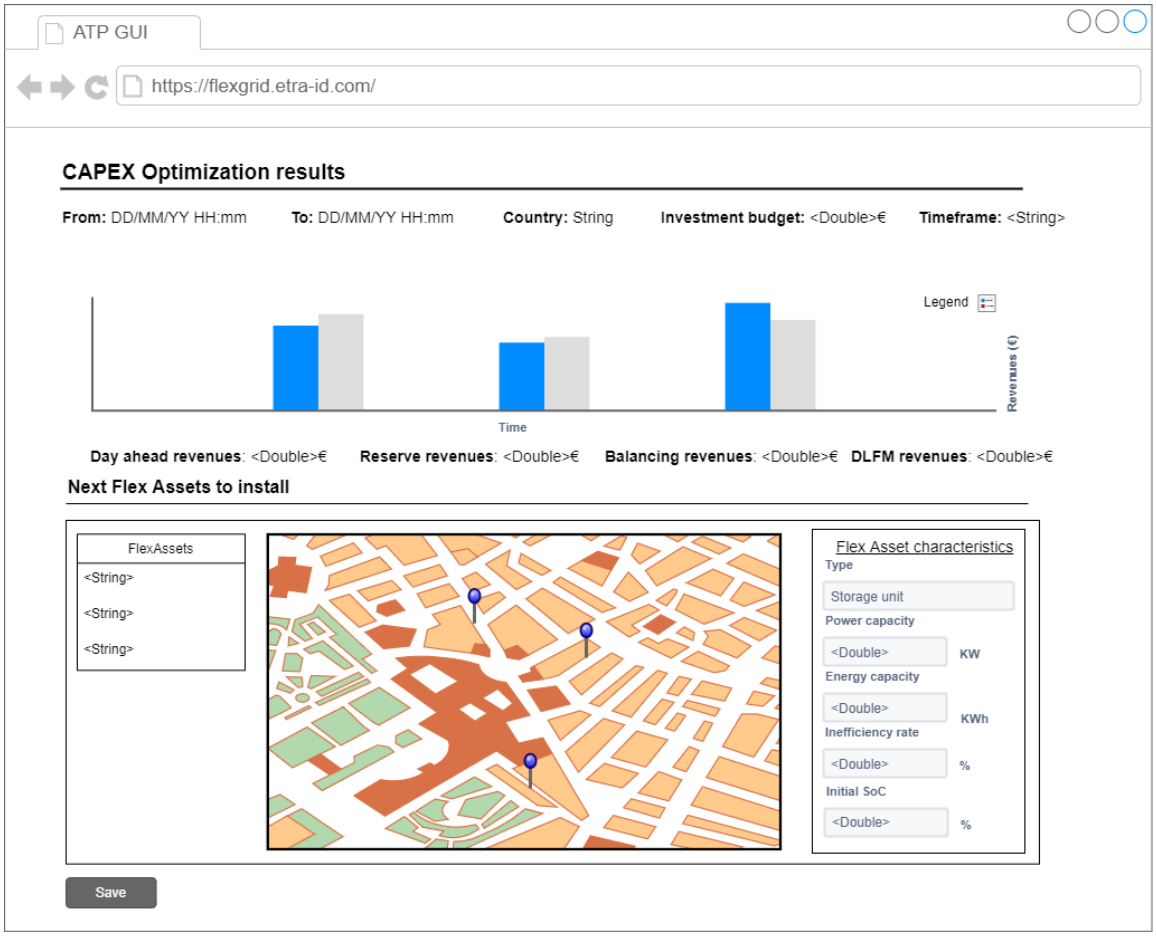


Figure 25: UCS2.2 – CAPEX optimization results

Description: There are two possible ways the ESP user can access this screen:

- By selecting one optimization from the list in the *CAPEX optimization historical view*.
- From the *CAPEX optimization configuration* view once the user triggers the optimization process when it is finished.

The information presented here contains:

- The dates selected for this optimization.
- The country this optimization was performed for.
- The investment budget for this optimization.
- The assumed timeframe for this investment.
- A unique bar chart containing the results from the selected markets when the optimization was configured.
- A map presenting the new flexibility assets that, according to the optimization, could be used for the optimization for obtaining better results.

To save the optimization on the Central DB the ESP should click on the “Save” button.

3.3.6 UCS 2.3– Maximize ESP’s stacked revenues

Profits optimization historical view (with revenues)

Users: ESP

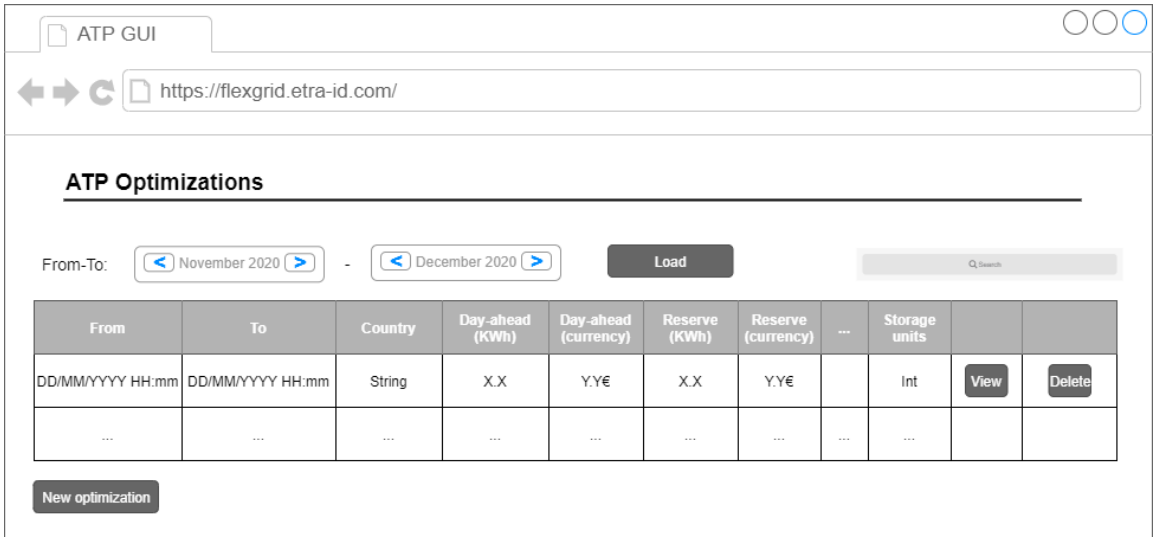


Figure 26: UCS2.3 – Profits optimization historical view (with revenues)

Description: By default, it will be listed all the stored optimizations for the actual month (Figure 26). It will also be possible to retrieve the stored optimizations for periods in the past by selecting a range of dates (above the table it could also be selected the day of the months) and clicking on the "Load" button.
It will be presented basic information including:

- The duration of the optimization (**From, To**)
- The country the optimization was run for (**Country**)
- The total amount of energy (measured in kWh) as result of the optimization for each market (**Day-ahead, Reserve, Balancing, DLFM**). If a marked was not considered in the optimization, no value will appear here
- The total amount of revenues (measured in the currency of the country) as result of the optimization for each market (**Day-ahead, Reserve, Balancing, DLFM**).). If a marked was not considered in the optimization, no value will appear here (the currency symbol will appear on each cell with values)
- The number of storage units configured (**Storage units**)

It will be possible to sort the table according to one selected column. It will be possible also to search for a concrete value by using the text box above the table.
By the different buttons presented is possible to manage the information available:

- By clicking on the "view" button it will be opened the screen with the details of the selected optimization.
- By clicking on the "delete" button it will be removed the selected optimization.
- By clicking on the "New optimization" button it will be opened the screen for configuring a new optimization.

Profits optimization historical view (without revenues)

Users: DSO, FMO
Description: This screen (Figure 27) will be almost the same as the previous one, with the following differences:

- On the table it won't be listed the revenues data
- The "New optimization" button is not available here
- The "delete" button is not available here

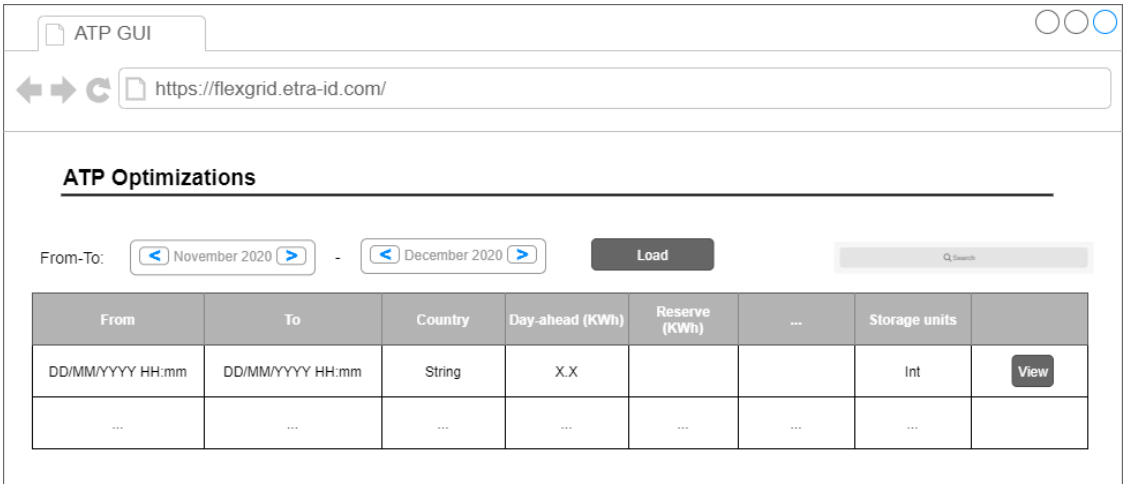


Figure 27: UCS2.3 – Profits optimization historical view (without revenues)

Profits optimization configuration

Users: ESP

Description: The user will configure the settings for running a new ATP optimization (Figure 28). He/she has to fill the following data:
Country: For which country this optimization is being performed. This field is mandatory

- From, To: The dates for the optimization. This field is mandatory
- Markets: The markets this optimization is being performed for. If the DLFM market is selected, 1 location area has to be selected from the available ones. At least 1 market has to be selected
- Storage units: The characteristics of the storage units configured for this optimization. At least 1 storage unit has to be configured filling all its 6 fields. By clicking on the "Add" button, a new set of empty 6 fields will appear below the filled ones. By clicking on the "Delete" button, that storage unit will be removed from the configuration (this button won't appear on the first row as one value at least 1 is mandatory)

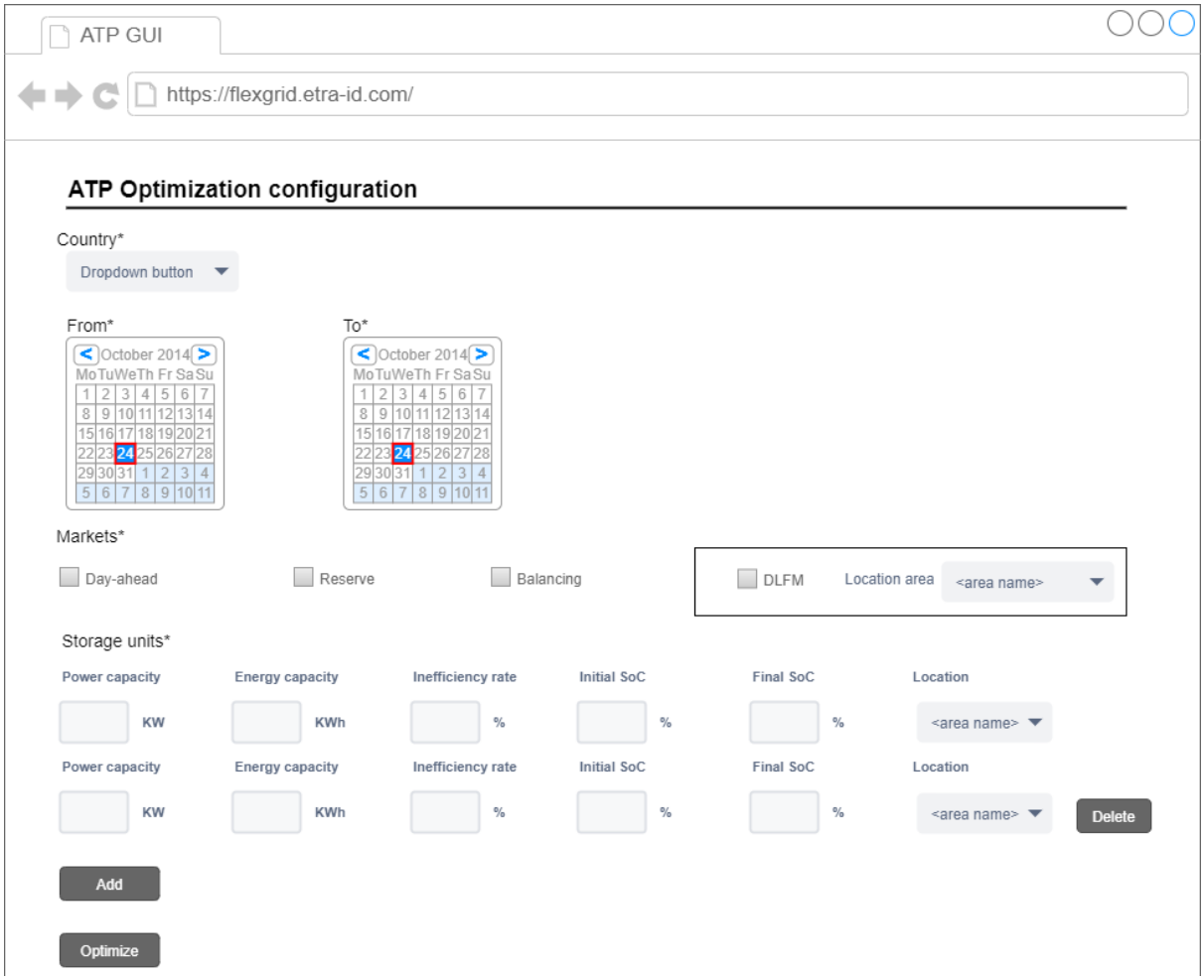


Figure 28: UCS2.3 – Profits optimization configuration

Once all the settings have been configured, by clicking on the "Optimize" button the ATP optimization process will be triggered. Once it has finished, the results will be presented on the screen *Profits optimization results (with revenues)*.

Profits optimization results (with revenues)

Users: ESP

Description: There are two possible ways the ESP user can access to this screen (Figure 29):

- By selecting one optimization from the list in the *Profits optimization historical view (with revenues)*
- From the
- *Profits optimization* configuration view, once the user triggers the optimization process and it has finished

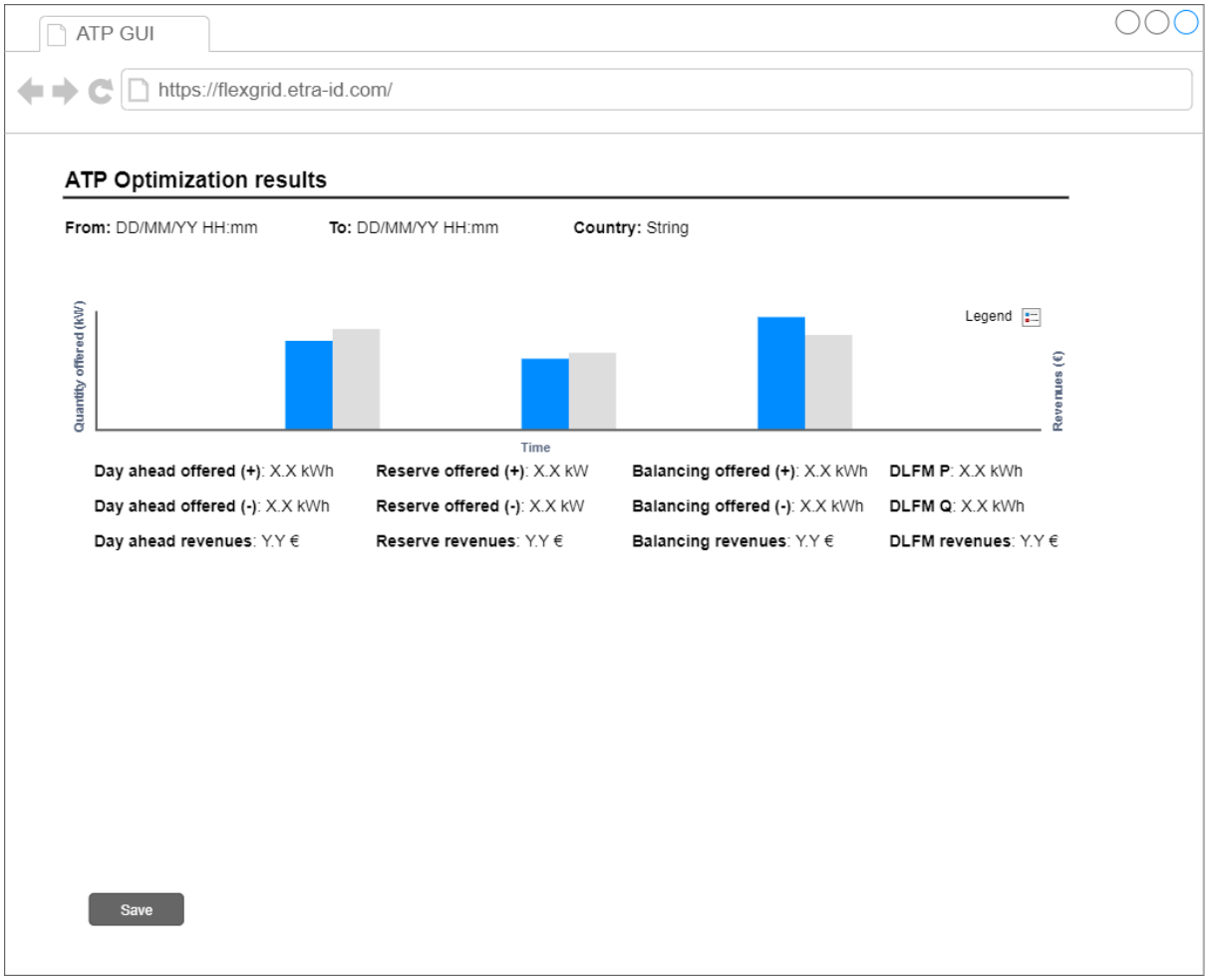


Figure 29: UCS2.3 – Profits optimization (with revenues)

The information presented here contains:

- The time period covered in this optimization
- The country this optimization was performed for
- A unique bar chart containing the results from the selected markets when the optimization was configured. By clicking on a series name (the ones from both y-axis) on the legend, it will be possible to hide that series (or to put it back again
    - Below the chart there will be up to 12 values (one for each series) with the aggregated data during the entire period. The total amount of quantity offered (left y-axis) is measured in kWh (kW for the reserve market), and the total amount of revenues (right y-axis) in the currency of the country.

By clicking on the "Save" button, the optimization results will be stored on the Central DB.

Profits optimization results (without revenues)

Users: DSO/FMO

Description: There is only one possible way the DSO/FMO user can access to this screen:

- By selecting one optimization from the list in the *Profits optimization historical view (without revenues)*

This screen will be almost the same as the previous one, with the following differences:

- No revenues information is presented
- No "Save" button is presented

Figure 30: UCS2.3 – Profits optimization (without revenues)

3.3.7 UCS 4.1– Manage a FlexRequest

FlexRequests dispatch optimization historical view

Users: Aggregator

Main front-end view:

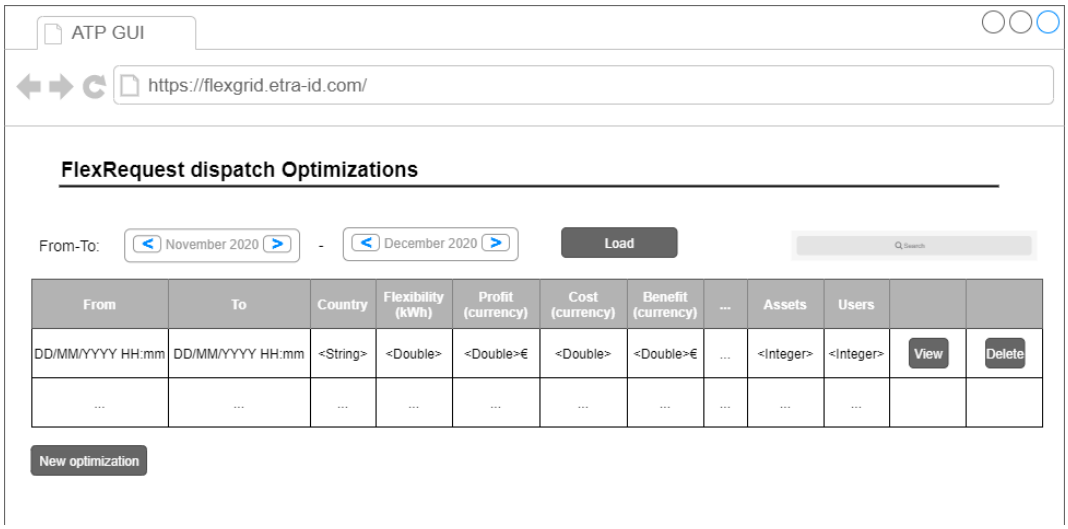


Figure 31: UCS4.1 – FlexRequest dispatch optimization historical view

Description: By default, all the stored optimizations for the actual month will be listed (Figure 25).

Basic information will be presented:

- The duration of the optimization (**From**, **To**). For this specific UCS the duration of the optimization will be a single day.
- The country or location the optimization was run for (**Country/Location**).
- The total amount of Flexibility that should be delivered (measured in kWh) as result of the optimization (**Flexibility**).
- The amount of money the aggregator could receive (**Revenue**) for delivering that flexibility under these circumstances and with these specific settings.

- The amount of money the aggregator will have to remunerate to its portfolio (Cost) for delivering that flexibility.
- The total amount of profit as result of the optimization (**Profit**).
- The number of assets and users involved in each optimization (Assets and Users).

Additional actions can be done:

- To search for a concrete value by using the text box above the table. (Select between past "manage a FlexRequest" optimization).
- By clicking on the "view" button, a screen with the details of the selected optimization will open.
- By clicking on the "delete" button the selected optimization will be removed.
- By clicking on the "New optimization" button a screen for configuring a new optimization to run the algorithm will open.

FlexRequest dispatch optimization configuration

Users: Aggregator
Description: The user will configure the settings to run a new FlexRequest dispatch optimization as follows:

Firstly, to the following need to be selected (Figure 32):

- *Country*: For which country or location (if more granularity is needed) this optimization is being performed. This field is mandatory.
- *From*, *To*: The dates for the optimization (mandatory). A single day will be the maximum period to run the optimization
- *Granularity*: The granularity of the data obtained (mandatory)

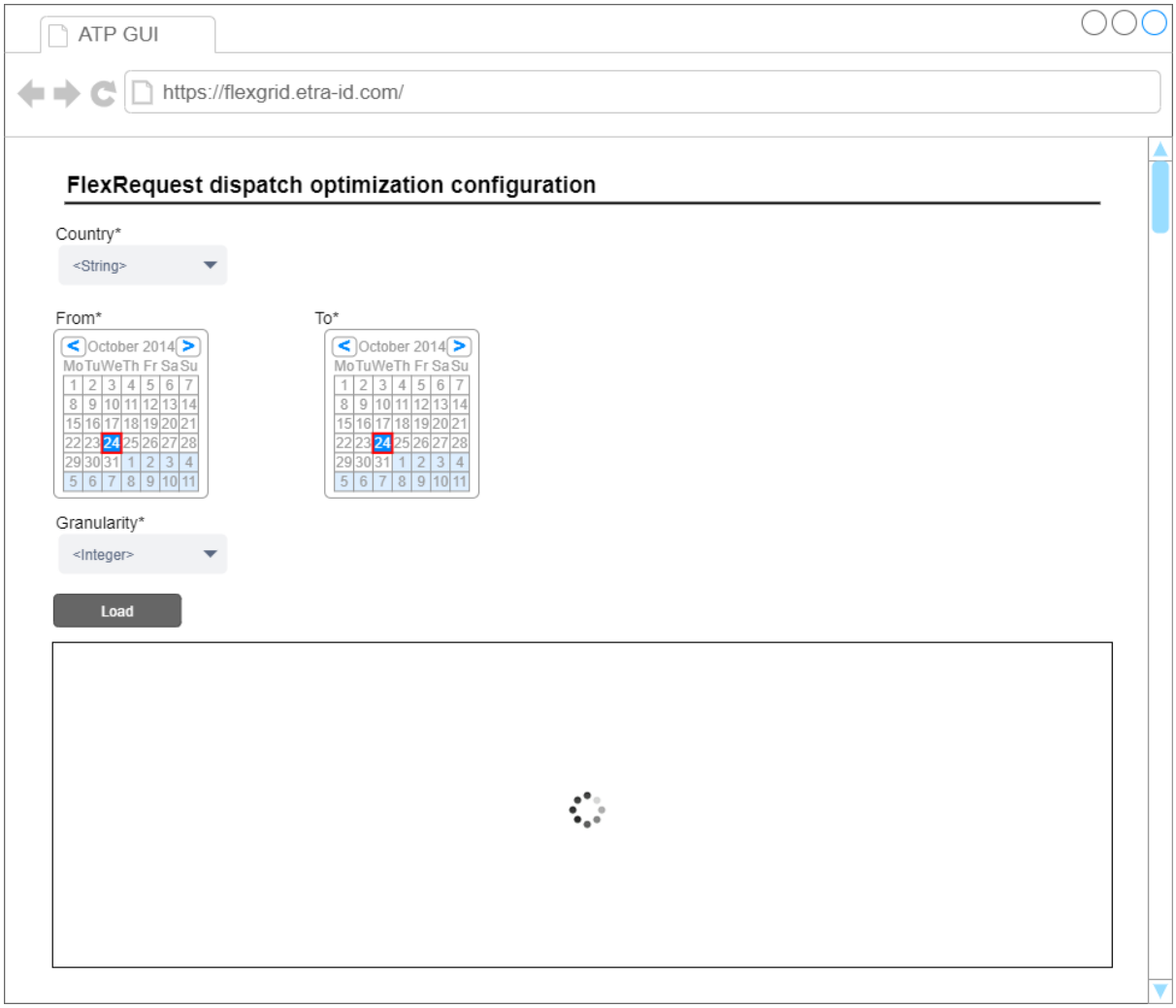


Figure 32: UCS4.1 – FlexRequest dispatch optimization configuration 1/10

Once these three mandatory fields have been filled, by clicking on the "Load" button it will be retrieved the information for "FlexRequest-Dispatch" which is an input and the "Portfolio" and "FlexRequest-Reserve"

For the *FlexRequest-Dispatch:*

- Information about total flexibility and revenues will be shown. By clicking on the "View" button it will be possible to see the area affected by the selected FlexRequest-Dispatch on a map (Figure 30).

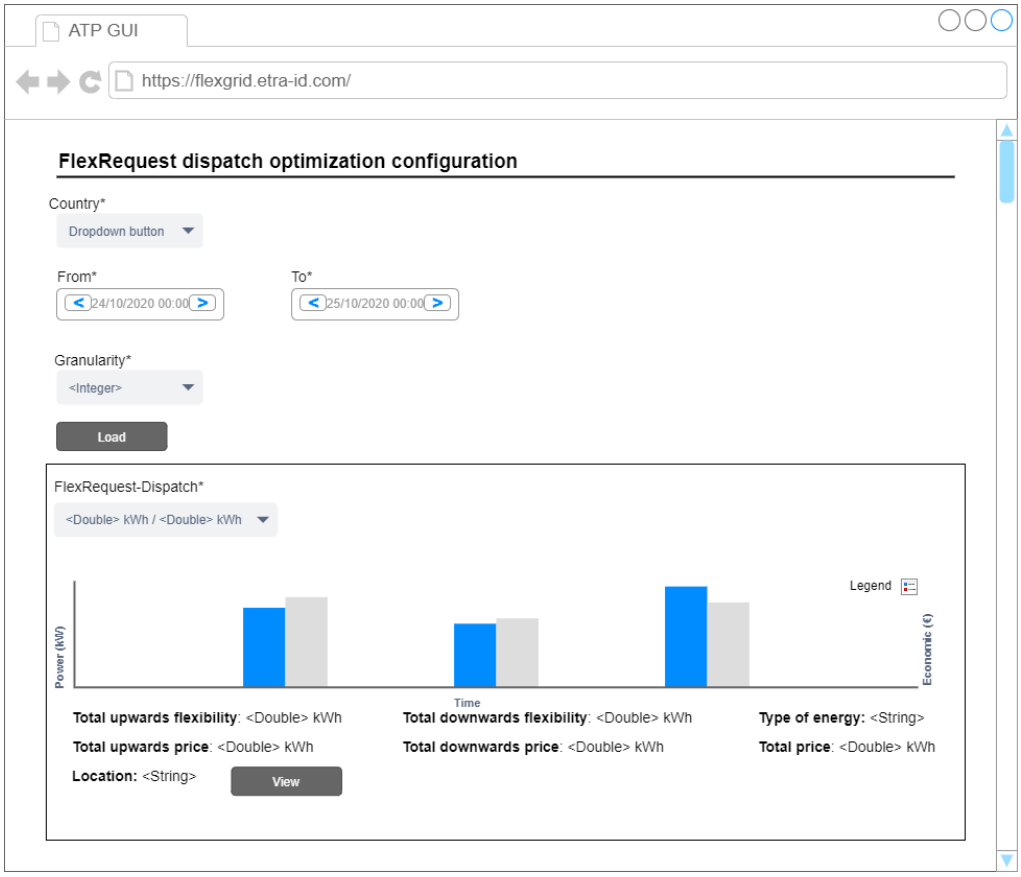


Figure 33: UCS4.1 – FlexRequest dispatch optimization configuration 2/10

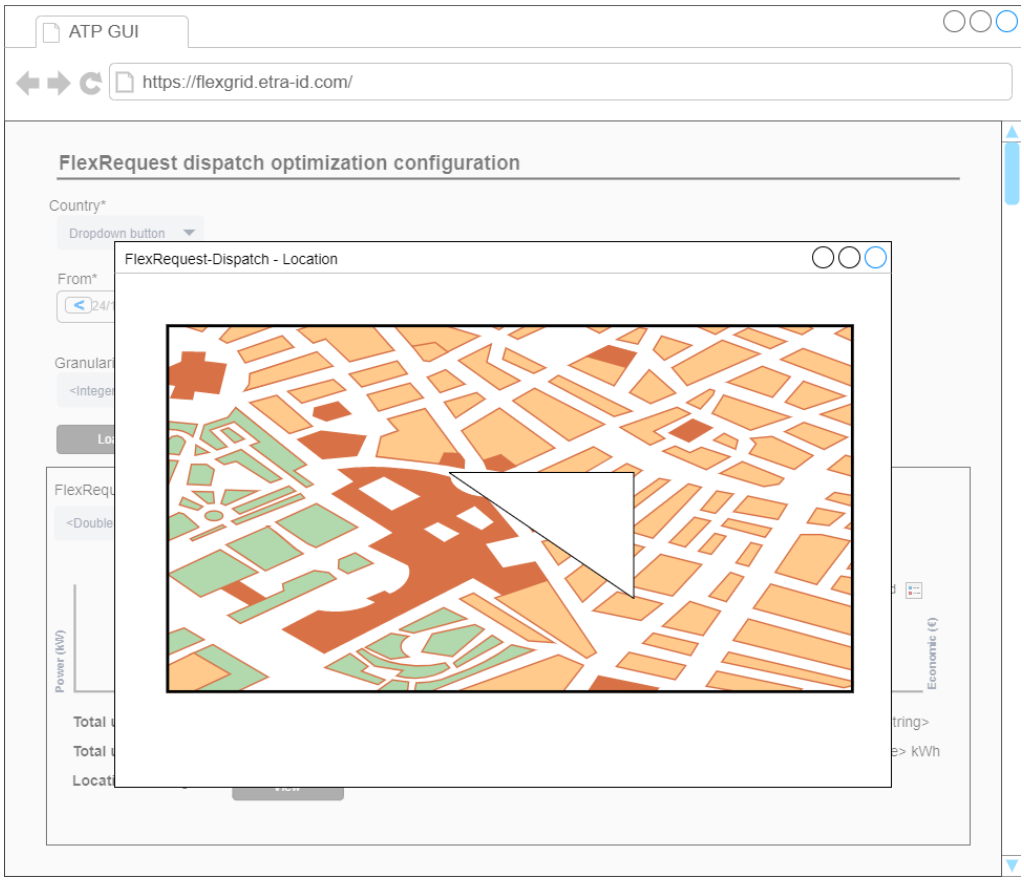


Figure 34: UCS4.1 – FlexRequest dispatch optimization configuration 3/10

For the *Portfolio* :

- At the main view (Figure 35) the portfolios of the aggregator currently running the optimization are shown including all the reserved flexibility
- From that view it is possible to modify the assets included in the portfolio
- With the "Load" button it will be presented on a table all the assets registered for the selected country and located market area. All the information will be showed in a graphical way as presented in Figure 35.
- In the view it is possible to simulate different scenarios by clicking on "Simulate activation" button.

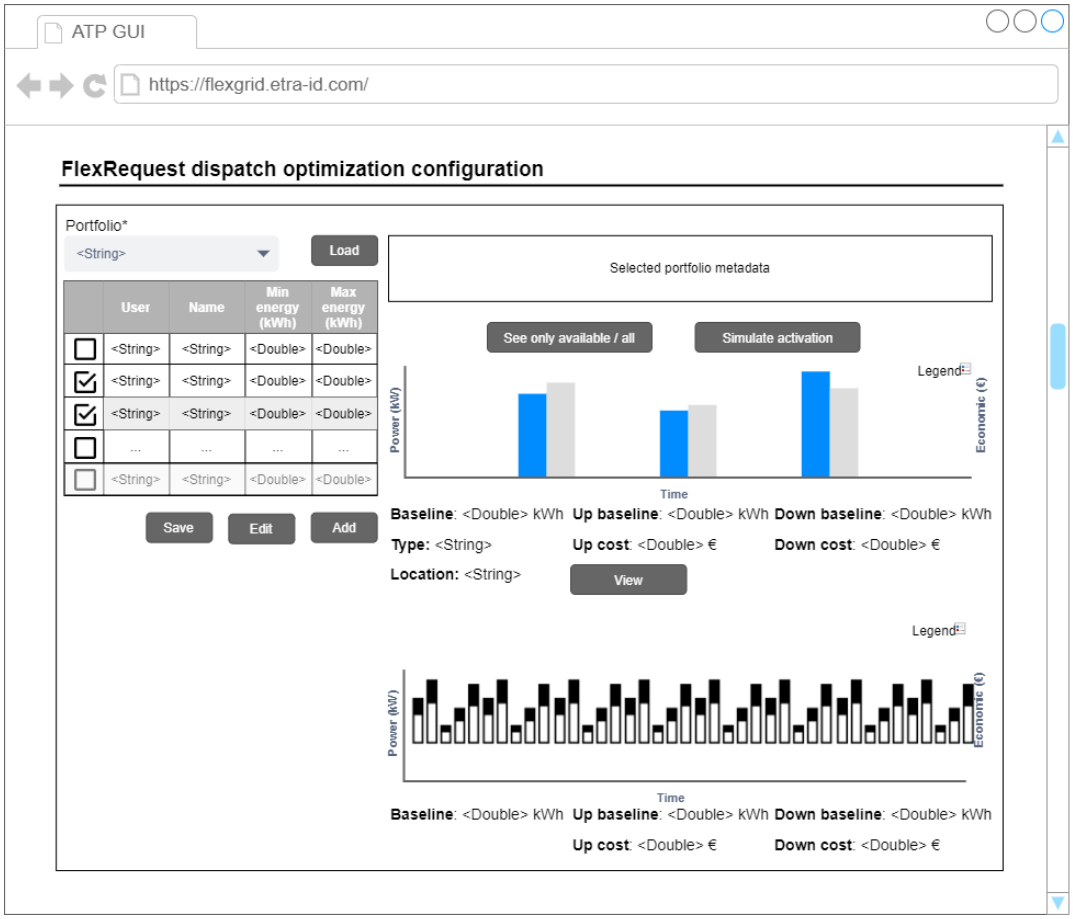


Figure 35: UCS4.1 – FlexRequest dispatch optimization configuration 4/10

For each asset (from the table) the following information is shown:

- Baseline: Is the energy of the asset if no flexibility is activated so, the baseline shows the scheduled demand or production during the selected date (measured in kWh).
- Up Flexibility: the availability to increase consumption flexibility during the selected date (measured in kWh).
- Down Flexibility: availability to decrease consumption during the selected date (measured in kWh).
- Up/Down revenue: The remuneration the user would receive for activating its flexibility (measured in the currency of the country).

By default, it will only be presented information during the MTUs where the FlexAsset is available. By clicking on the "See only available/all" it will also be presented the information

during the non-available MTUs. That information can be hidden once again by clicking again this button.

It will be possible to modify (Figure 36) some of the information and values of the listed assets. It will not be possible to modify the following fields: type of asset, location, and user. Regarding the baseline, flexibility for up and down regulation, revenue, cost, availability and status, it will be possible to modify all those values for all the existing MTUs

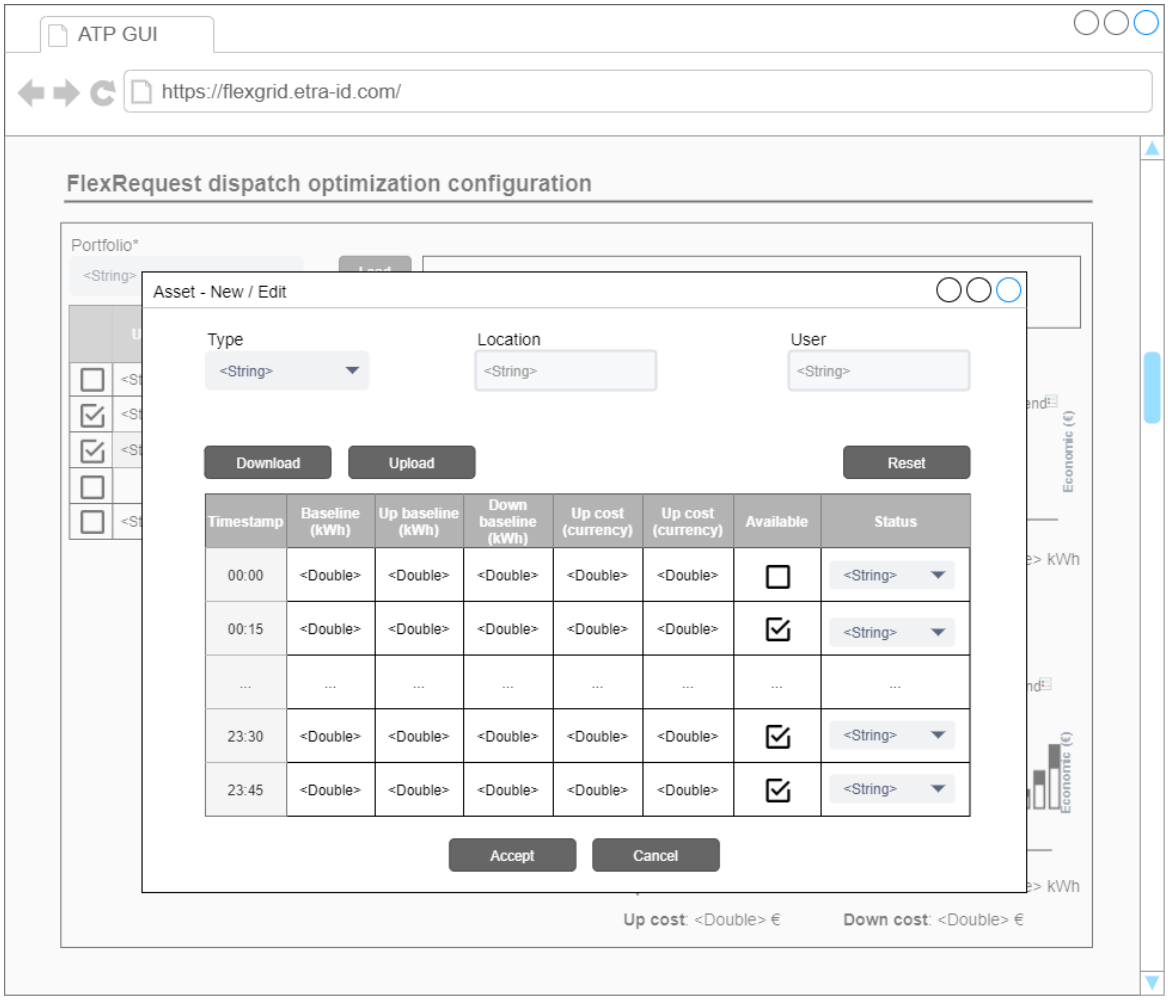


Figure 36: UCS4.1 – FlexRequest dispatch optimization configuration 5/10

Toarrange a simulation the ATP users should click on the “Simulate activation” button. A window will pop up (Figure 37) indicating:

- The starting MTU for the simulation. On the dropdown it will be presented the timestamp for the MTU and the total flexibility available on it.
- The amount of flexibility considering the different assets before selected at that MTU to be considered for the simulation.

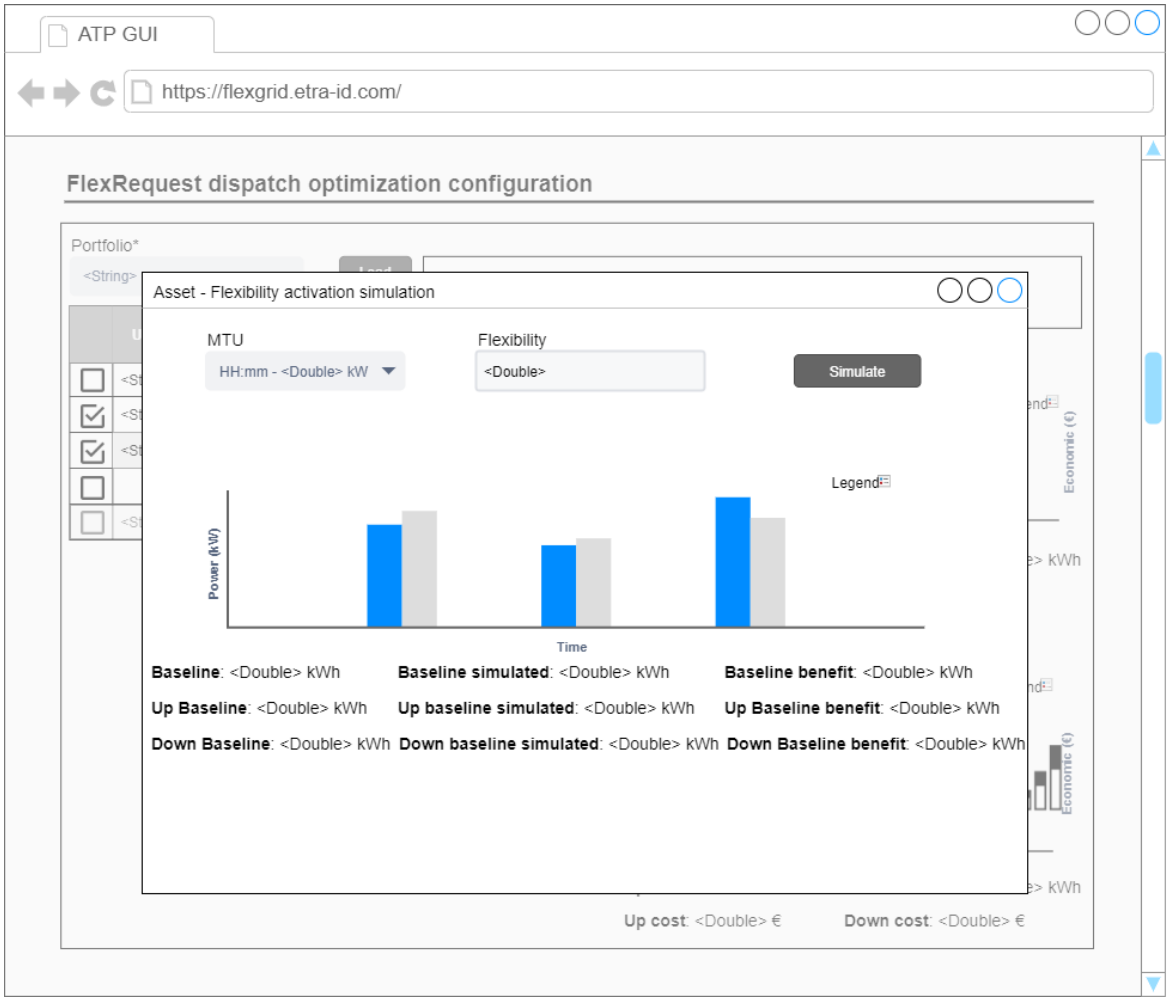


Figure 37: UCS4.1 – FlexRequest dispatch optimization configuration 6/10

When the simulation runs by clicking the "Simulate" button is clicked, a new chart is presented with relevant information:

- For the entire time window (single day):
  - The planned operation before the optimization (measured in kW)
  - Available Flexibility (measured in kW)
  - The modified operation as result of the optimization (measured in kW)
  - Updated portfolio

And below , this information at aggregated level during the entire time frame (measured in kWh), not only during the selected MTU onwards

- Baseline: The baseline during the simulation time before running the algorithm.
- Operation with Flexibility Activation: T Flexibility Activated.
- Monetary profit.

To do so, when clicking on an asset from the table (there is no need to check it) and pressing the "Edit" button (Figure 35) a new window pops up with all relevant asset characteristics current information for that asset; for editing this data two approaches can be followed:

- Manually: For the numeric values, by clicking on the cells and typing the new numeric values. For the availability selecting/deselecting the checkbox. For the status selecting it from the dropdown.

- Uploading it from a .csv file: Firstly the template has to be downloaded by clicking on the “Download” button (this template will already contain the actual values of the table). After filling it, it has to be uploaded back again by clicking on the “Upload” button. Once fully loaded, all its information will be presented on the table.

By clicking on the “Accept” button all the changes will be applied, and the asset will appear in italics on the table to indicate it has been modified.

It will also be possible to define a new asset by clicking on the “Add” button (Figure 35). The screen will be the same as for the editing process (Figure 36), but in this case the “type of asset” field can be filled; on the other hand, the location will automatically be set to the same as in the *FlexRequest-Dispatch* selected, while the user will remain empty. Also, the .csv file downloaded in this case will contain 0 values for all the numeric fields. Once the new asset has been accepted by the aggregator (by clicking on the “Accept” button) it will appear with the text in green on the list, with the checkbox checked.

At any moment the user can set back the settings of the assets to their original values by selecting it, clicking on the “Edit” button and then on “Reset” (Figure 36). In the case of the new created assets, they will be set back again to 0 values.

In both cases, the edited assets and the new ones remain visible in this section of the GUI until the user delete it, they won’t be stored on the system.

Below the chart on the asset level, a second chart is presented containing all the aggregated information of all the selected assets (Figure 35). This chart will only consider the MTUs of the selected assets with availability. This chart will contain:

- Baseline: The baseline during the simulation time before running the algorithm.
- Operation with Flexibility Activation: T Flexibility Activated.
- Monetary profit)

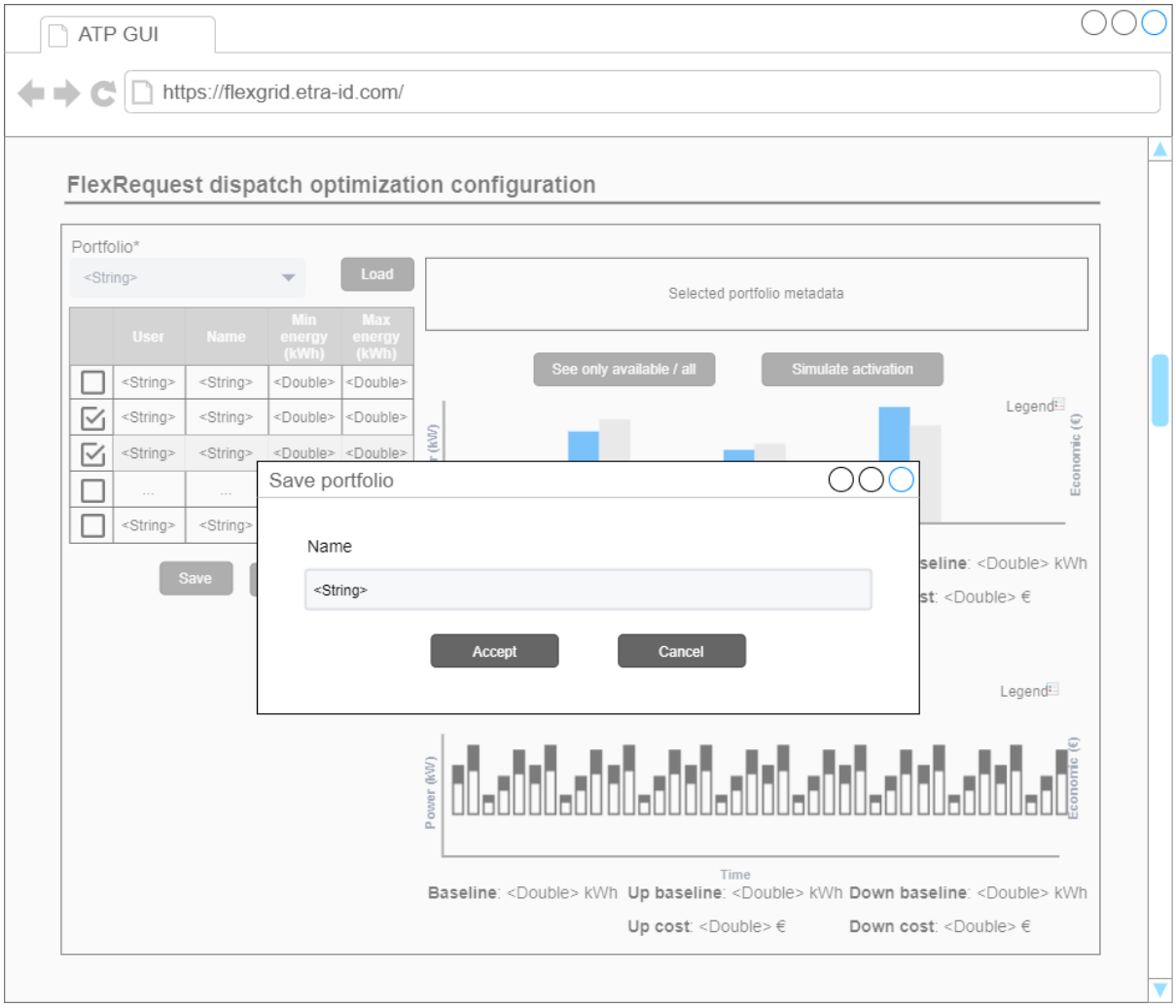


Figure 38: UCS4.1 – FlexRequest dispatch optimization configuration 7/10

It will also be possible to store the modified portfolios resulting from the previously mentioned operations: create new assets, edit assets , add assets to the selected portfolio, and remove assets from the selected portfolio, by clicking on the "Save" button (Figure 35). Once clicked a new window pops up (Figure 38) to name the customized portfolio. By clicking on the "Accept" button, the saved portfolio will be added to the *Portfolio* dropdown, it will appear as selected, and its text will appear in green.

Similar to the asset operations, this newly created portfolio will remain visible on this section of the GUI until the user leaves it, it won't be stored in the system. The aggregation will implement and run many optimizations, although a button for storage the portfolio defined can be included, not all the configurations are recommended to be stored in the DB as not all the new portfolio defined will be useful for the future

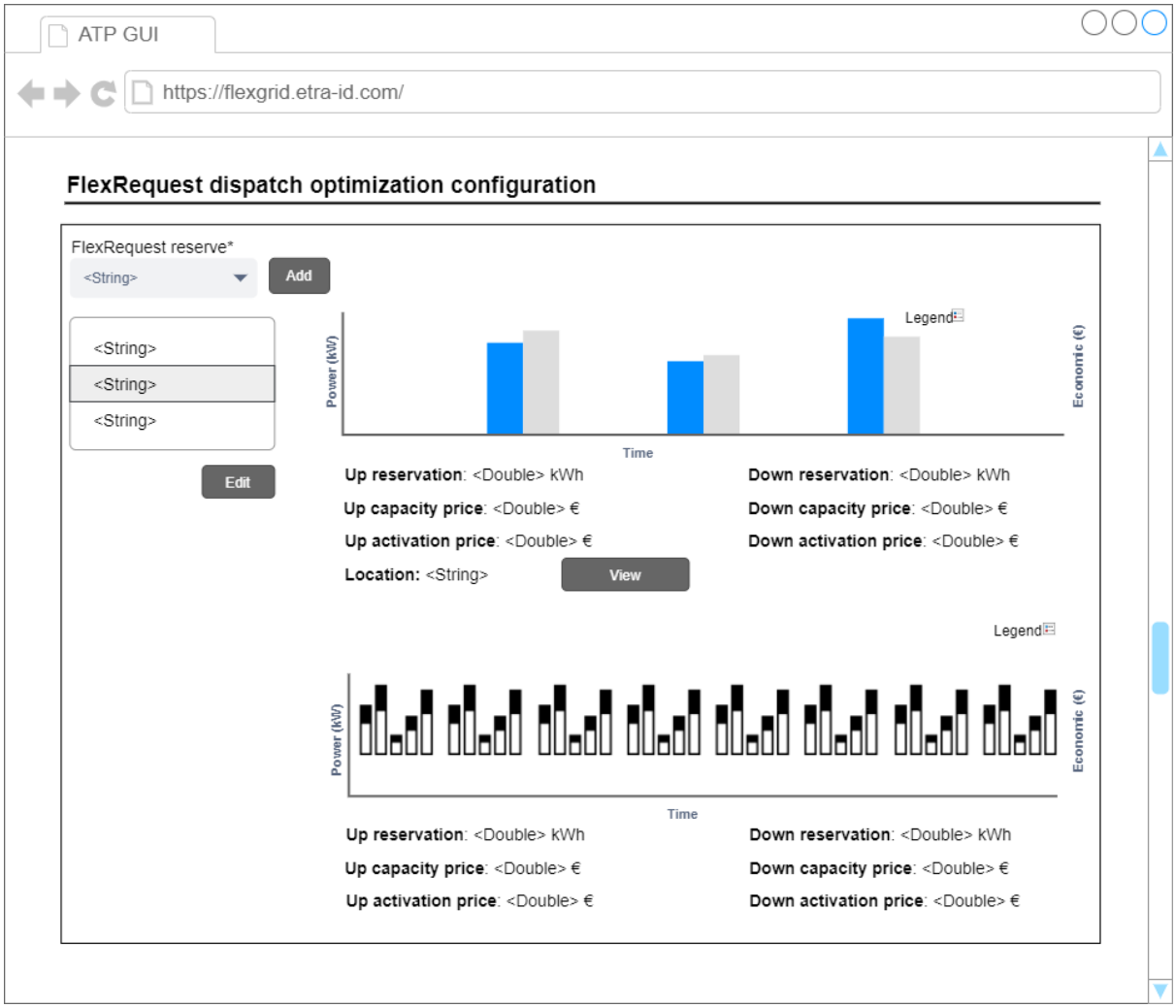


Figure 39: UCS4.1 – FlexRequest dispatch optimization configuration 8/10

Ultimately, some other already existing FlexRequests must be selected to be taken into account by the algorithm (Figure 39). Only *FlexRequest-Reserve* in the same area as the selected *FlexRequest-Dispatch* will be presented on this dropdown.

Once a FlexRequest-Reserve has been selected in the dropdown, its individual information will be presented on the top chart. By clicking on the "Add" button it will be added to the list of FlexRequests-Reserve to be considered by the algorithm.

The information presented on this chart will be:

- Up/Down reservation: The reserved flexibility (measured in kWh)
- Up/Down capacity revenue: The remuneration received just for having all the flexibility amount reserved (measured in the currency of the country)
- Up/Down activation price: The remuneration received for activating all the flexibility amount (measured in the currency of the country)

Below this chart it will also be presented all this information but at aggregated level during the entire period (measured in kWh), in addition with the location of the area where this FlexAsset is located; by clicking on the "View" button it will be possible to see the area on a map as explained before (Figure 34).

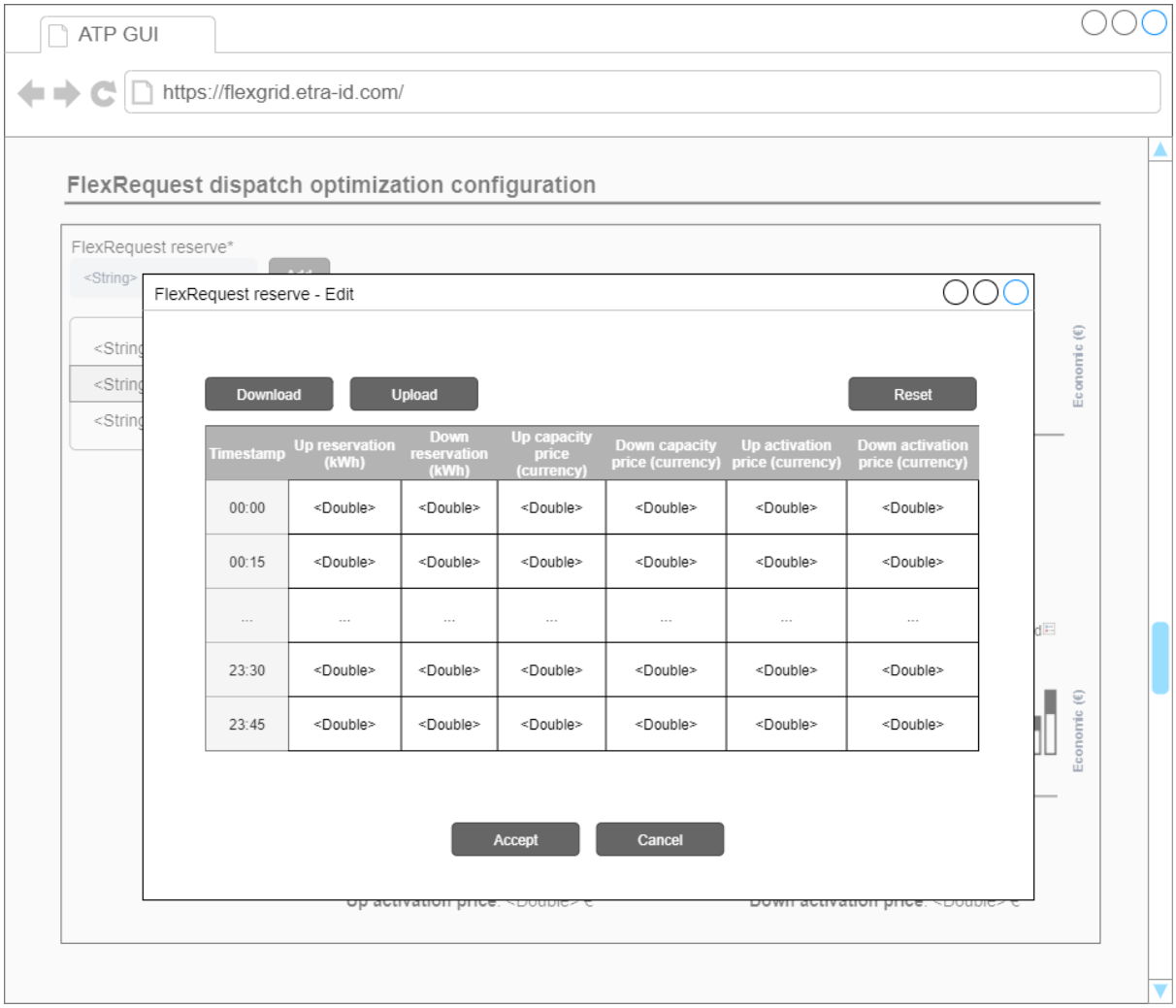


Figure 40: UCS4.1 – FlexRequest dispatch optimization configuration 9/10

By clicking on an element and on the "Edit" button it will be presented a pop-up window to edit some of its values (Figure 40). This information can be filled following the same steps as presented in the new assets/edit assets section (Figure 36). Once the changes are accepted, the edited *FlexRequest-Reserve* element will appear in italics in the table and in the dropdown

In addition, in the second chart it will appear the aggregated information of all the selected FlexRequest-Reserve.

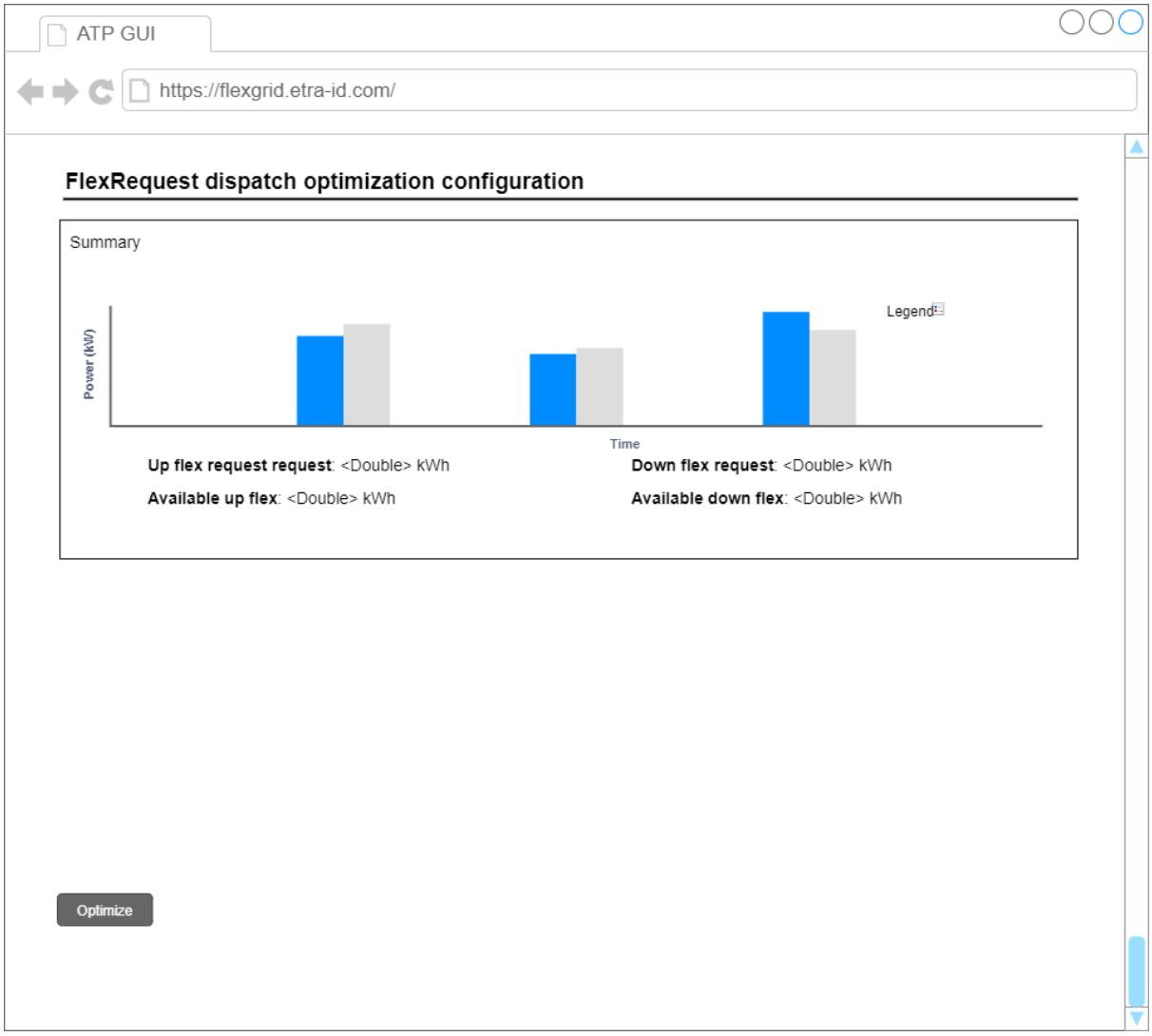


Figure 41: UCS4.1 – FlexRequest dispatch optimization configuration 10/10

Finally, at the end of this configuration section (Figure 41) it will be presented a final chart

- Up/Down flexibility requested: The flexibility of the *FlexRequest-Dispatch*
- Available up/down flexibility: The available flexibility calculated as the difference between the flexibility that can be obtained from the selected *portfolio* and the flexibility already reserved for the selected *FlexRequest-reserves*

Once all the settings have been configured, by clicking on the "Optimize" button the FlexRequest dispatch optimization process will be initiated. Once it has finished, the results will be presented on the screen *FlexRequest dispatch optimization results*.

FlexRequest dispatch optimization results

Users: Aggregator
Main front-end view:

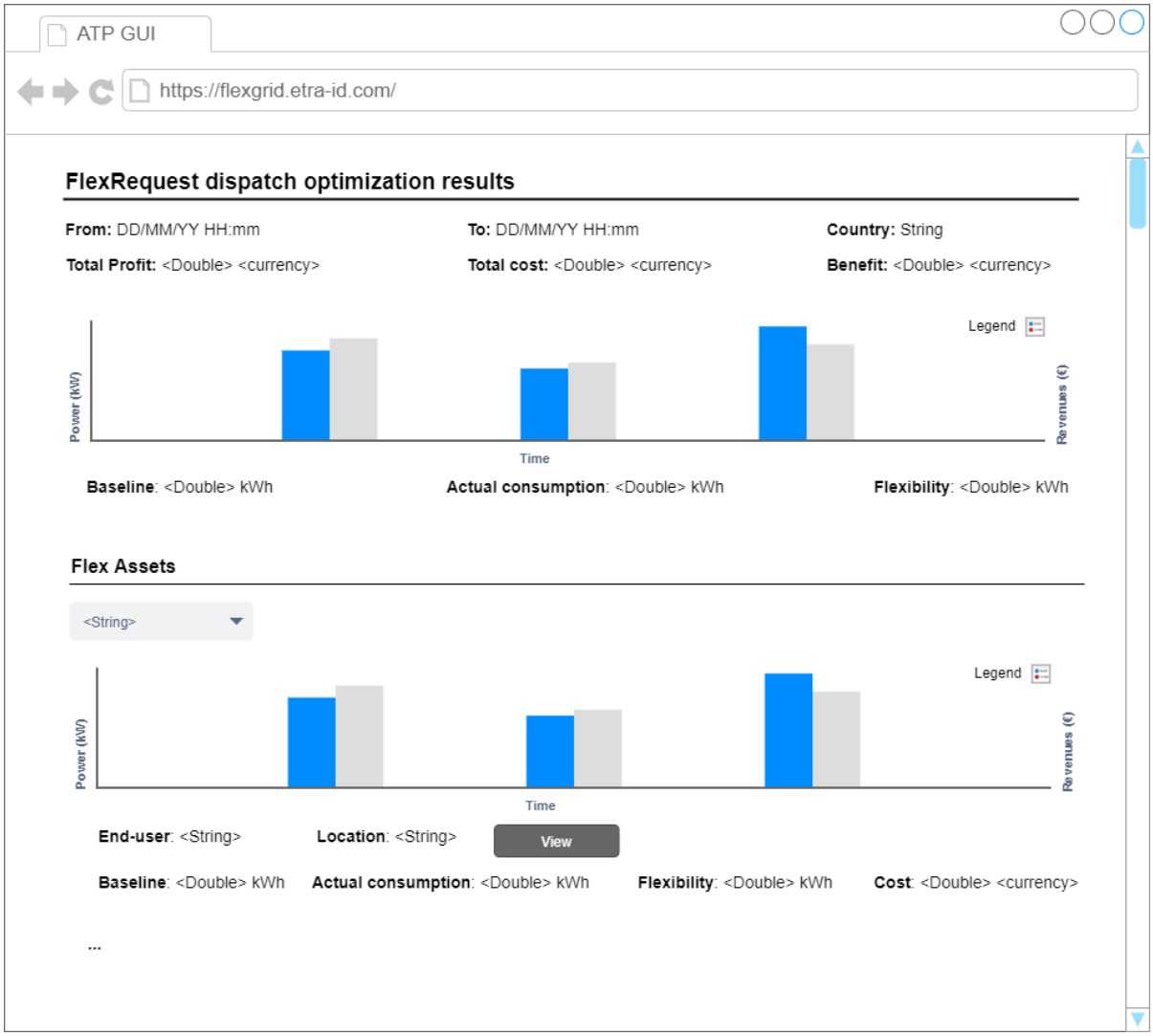


Figure 42: UCS4.1 – FlexRequest dispatch optimization results 1/2

Description: The information presented here can be categorized in 2 different groups: (i) General information and (ii) FlexAsset data. The Aggregator has two possible ways to visualize the information: (i) from the list in the *FlexRequest dispatch Optimization* or (ii) from the *FlexRequest dispatch Optimization configuration* view.

The information the Aggregator have access to (Figure 42):

- The period covered by this optimization
- The country this optimization was performed for
- The total profit this user would get by applying the results of this optimization
- The total cost this user would have by applying the results of this optimization
- The total benefit this user would have as result of this optimization (this value is the difference between profit and cost, and the goal is to have a positive value)
- A bar chart containing the aggregated results of all the assets involved in the optimization.

For the data at FlexAsset level (Figure 42), a dropdown menu will be visualized showing the assets considered in the optimization. Upon selection of an asset the following will be presented:

- A bar chart containing the same information presented in the previous one but showing only information for the selected asset. Additionally, the following information will be presented:
    - A new time series with the cost of activating this flexibility
    - Two new fields presented with the aggregated data below the chart:
        - The owner of this asset
        - The location of this asset. By clicking on the button, it can be visualized its position on a Map, as shown before (Figure 34).

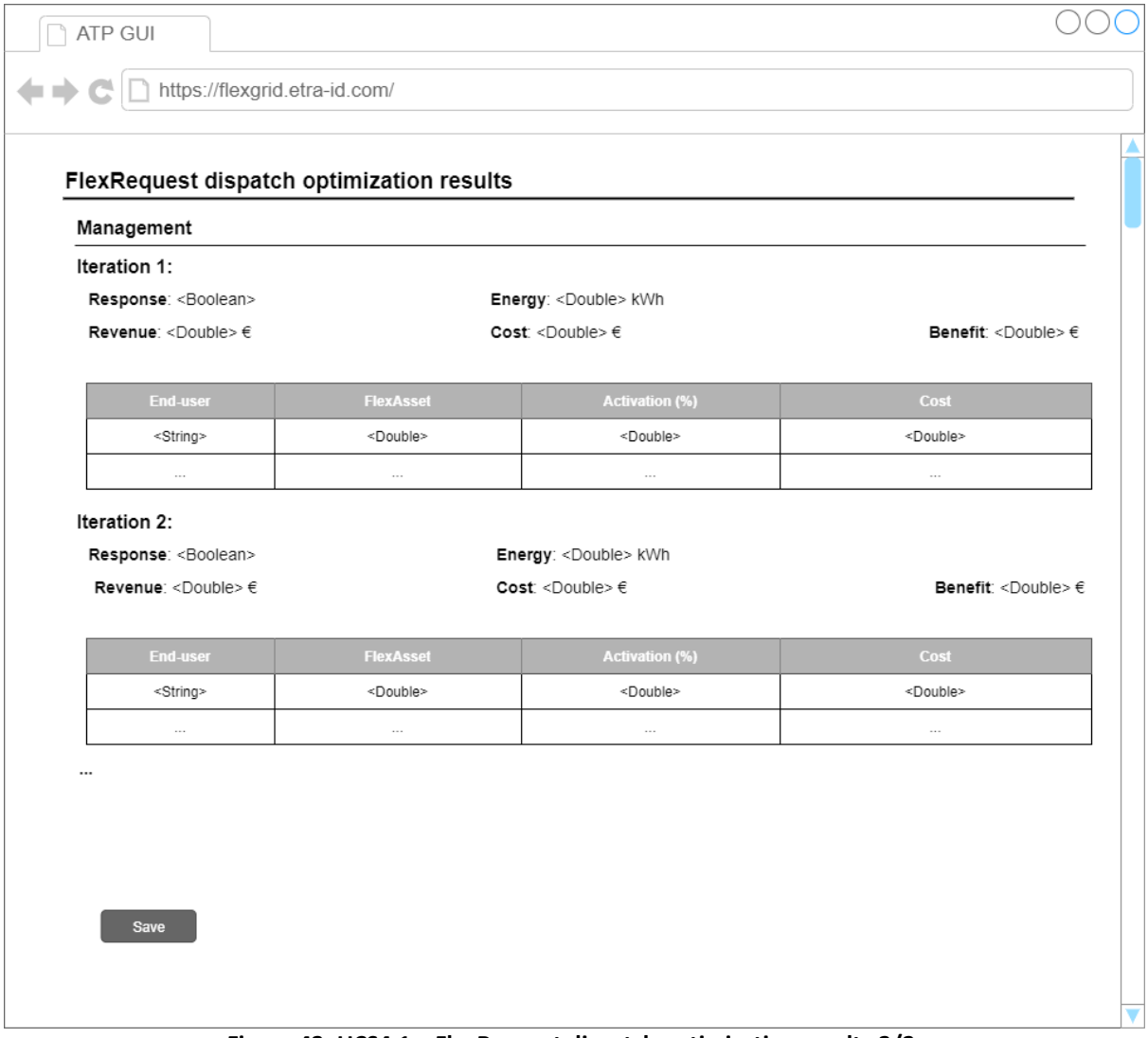


Figure 43: UCS4.1 – FlexRequest dispatch optimization results 2/2

Information about each algorithm iteration will be presented (Figure 43):

- Response: If the response during this iteration was for upward regulation (positive) or downwards regulation (negative)
- Revenue: The income remuneration from the FlexRequest
- Energy: Amount of flexible energy
- Cost: The remuneration to the assets participating on this FlexRequest
- Benefit: The profit. The difference between revenue and cost

In the table it will be presented 1 row for each involved FlexAsset, containing:

- End-user: The user this asset belongs to
- FlexAsset: The name of this FlexAsset
- Activation: The percentage of flexibility activated
- Profit: The remuneration the user will get for the activation

By clicking on the “Save” button, the optimization results will be stored on the Central DB.

3.3.8 UCS 4.2– Manage a novel B2C flexibility market

Retail pricing optimization historical view

Users: Aggregator

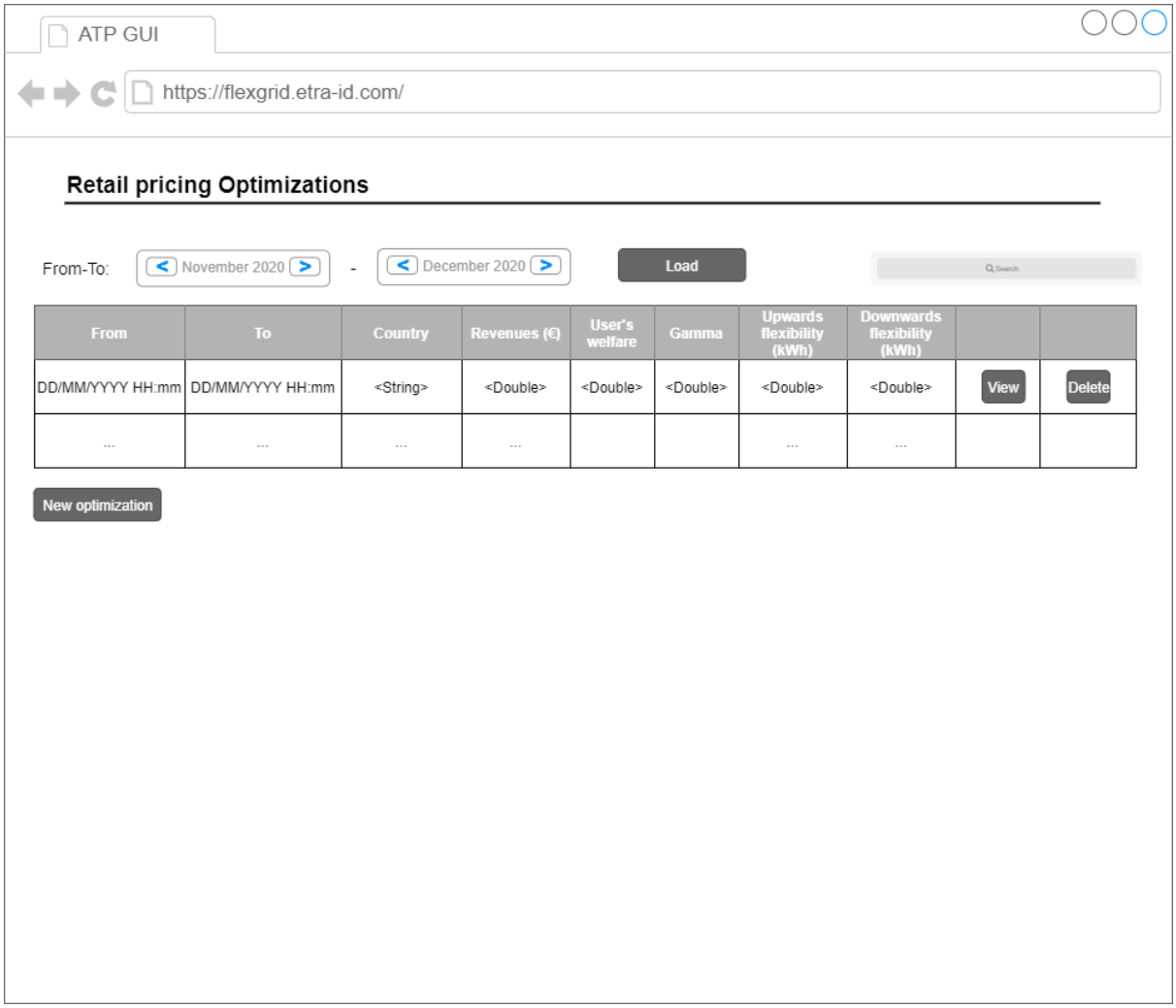


Figure 44: UCS4.2 – Retail pricing optimization historical view

Description: All the stored optimizations for the actual month will be listed (Figure 44). It will also be possible to retrieve the stored optimizations for past operations (days, months…) by selecting a range of dates and clicking on the “Load” button

It will be presented basic information including:

- The duration of the optimization (**From**, **To**)
- The country the optimization was run for (**Country**)

- The received remuneration for activating all the flexibility requested by the DSO (**Revenues**). Measured in the currency of the country
- The foreseen average aggregated user's welfare if all the requested flexibility is activated (**User's welfare**)
- Gamma: The average gamma value for the entire optimization (Gamma). The Gamma value represents the type of FlexContract (or else retail pricing scheme). When γ=0, we have the Real Time Pricing (RTP) model in which all end users get the same reward in €/flexibility unit, even though some of them did not contribute anything in the FlexRequest. When γ=1, we have a fully personalized RTP scheme, in which the flexible end users get rewarded according to each one's contribution, while inflexible end users do not get any reward. When γ>1 (cf. emergency network situation), then the inflexible end users get penalized, because they did not contribute anything in a case of a critical FlexRequest.
- The total quantity of flexibility in upward and downward regulations that can be delivered by the selected portfolio for this optimization during the entire time frame

Other functionalities available will be:

- To sort the table according to one selected column.
- To search for a concrete value by using the text box above the table.

By selecting different button defined it will be possible to:

- By clicking on the "view" button the screen with the details of the selected optimization will be opened.
- By clicking on the "delete" button the selected optimization will be removed.
- By clicking on the "New optimization" button a window for configuring a new optimization will be opened.

Retail pricing optimization configuration

Users: Aggregator
Description: The user will configure the settings to run a new retail pricing optimization. He/she has to follow the steps depicted below.

Firstly it has to be selected the following mandatory information (Figure 45):

- *Country*: For which country this optimization is being performed.
- *From*, *To*: The dates for the optimization.
- *Granularity*: The granularity of the time series that is being used in this optimization. It can be 15-minutes, 1-hour, 1 day.
- *Timeframe*: The desired timeframe of the week/day for running the optimization. It can be default (complete week) or any combination the aggregator needs (Mon-Fri only, Weekend only, night hour only…).

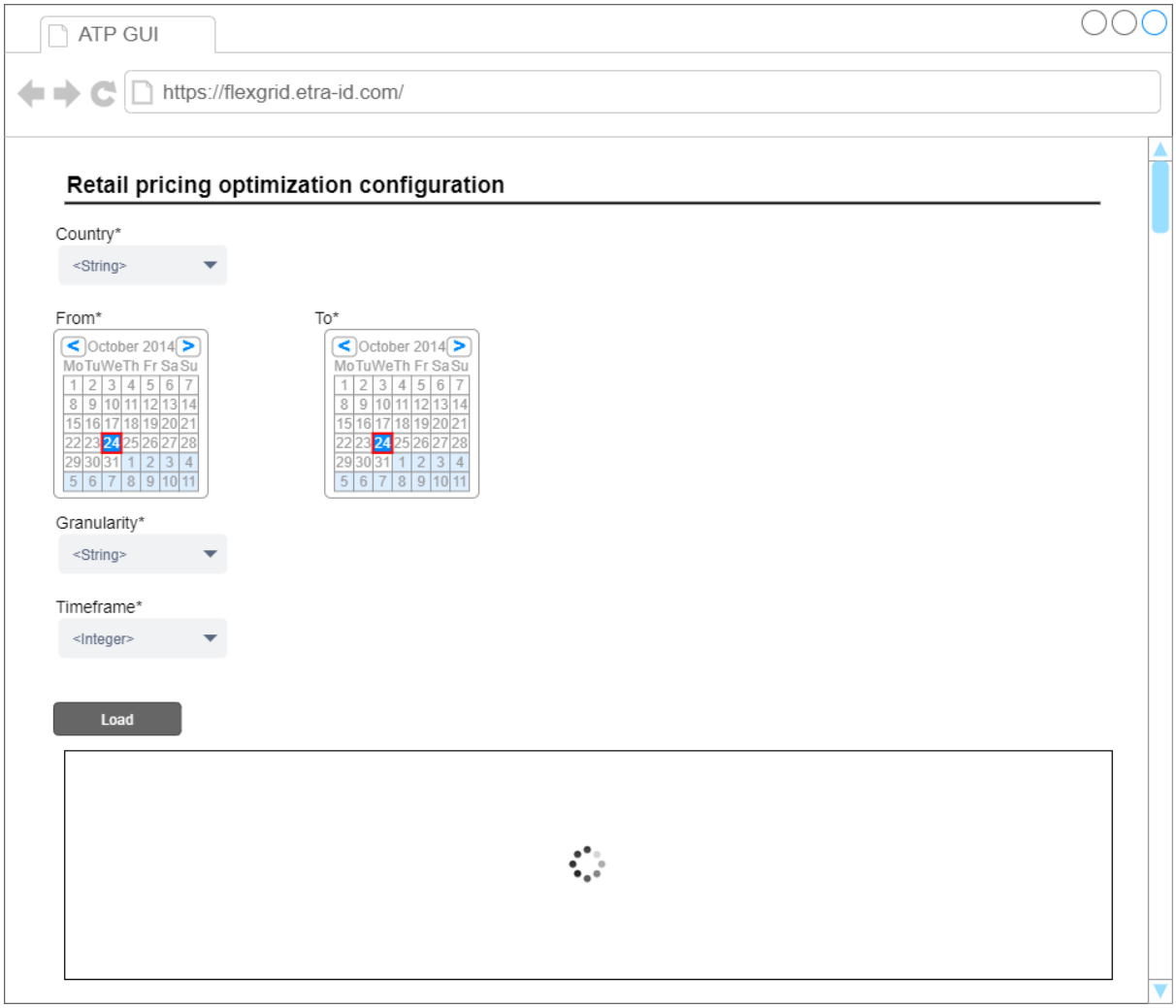


Figure 45: UCS4.2 – Retail pricing optimization configuration 1/6

Once these four mandatory fields have been filled, by clicking on the “Load” button all the *FlexRequest* for the selected dates will be retrieved (Figure 46). Per default one of the offers will be automatically selected on the dropdown and its information will be presented in a chart; by selecting another *FlexRequest* from the dropdown, the respective information will be presented in the chart.

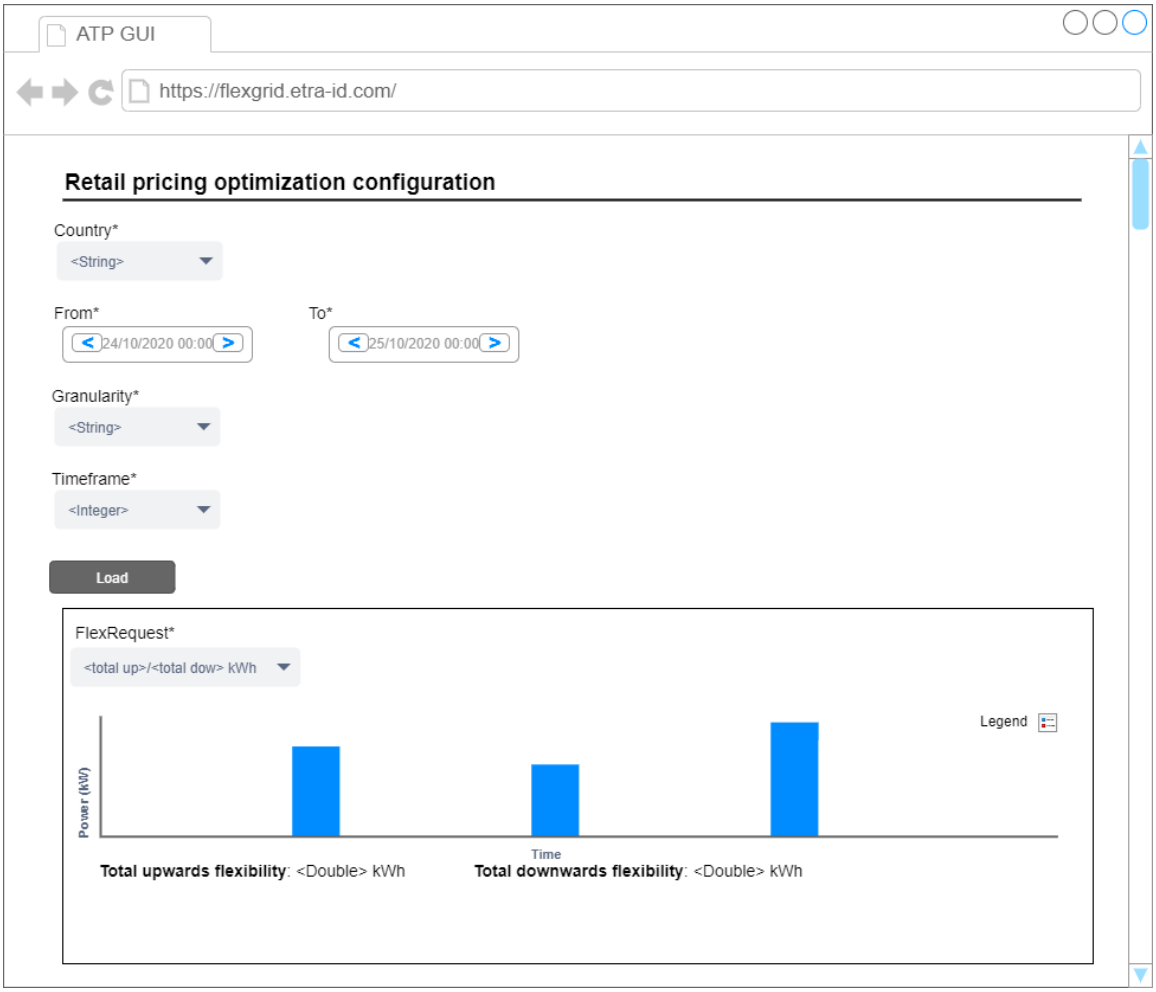


Figure 46: UCS4.2 – Retail pricing optimization configuration 2/6

The next to be filled are the *Users* to be included in the optimization. For doing this the aggregator should use the dropdown menu on which one or several user(s) can be selected. The selectable users belong only to the selected country for the optimization. Once the selection has been finished, by clicking on the "Load" button it will be retrieved the aggregated demand and/or production from the selected users (Figure 47). This information is also mandatory.

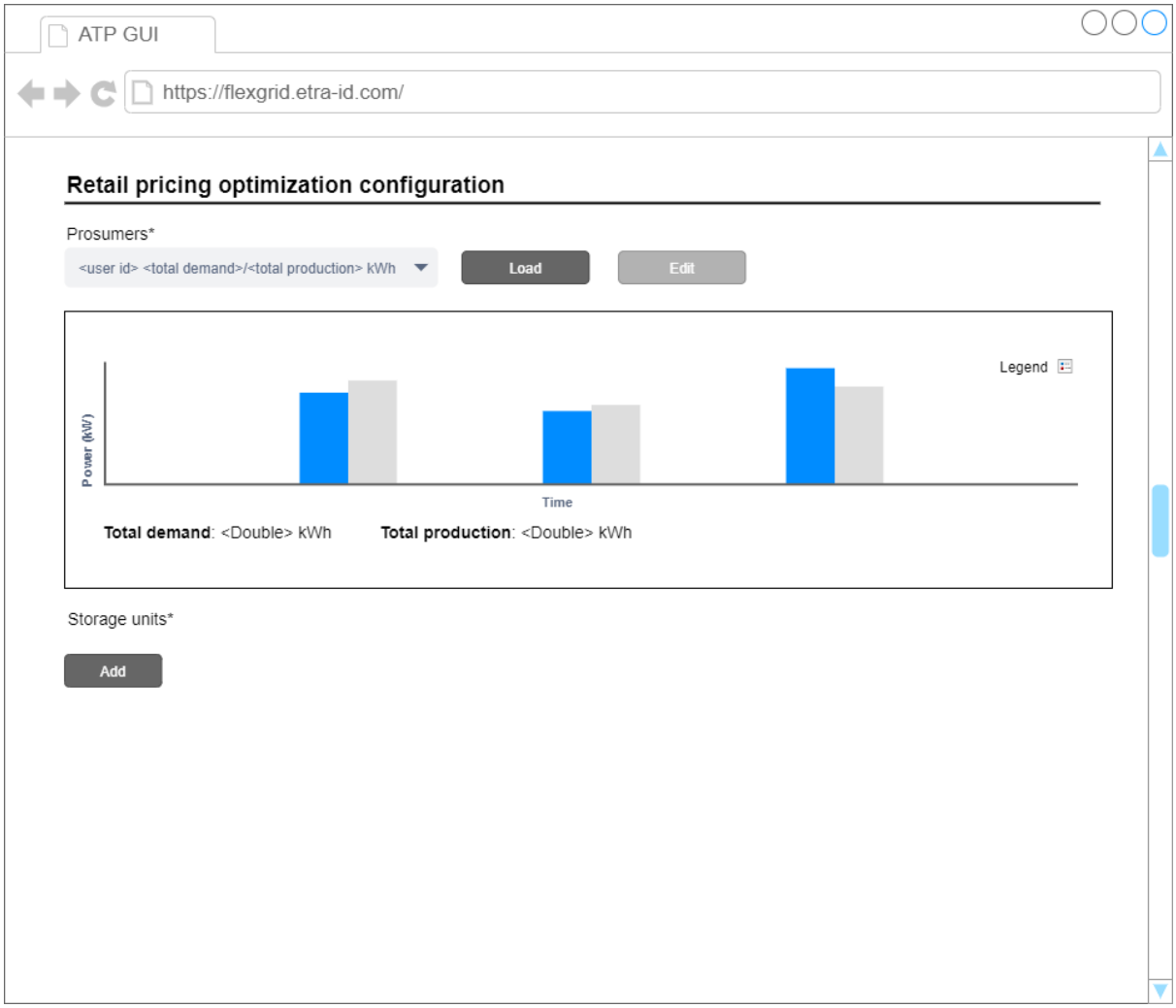


Figure 47: UCS4.2 – Retail pricing optimization configuration 3/6

After this, the *Storage Units* to be considered in the optimization process have to be selected. The "Add" button (Figure 47) will only be available after selecting 1 or more users; by clicking on it, it will be presented on a new pop-up window (Figure 48) a map with all the storage units available for the selected country on the 1st dropdown menu. By clicking on any element on the map, the respective information will be presented on the left side. By clicking on the "Select" button for each asset the information of the selected storage unit will be presented can be edited for optimization purposes only, so any update here will not be stored in the system (Figure 49).

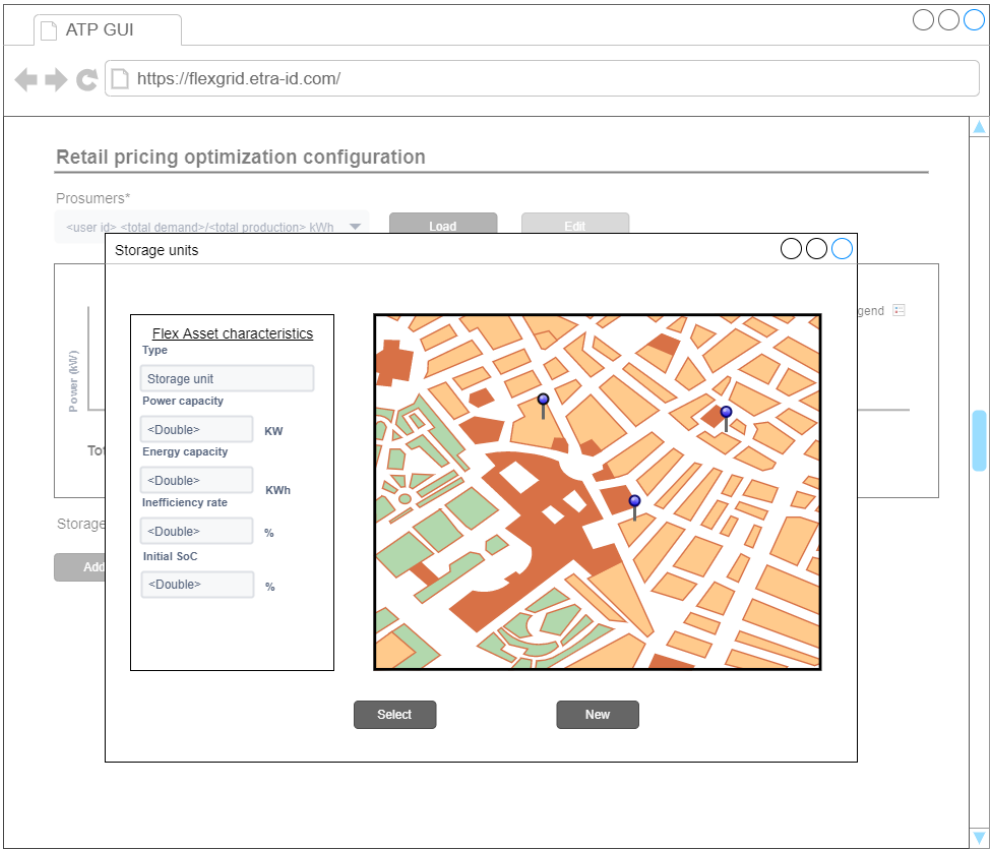


Figure 48: UCS4.2 – Retail pricing optimization configuration 4/6

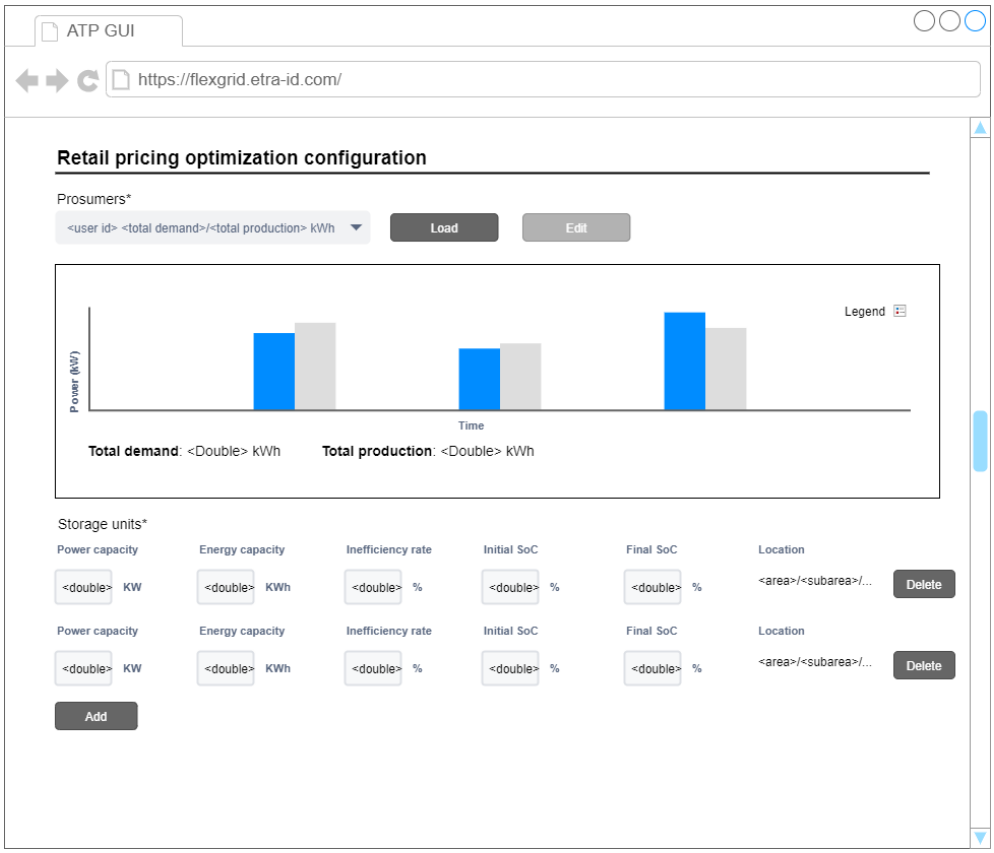


Figure 49: UCS4.2 – Retail pricing optimization configuration 5/6

New units can be added by clicking on the “New” button. This new asset will be only used for the optimization and will not be stored at the Central DB. By clicking the “Delete” button the selected asset will be removed.
In addition to the storage units, controllable loads should be necessary to run the algorithm. It is possible to edit, add and delete the loads in the same way as the storage units (Figure 50).

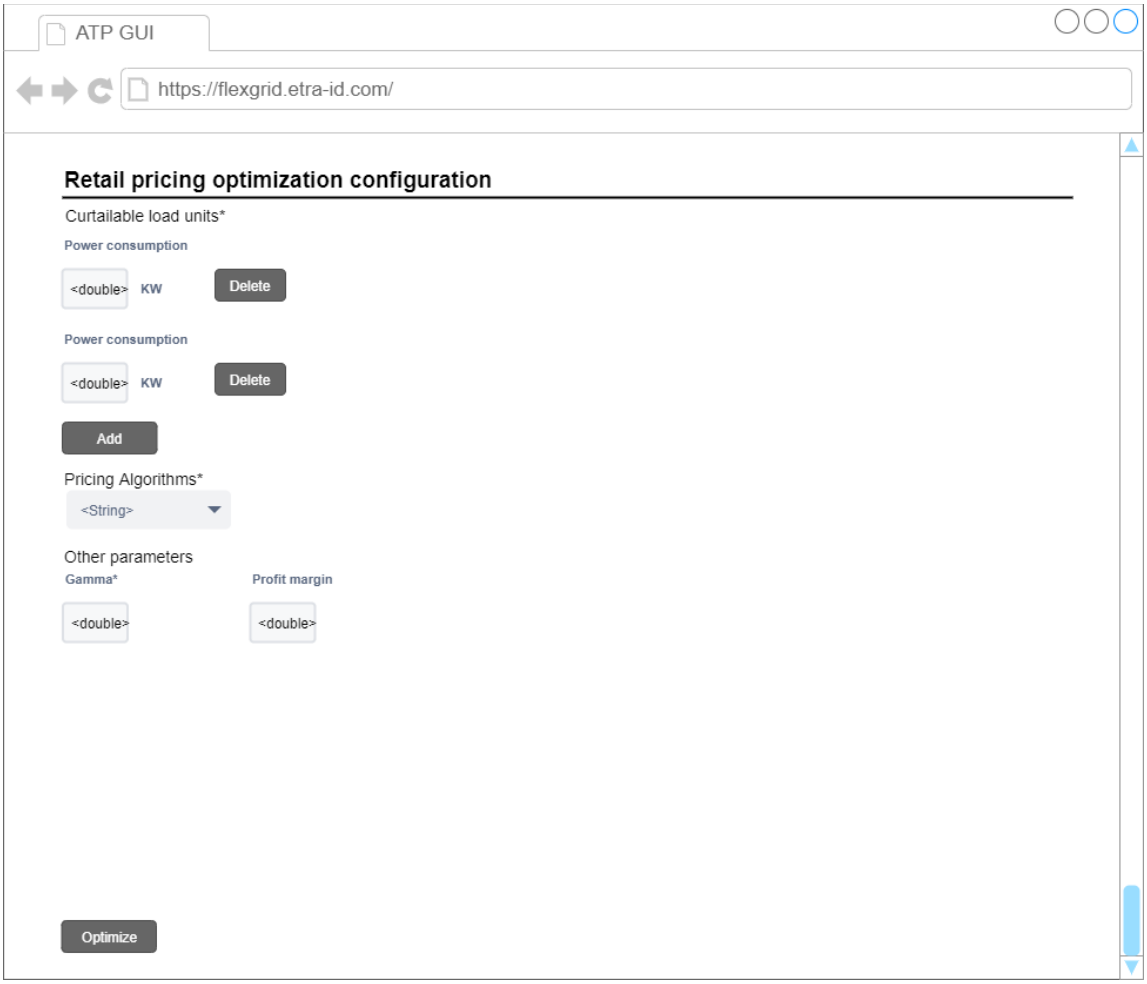


Figure 50: UCS4.2 – Retail pricing optimization configuration 6/6

Finally the optimization will be done by running different algorithm methods (It is mandatory to select at least 1 algorithm):

- Fixed pricing
- Real-Time Pricing (γ=0)
- Behavioural RTP (γ=1)
- Behavioural RTP (γ=0.5)
- Behavioural RTP (γ=1.5)

Once all the settings have been configured, by clicking on the “Optimize” button the Retail pricing optimization process will be triggered. Once it has finished, the results will be presented on the screen Retail pricing optimization results.

Retail pricing optimization results

Users: Aggregator

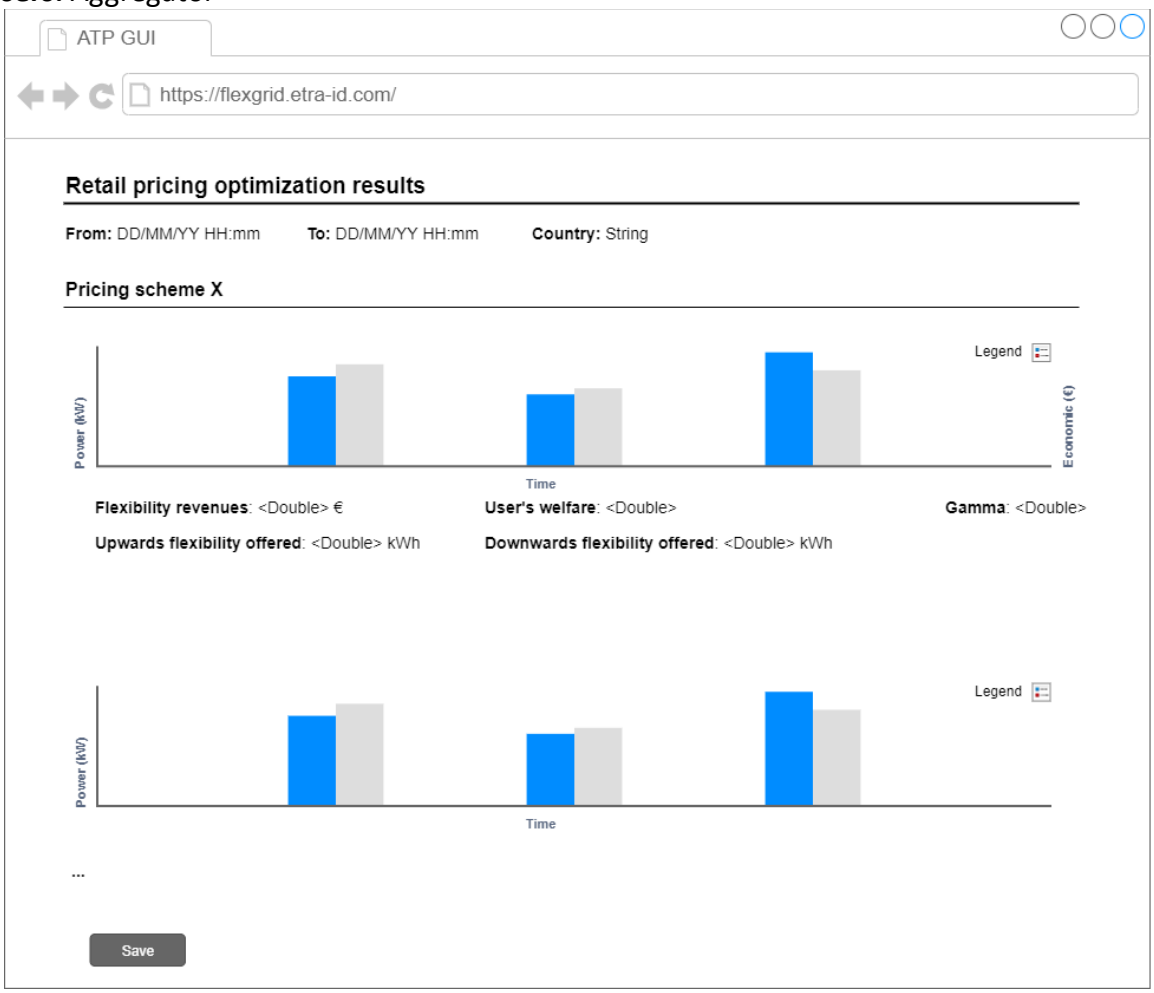


Figure 51: UCS4.2 – Retail pricing optimization results

Description: There are two possible ways the Aggregator user can access to this screen (Figure 51):

- By selecting one optimization from the list in the *Retail pricing Optimization* view
- From the *Retail pricing configuration* view, once the user triggers the optimization process and it has finished

The information presented here contains:

- The dates contained on this optimization
- The country this optimization was performed for

Below this general information, for each pricing algorithm selected it will be visualized:

- A bar chart containing part of the general results of the optimization. By clicking on a series name (the ones from both y-axis) on the legend, it will be possible to hide that series (or to put it back again):
- A histogram chart containing part of the general results of the optimization.

By clicking on the “Save” button, the optimization results will be stored on the Central DB.

3.3.9 UCS 4.3– Create a FlexOffer

Flexibility offer optimization historical view (with revenues)

Users: Aggregator
Description: All the stored optimizations for the actual month will be listed (Figure 52). It will also be possible to retrieve the stored optimizations for past operations (days, months…) by selecting a range of dates and clicking on the "Load" button.

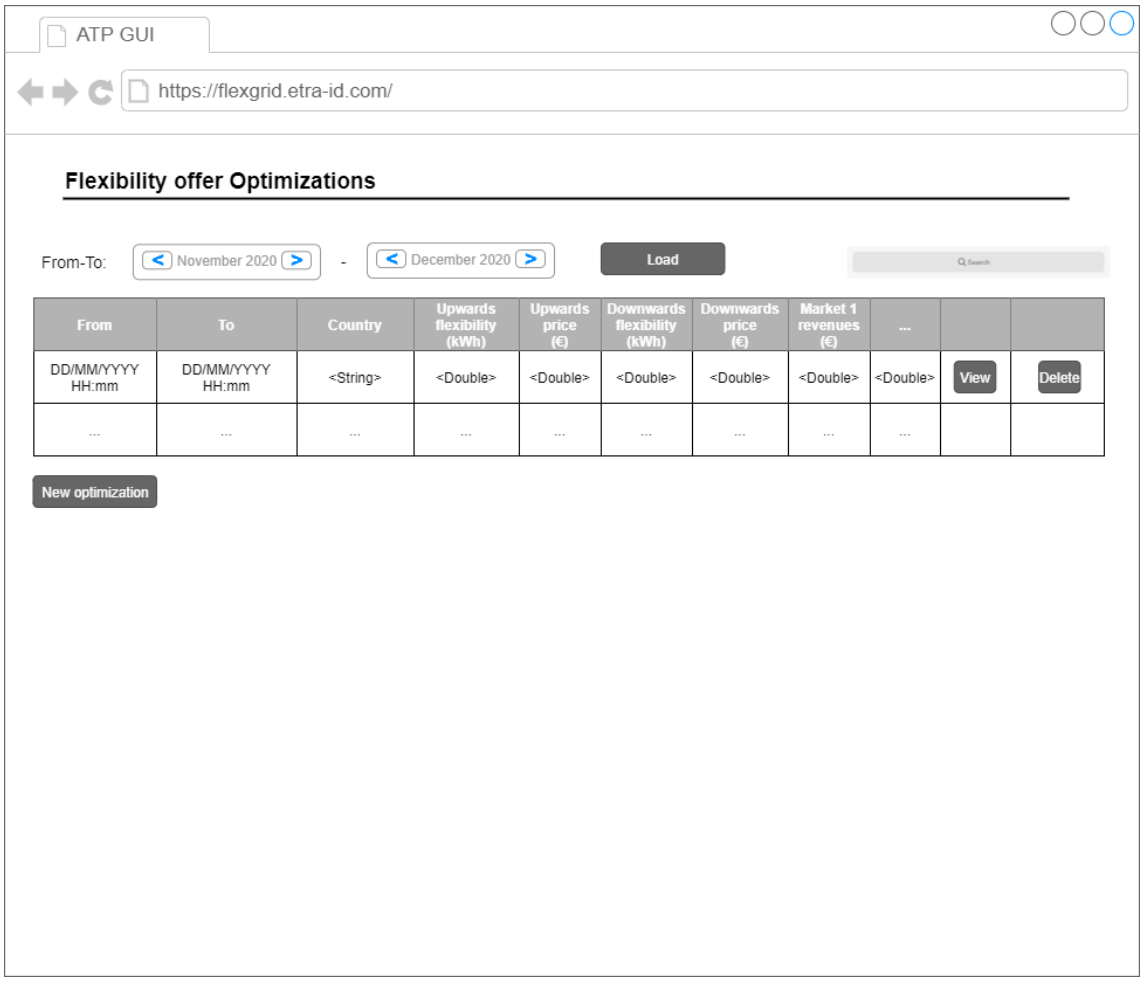


Figure 52: UCS4.3 – Flexibility offer optimization historical view (with revenues)

It will be presented basic information including:

- The duration of the optimization (**From, To**)
- The country the optimization was run for (**Country**)
- The total amount of flexibility in upwards and downwards regulations included in the offer
- The total price for up and down flexibility is included in the offer.
- The revenues this offer can get from the different markets (**Market 1…**). Measured in the currency of the country

Other functionalities available will be:

- To sort the table according to one selected column.
- To search for a concrete value by using the text box above the table.

By selecting different button defined it will be possible to:

- By clicking on the “view” button the screen with the details of the selected optimization will be opened.
- By clicking on the “delete” button the selected optimization will be removed.
- By clicking on the “New optimization” button a window for configuring a new optimization will be opened

Flexibility offer optimization historical view (without revenues)

Users: DSO, FMO
Description: This screen (Figure 53) will be almost the same as the previous one, with some differences due to the information here available can be showed by the DSO and FMO:

- On the table it won’t be listed the revenues data
- The “New optimization” button is not available here
- The “delete” button is not available here

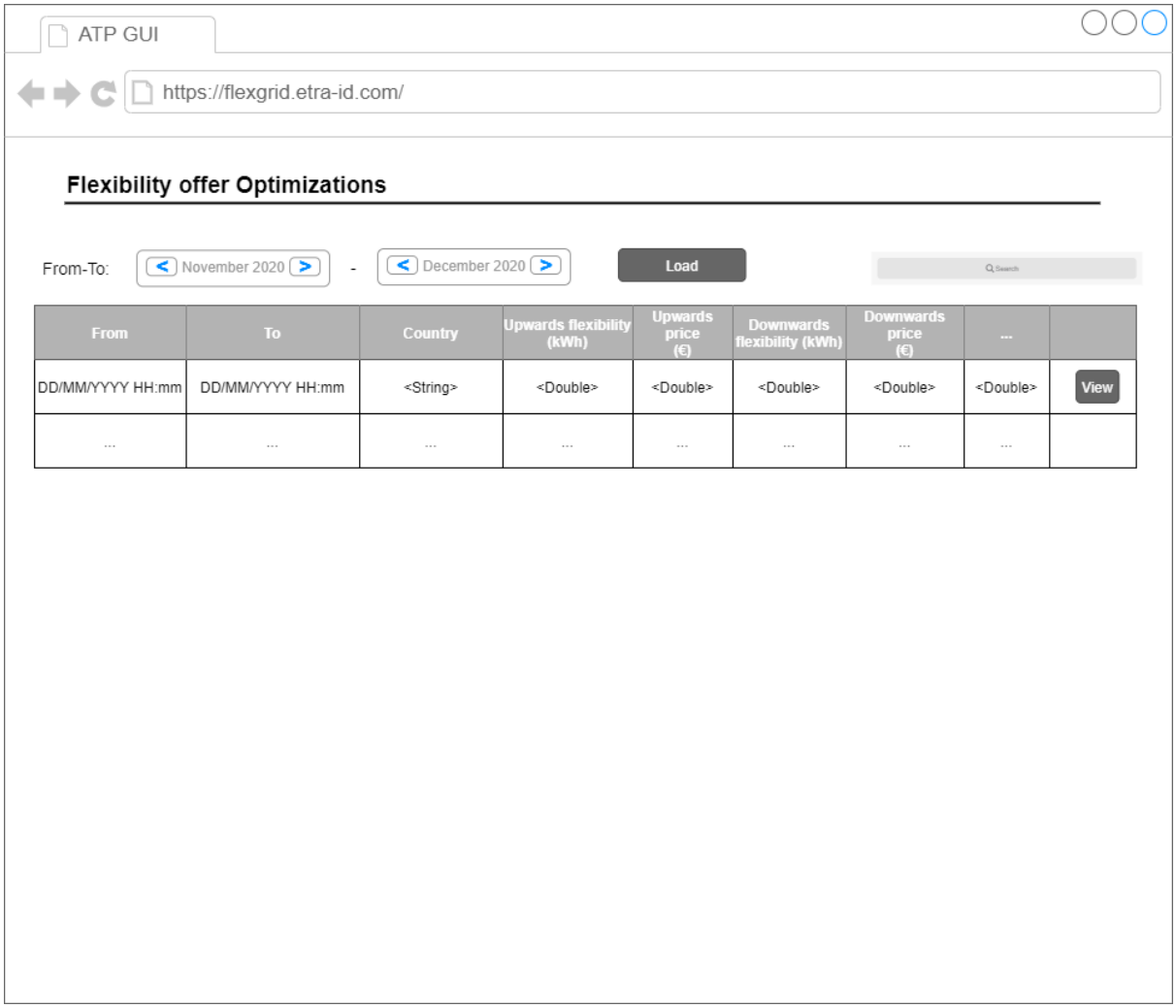


Figure 53: UCS4.3 – Flexibility offer optimization historical view (without revenues)

Flexibility offer optimization configuration

Users: Aggregator
Description: Only the aggregator is able to configure the algorithm for the optimization. The DSO and the FMO are able to view the results and relevant general information. The aggregator will configure the settings to run a new flexibility offer optimization as follows.

Firstly to introduce the next mandatory fields (Figure 54):

- *Country*: For which country this optimization is being performed. This field is mandatory
- *From*, *To*: The dates for the optimization. This field is mandatory
- *Granularity*: The granularity of the time series that is being used on this optimization. It can be 15-minutes, 1-hour, 1 day. This field is mandatory

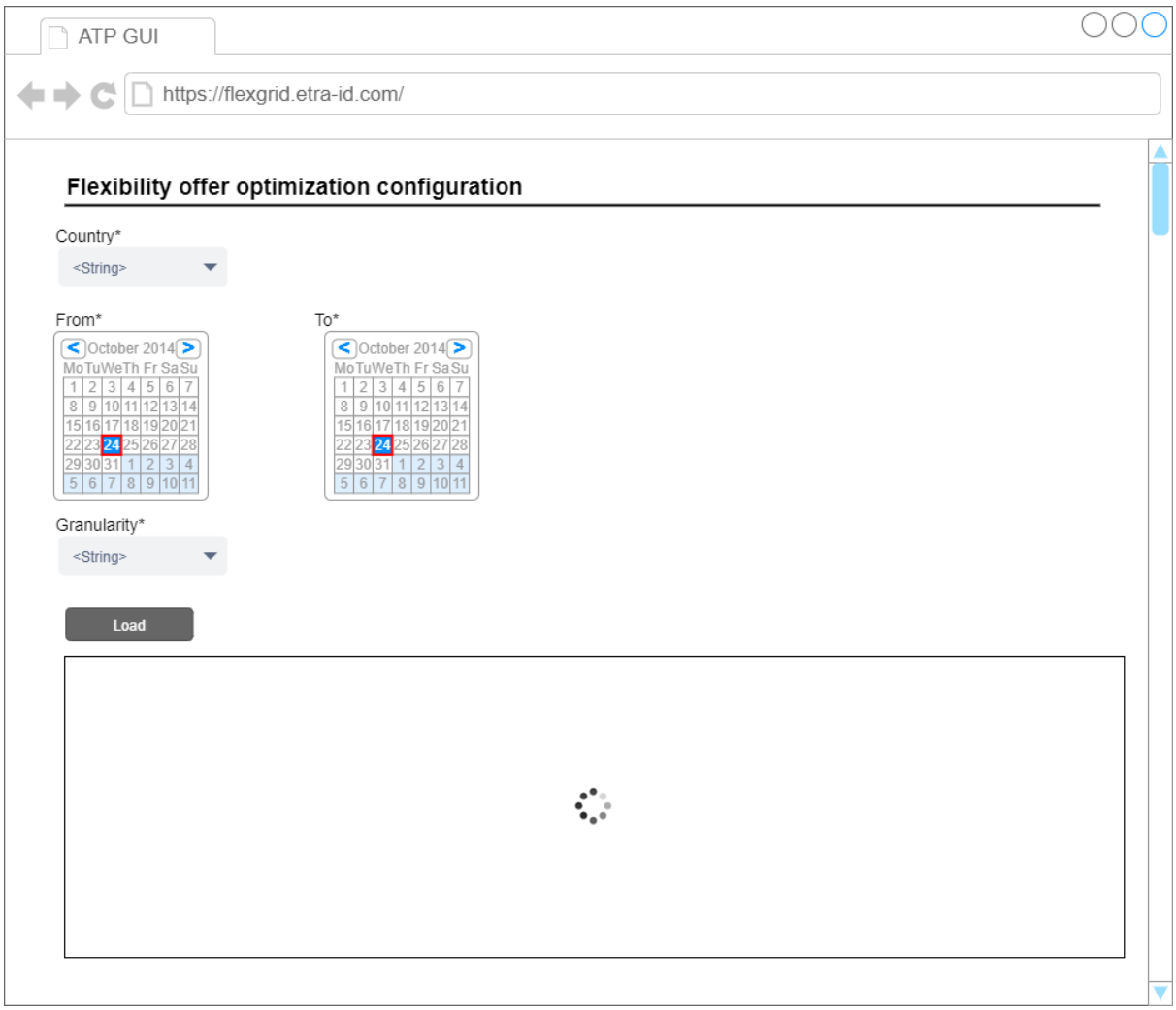


Figure 54: UCS4.3 – Flexibility offer optimization configuration 1/3

Once these 3 fields have been filled, by clicking on the "Load" button it will be presented on a map the different areas contained within the selected country (Figure 55). By default all of them will appear selected, but the user can deselect some of them by clicking on the different areas of the map.

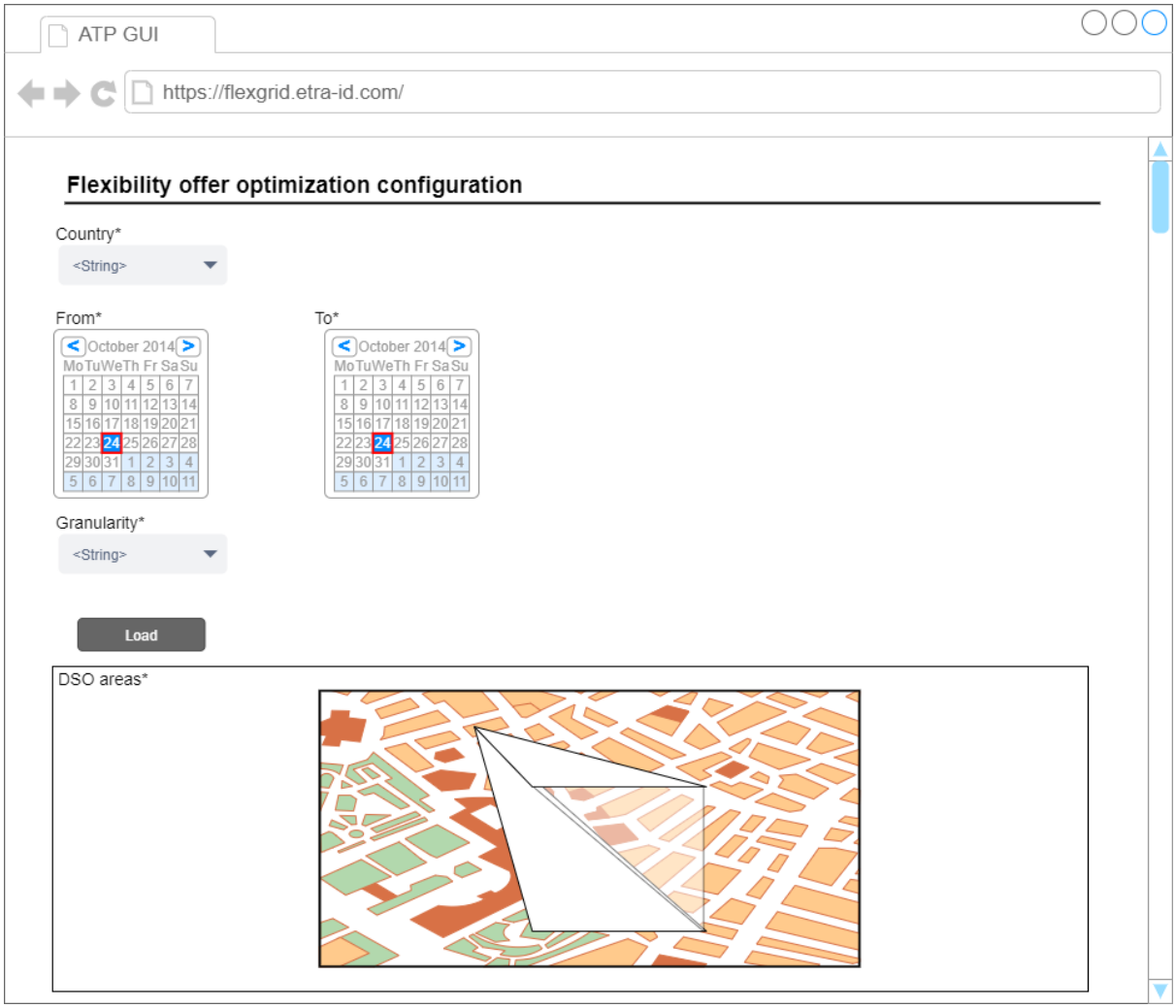


Figure 55: UCS4.3 – Flexibility offer optimization configuration 2/3

Finally, the *FlexAssets* from the portfolio to be considered in the optimization process have to be selected. In addition to the map, all users contained in the portfolio of the aggregator for the selected country will also be presented on a table (Figure 56).

By default, none of them will appear selected. By clicking on the "Select all" button all the FlexAssets will be selected; by clicking on the "Deselect all" button all the FlexAssets will be deselected; by checking-unchecking each checkbox that asset will be selected/deselected.

It will be possible to see the individual data of each FlexAsset by clicking on its row of the table (there is no need to check it). Once done this, it will be presented on the first chart its detailed information, containing:

- Baseline: The forecasted demand during the selected dates (measured in kW)
- Up/Down flexibility: The forecasted demand plus the available flexibility (up or down) during the selected dates (measured in kW)
- Up/Down revenue: The remuneration the user would receive for activating its flexibility (measured in the currency of the country)

On the second chart it is presented the information regarding the FlexAsset Optimization but aggregated during the entire time frame (measured in kWh). In each bar of this chart it will be stacked all the FlexAssets' data at each timestamp.

Once all the settings have been configured, by clicking on the "Optimize" button the optimization process will be triggered. Once it has finished, the results will be presented on the screen *Flexibility offer optimization results (with revenues)*.

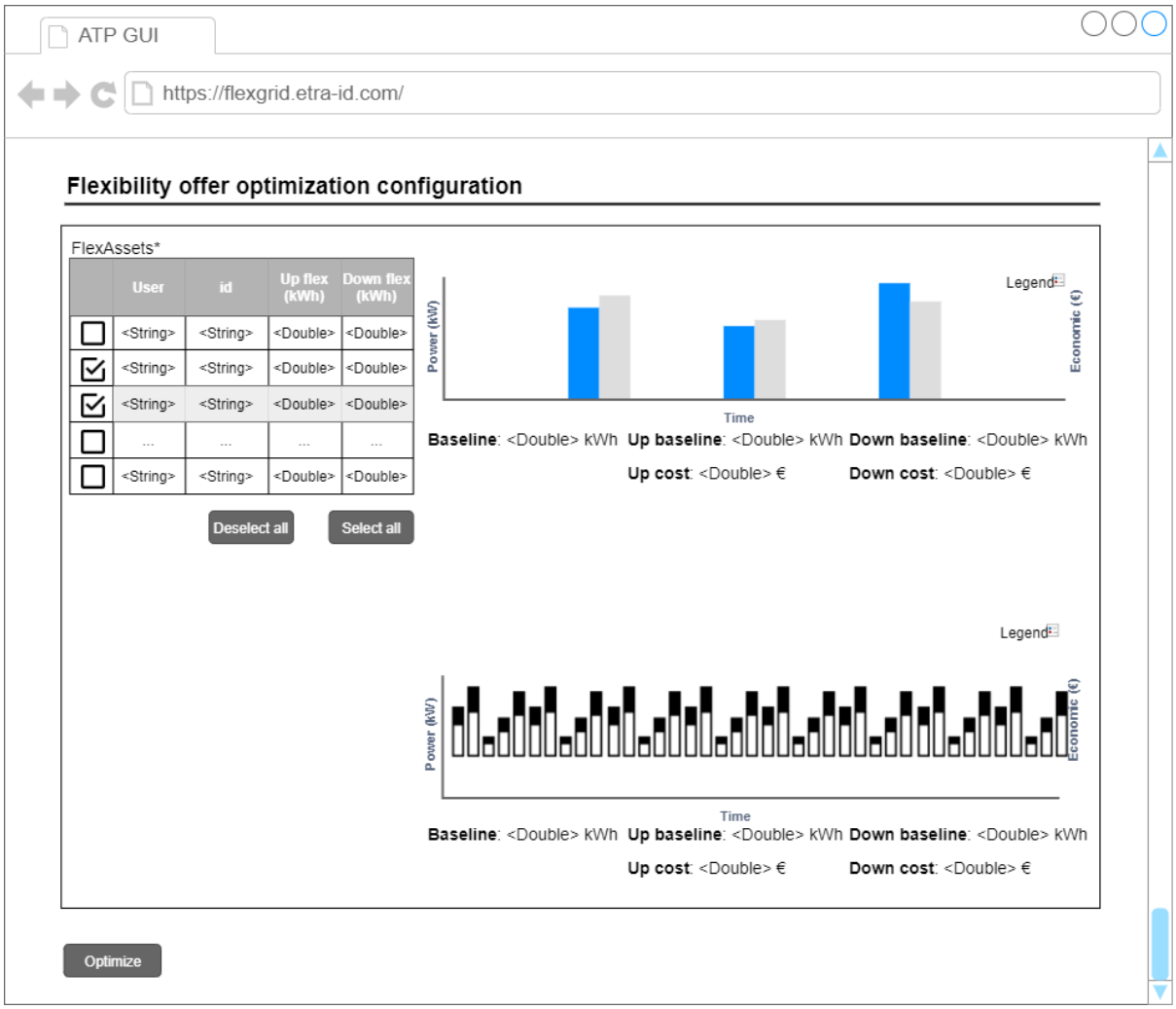


Figure 56: UCS4.3 – Flexibility offer optimization configuration 3/3

Flexibility offer optimization results (with revenues)

Users: Aggregator

Description: There are two possible ways the Aggregator user can access to this screen (Figure 57):

- By selecting one optimization from the list in the *Flexibility offer optimization historical view (with revenues)*
- From the *Flexibility offer optimization configuration* view, once the user triggers the optimization process and it has finished

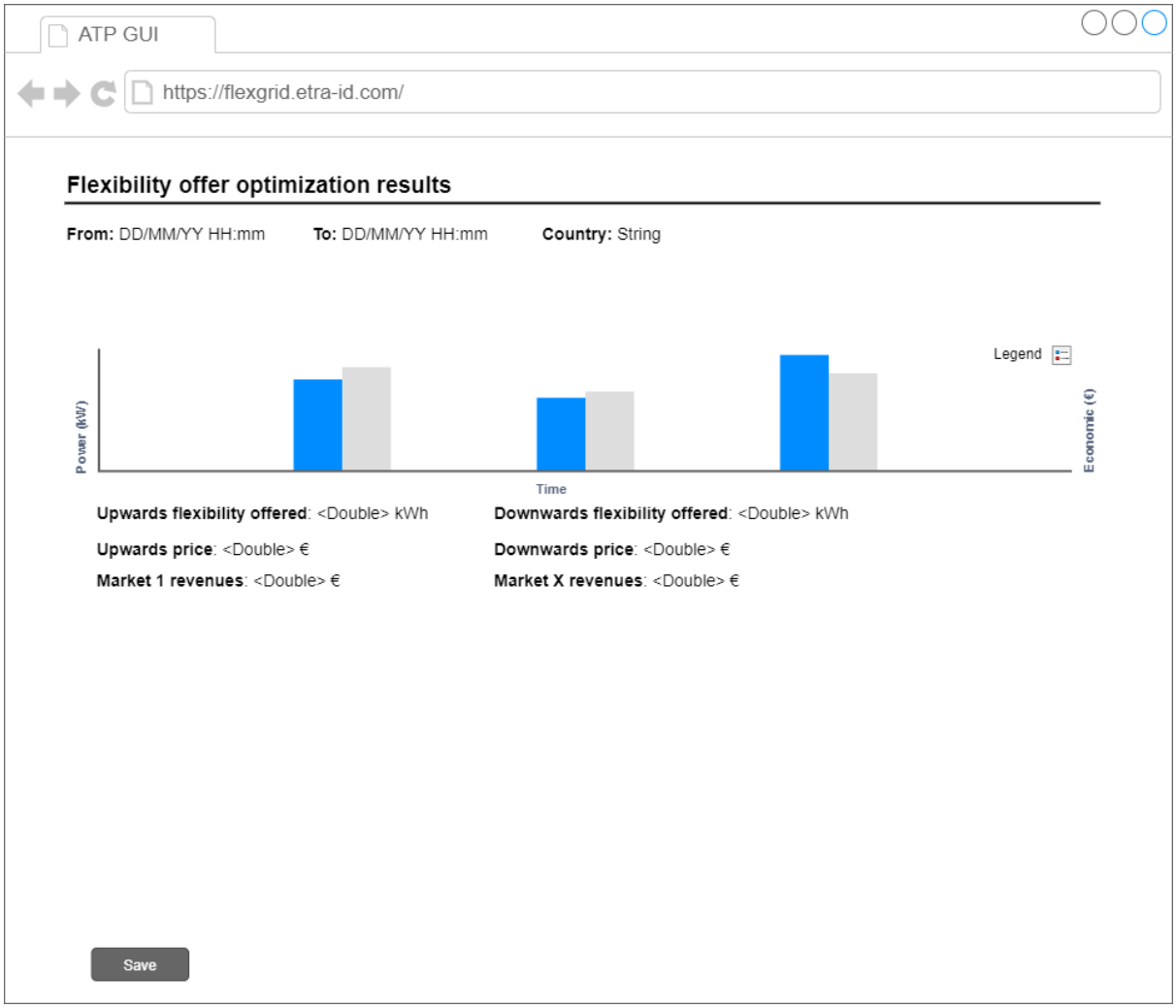


Figure 57: UCS4.3 – Flexibility offer optimization results (with revenues)

The information presented here contains:

- The dates contained on this optimization
- The country this optimization was performed for
- A unique bar chart containing the results of the optimization process. By clicking on a series name (the ones from both y-axis) on the legend, it will be possible to hide that series (or to put it back again)

By clicking on the "Save" button, the optimization results will be stored on the Central DB.

Flexibility offer optimization results (without revenues)

Users: DSO/FMO
Description: There is only one possible way the DSO/FMO user can access to this screen:

- By selecting one optimization from the list in the *Flexibility offer optimization historical view (without revenues)*

This screen will be almost the same as the previous one (Figure 57), with the following differences:

- No revenues information is presented
- No “Save” button is presented

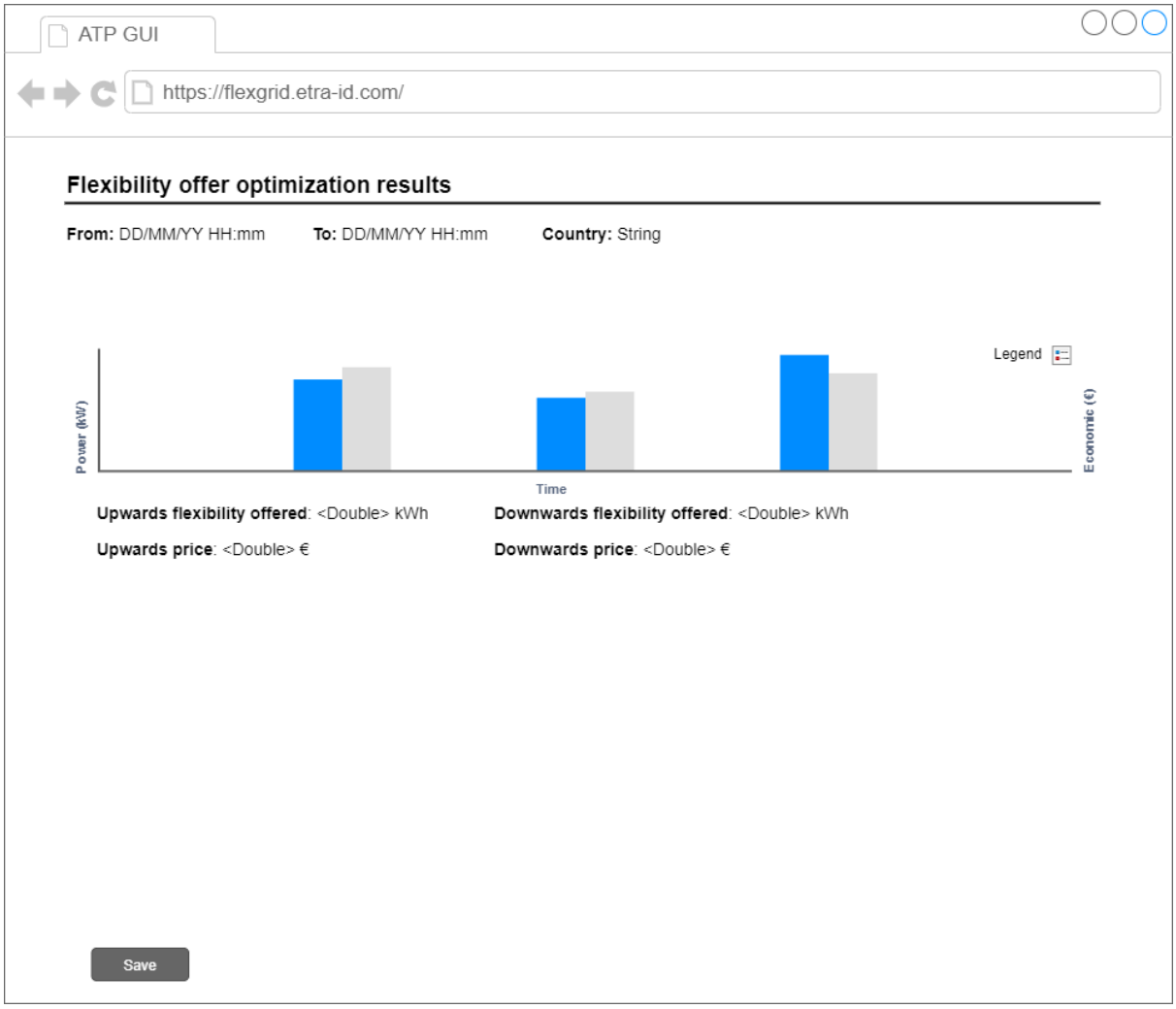


Figure 58: UCS4.3 – Flexibility offer optimization results (without revenues)

3.3.10 UCS 4.4– Market price forecasting

Market price forecasting historical view

Users: ESP

Description: All the stored forecast for the actual month will be listed (Figure 59). It will also be possible to retrieve the stored forecast for past operations (days, months…) by selecting a range of dates and clicking on the “Load” button

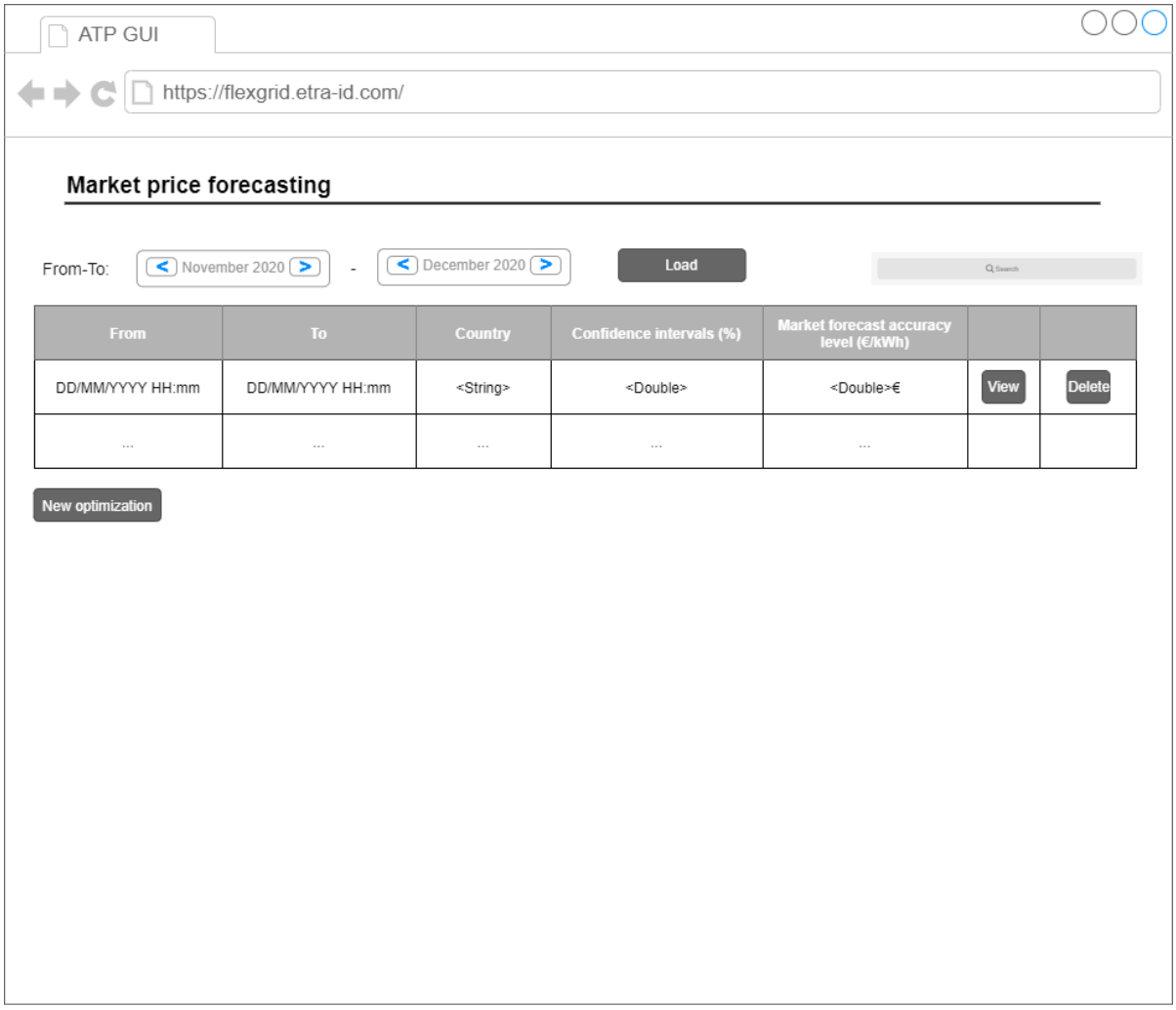


Figure 59: UCS4.4 - Market price forecasting historical view

It will be presented basic information including:

- The duration of the forecast (**From**, **To**)
- The country the forecast was run for (**Country**)
- The interval of prices (**Confidence intervals**) that the actual price is expected to be with a degree of confidence (measured in %)
- The difference between the forecast and actual price market (**Market forecast accuracy level**) (measured in €/MWh)

Other functionalities available will be:

- To sort the table according to one selected column.
- To search for a concrete value by using the text box above the table.

By selecting different button defined it will be possible to:

- By clicking on the “view” button the screen with the details of the selected forecast will be opened.
- By clicking on the “delete” button the selected forecast will be removed.
- By clicking on the “New forecast” button a window for configuring a new forecast will be opened.

Market price forecasting configuration

Users: ESP
Description: The user will configure the settings for running a new Market price forecasting following the steps below:

Firstly it has to be selected (Figure 60):

- *Country*: For which country this forecast is being performed. This field is mandatory
- *From*, *To*: The dates for the forecast. This field is mandatory
- *Granularity*: The granularity the user of the ATP needs for the calculation and results. This field is mandatory

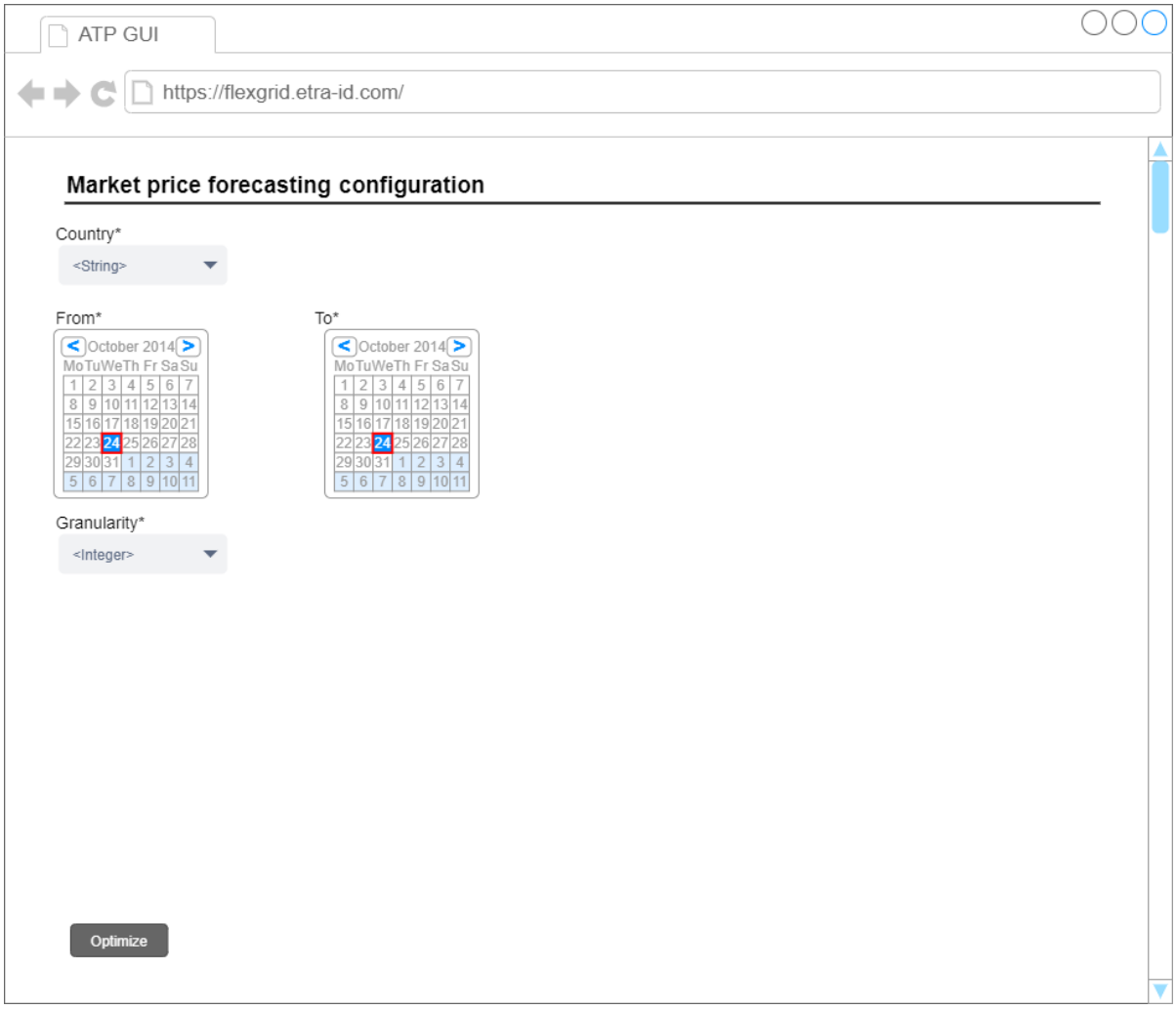


Figure 60: UCS4.4 – Market price forecasting configuration

Once all the settings have been configured, by clicking on the "Optimize" button the Market price forecasting process will be triggered. Once it has finished, the results will be presented on the screen *Market price forecasting results*.

Market price forecasting results

Users: ESP

Description: There are two possible ways the ESP user can access to this screen:

- By selecting one forecast from the list in the *Market price forecasting* view
- From the *Market price forecasting configuration* view, once the user initiates the forecasting process and it has finished

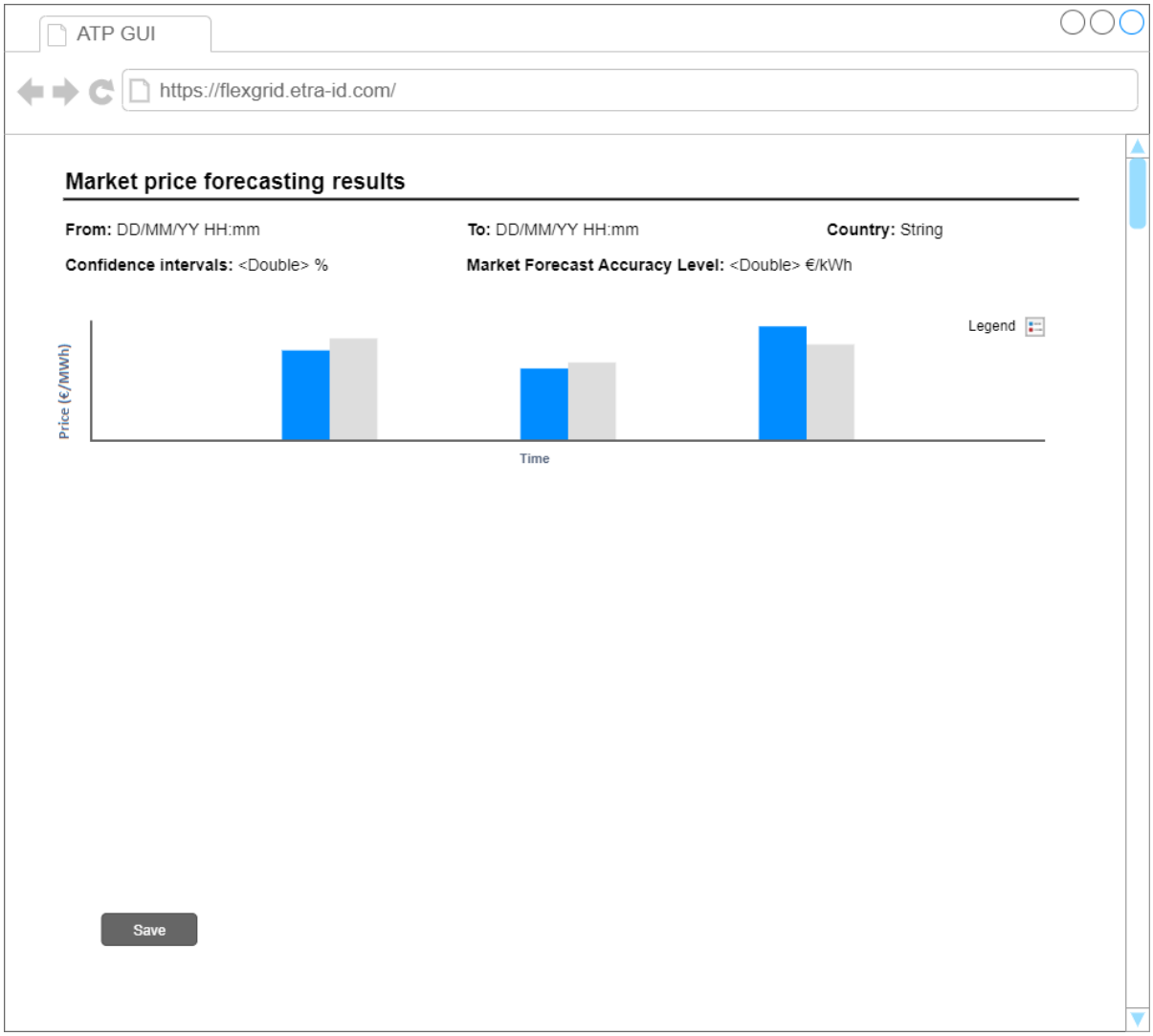


Figure 61: UCS4.4 – Market price forecasting results

The information presented here contains:

- The period covered by this forecast
- The country this forecast was performed for
- The price intervalwith an indication ofthe confidence level
- The difference between the forecast and actual price market
- A unique bar chart containing the results of the forecast. By clicking on a series name (the ones from both y-axis) on the legend, it will be possible to hide that series (or to put it back again):
    - x-axis: Time series
    - y-axis (left): Measured in €/MWh.
        - The price of the MWh

By clicking on the "Save" optimization, the forecast results will be stored on the Central DB.

3.3.11 UCS 4.4– PV generation forecasting

PV generation forecasting historical view

Users: ESP
Description: All the stored forecast for the actual month will be listed (Figure 62). It will also be possible to retrieve the stored forecast for past operations (days, months…) by selecting a range of dates and clicking on the "Load" button

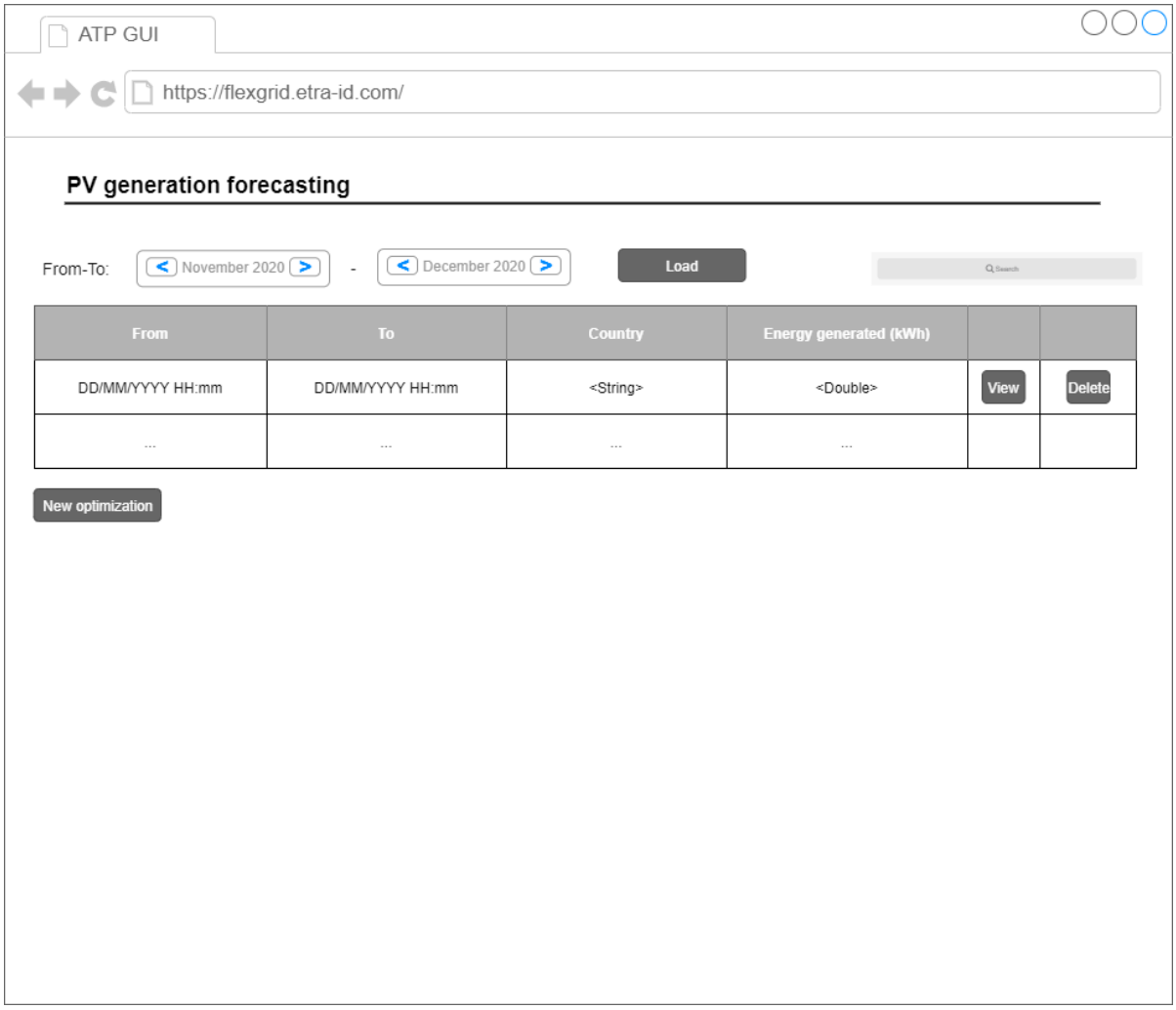


Figure 62: PV generation forecasting historical view

It will be presented basic information including:

- The duration of the forecast (**From**, **To**)
- The country the forecast was ran for (**Country**)
- The forecast of the total amount of energy to be generated (**Energy generated**) by the PV system (measured in kWh)

Other functionalities available will be:

- To sort the table according to one selected column.
- To search for a concrete value by using the text box above the table.

By selecting different button defined it will be possible to:

- By clicking on the “view” button the screen with the details of the selected forecast will be opened.
- By clicking on the “delete” button the selected forecast will be removed.
- By clicking on the “New forecast” button a window for configuring a new forecast will be opened.

PV generation forecasting configuration

Users: ESP

Description: The user will configure the settings to run a new PV generation forecasting process following the steps below:

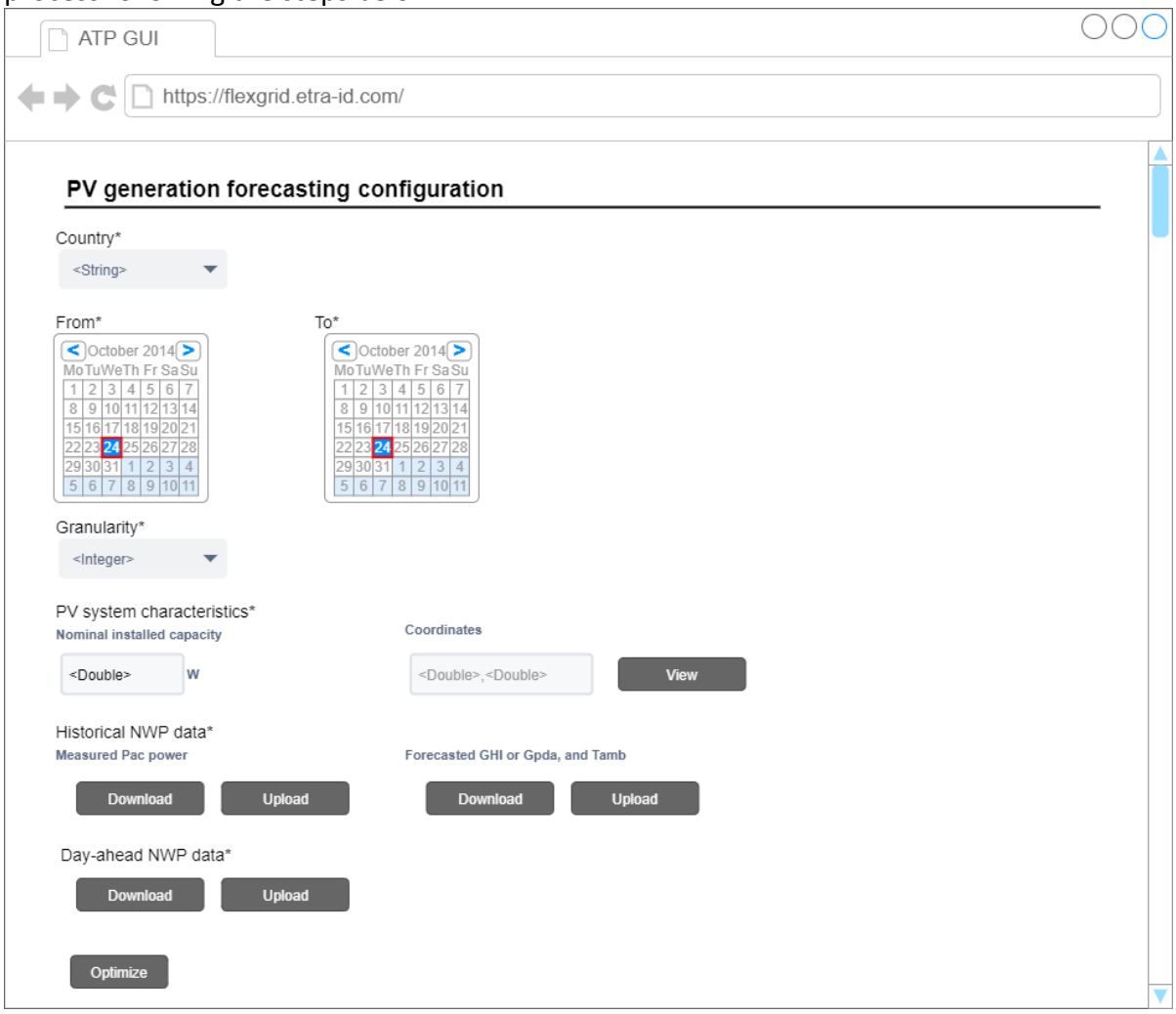


Figure 63: UCS4.4 – PV generation forecasting configuration 1/2

Firstly it has to be selected (Figure 63):

- *Country*: For which country this forecast is being performed. This field is mandatory
- *From*, *To*: The dates for the forecast. This field is mandatory
- *Granularity*: The granularity the user or the ATP needs for the calculations. This field is mandatory
- *Nominal installed capacity*: The maximum power capacity of the PV system to be forecasted

- *Coordinates:* Where the PV power plant is located. This information can be filled (Figure 64) by clicking on the "View" button, following two possible approaches. In both cases, the coordinates will be accepted by clicking on the "Accept" button:
    - Manually indicating them, by filling the "Latitude" and "Longitude" fields. Once the "Validate" button has been pressed, the map will be centred on these coordinates
    - Using the map. Clicking on it a new marker will appear on it (if there is already a marker on it, it will be moved to the new position). The coordinates' value of the marked point will be automatically filled on the textboxes.

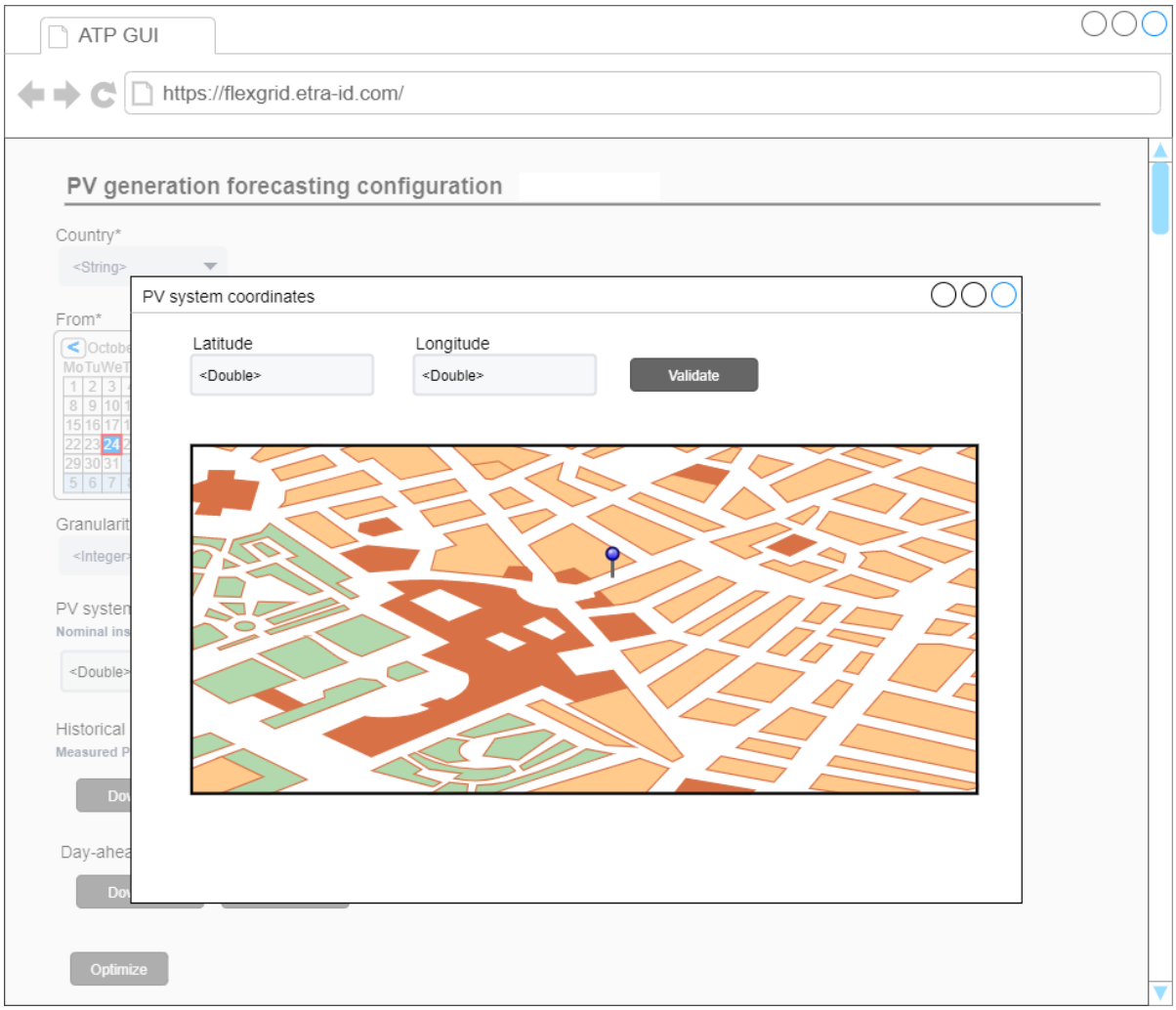


Figure 64: UCS4.4 – PV generation forecasting configuration 2/2

Finally, it has to be provided some external information:

- *Historical data*:
    - Historical measured power on the AC side of the PV (Pac).
    - Historical Numerical Weather Predictions (NWPs) or actual measurements: Global Horizontal Irradiance (GHI) and Ambient Temperature (Tamb).
- *Day-ahead NWP data*: NWP data per PV system location area

In both cases the steps to be followed are the same:

1. It has to be downloaded using the .csv template by clicking on the “Download” button.
2. Once it has been filled, it has to be uploaded back again into the system by clicking on the “Upload” button.

Once all the settings have been configured, by clicking on the “Optimize” button the PV generation forecasting process will be triggered. Once it has finished, the results will be presented on the screen *PV generation forecasting results*.

PV generation forecasting results

Users: ESP

Description: There are two possible ways the ESP user can access to this screen:

- By selecting one forecast from the list in the *Market price forecasting* view
- From the *Market price forecasting configuration* view, once the user initiates the forecast process and it has finished

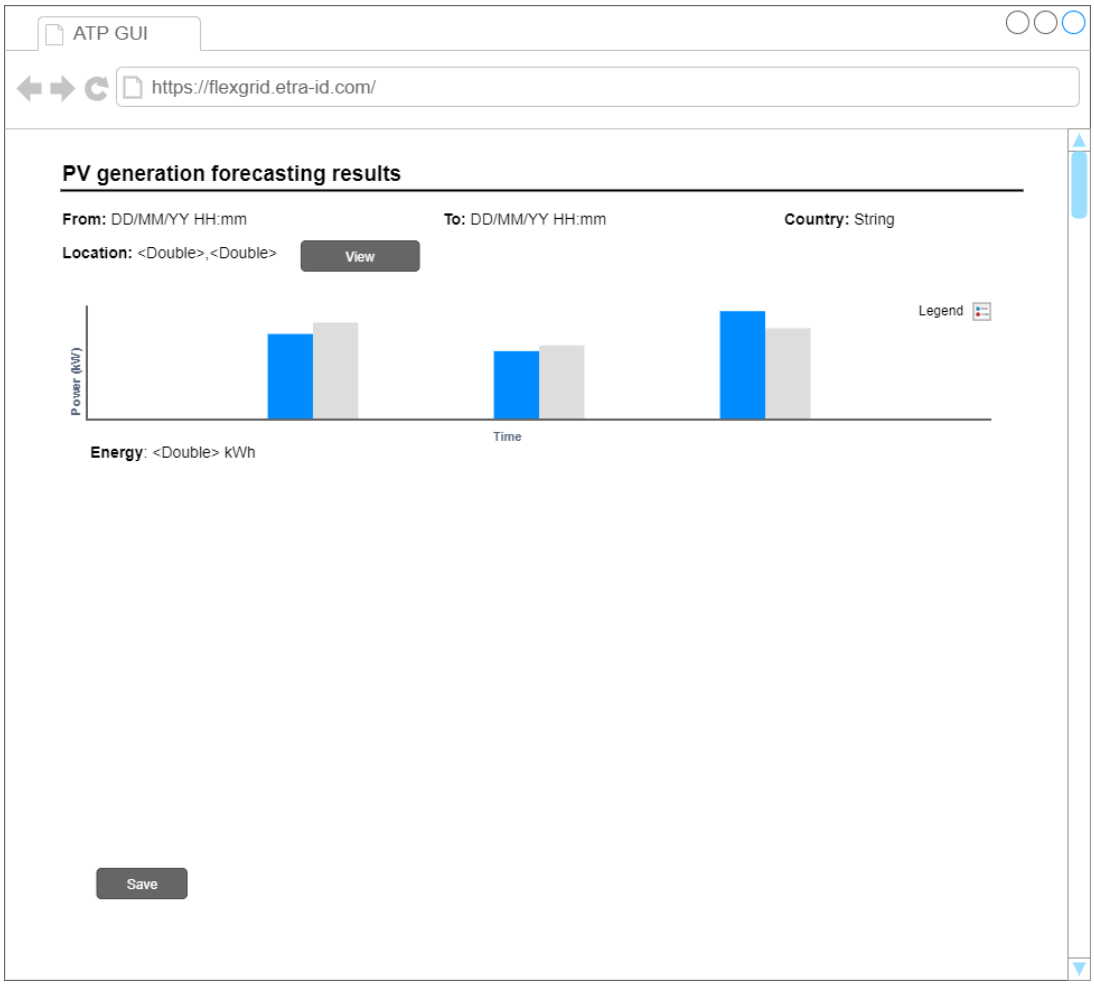


Figure 65: UCS4.4 – Market price forecasting results

The information presented here contains:

- The dates contained on this forecast
- The country this forecast was performed for

- The interval of prices that the actual price is expected with a degree of confidence to lie
- The difference between the forecast and actual price market
- A unique bar chart containing the results of the forecast.

By clicking on the “Save” button, the forecast results will be stored on the Central DB.

4 API Integration

4.1 Introduction

This section is related to Task 6.2 “Design of APIs and S/W Development”, whose main objective is the software development of the Application Programming Interfaces (API) that will facilitate the interaction among the modules developed within WP3, WP4, WP5 and the FLEXGRID system itself. For this purpose, different REpresentational State Transfer APIs (REST APIs) will be implemented for each module exposing their functionalities, so that everybody with credentials is able to use them.

Before the development phase of the APIs, a few agreements about the format of the inputs and the outputs have to be determined between the partner developers of the exposed components through the API and the partners using their services. A less technical and easily readable common methodology (explained in Section 0) documentation of APIs used in Swagger[2] is available in Section 2.3.

It has to be highlighted that **this API definition process is still ongoing**, so the API details about each UCS presented on this Deliverable may be different from the final one. The final version of the API will be presented in D6.3 “Final version of FLEXGRID S/W prototype” delivered in M33. The API integration and the S/W Development is an iterative process that will continue during the following months, but the work performed until now is documented here and will be the basis for the following discussions about the final version of the FLEXGRID API.

In Section 0 some snapshots with the actual content of the defined spreadsheets for each UCS have been included, except for UCS2.3, UCS4.2 and UCS4.3 APIs that can be directly provided into the .yaml file for the Swagger API documentation.

4.2 FLEXGRID SW architecture

As defined in D2.1, according to the requirements of each UCS the technical specification of FLEXGRID will be defined. For the SW architecture definition, it is important to recognize the different types of users that can use the FLEXGRID S/W platform. The user requirements defined in D2.1 for core and supplementary users are taken into consideration for defining the APIs structure. Furthermore, the three major subsystems defined in FLEXGRID project influence the architecture and the APIs` interaction and connection.

The Figure 66 shows the general scheme as the main definition of the SW structure as starting point for the implementation of the APIs.

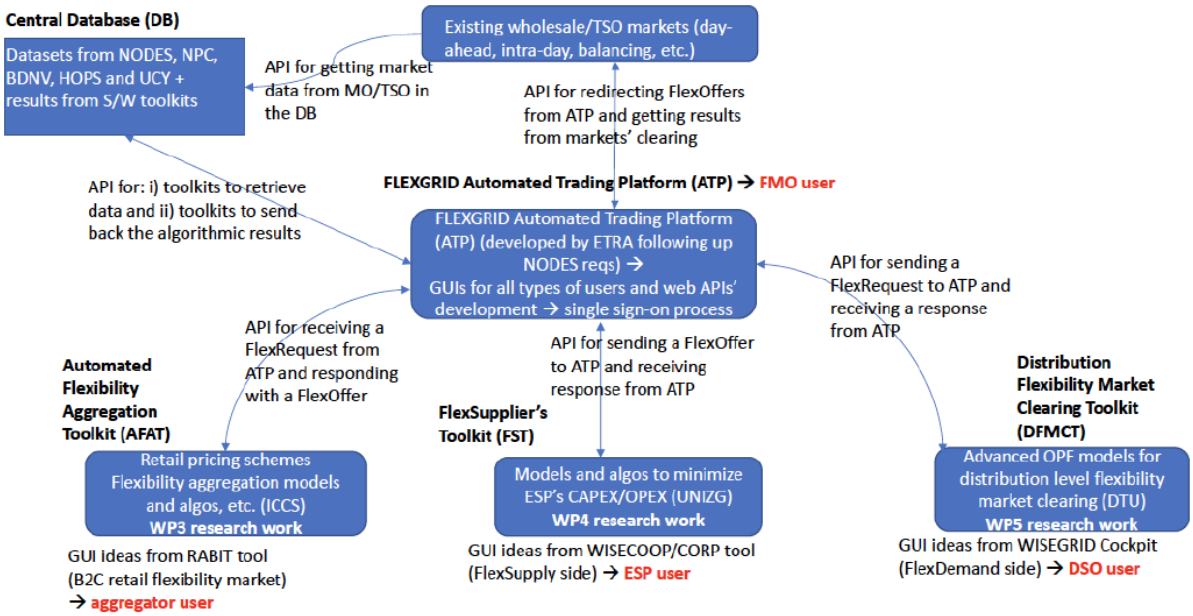


Figure 66 FLEXGRID S/W architecture design

The ATP platform shall provide a real time information and scalable trading platform for buyers and sellers of flexibility and shall interact with the other toolkits defined (AFAT, FST, FMCT) to provide the best output for every user and his specific needs. The modular-by design approach enables the integration and use of the different toolkits and modules as a whole or independently. The goal is to create an ATP that allows for a dynamic and effective use of distributed and local flexibility by interconnecting different services and algorithms that are designed to find the best solution for a large variety of user-specific problems.

Based on the information defined in D2.2 (section 3 and 4) and the deliverables developed in WP3, WP4 and WP5, each module that will be integrated in the ATP is linked with a specific UCS to be tested:

- AFAT is linked with the UCS 4.1, UCS 4.2, UCS 4.3 and UCS 4.4.
- FST is related with the UCS 2.1, UCS 2.2 and UCS 2.3.
- FMCT integrated the UCS 1.1, UCS 1.1 and UCS 1.3.

4.3 Use Cases Scenarios integration

As mentioned in the introduction, instead of directly define the .yaml files for documenting Swagger APIs, a less technical methodology has been defined. This methodology uses spreadsheet files to fill in all details of the required inputs and produced outputs by each exposed service. Afterwards these files will be converted into the aforementioned .yaml files.

On each one of the spreadsheets (1 file for each exposed service), each module owner has to fill in the following information in different tabs:

- General info
- Services
- Inputs
- Outputs

General info
Some general info about this service, including: a short name for the service, a service description, its actual version, the person in charge, and the type of license.

Services
Some initial details about the service exposed, including:

- Name: A "one word name" to describe the operation; something like "optimization", "calculateRevenues", "makeFlexOffer".
- Summary: A short explanation about what this service is about
- Type: The type of HTTP operation this service will expose. Its possible values are GET (for read operations), DELETE (for dele operations), PATCH (for update operations), POST (for create operations), PUT (also for update operations). In general, in all the UCS this type will be POST as the exposed services are the different algorithm developed within each WP that have to be triggered for generating results.
- Inputs: This is the link with the "inputs tab": Even if the algorithm needs several inputs, all of them can be grouped into one root element as documented here
    - Name: A short name for this variable
    - Array: At this point the only thing to be indicated is if that input is a single element or a vector: "yes" or "no".
    - minItems, maxItems: This has to be filled only if the input is a vector. Here it will be indicated the minimum and maximum number of items accepted on this variable, which can be from a minimum of 1 element (i.e if we support to run an optimization only for 1 hour) and an unspecified maximum
- Outputs: This is the link with the "outputs tab": The same explanation as with the inputs applies here but related to the results of the service.

Inputs
Before going on with the explanation of the Inputs and Outputs tabs, a short introduction about what kind of complex data type is needed. A complex type contains more than 1 unique attribute. For instance, let's imagine the definition of a person, so that could be something like:

- Person
    - Age: <number>
    - Size: <number>
    - Job: <text>
    - Body
        - Left arm
            - Shoulder
            - Elbow
            - Hand
                - Finger1
                - Finger2
                - …

This hierarchical definition can be read as: A person has some "simple" attributes as his/her age, height, job… and he/she has a body, which also contains some properties that are also complex.

In this section of the spreadsheet all the details about the attributes (also named variables) of the input of the service are depicted, including:

- Name: A short name for this variable. Once the "type" column has been explained, it will be understood why in this section several Name rows can appear
- Properties: Type of properties, that can be complex or simple.
- Required: If this property is required or it can be omitted in some cases. For example, going back to the "person" example, the Job is something that is not needed for defining how a person is, so it can be considered a non-required property.
- Array: The same as commented before. For example, "hand" will be an array of type "Finger".
- minItems, maxItems: The same as commented in the **services** explanation. Then continuing with the Hand array, it will contain a maximum of 5 fingers, and a minimum of 0.
- Type: It can be a basic type as integer number (integer), floating point number (double), text (string), true or false (Boolean), document (file)… or another complex type (in this case a custom type of name has to be filled here). The final objective of this classification is to have the simplest type and know all the possible types involved in the API.
- Value restrictions: Possible restrictions of the variable
    - Format: If it is a date, an email, a telephone… the string type is quite generic, so for some variables some restrictions should be defined.
    - Minimum, Maximum: If the variable is a number (integer or double), it may have some up and down boundaries. For example, a number for depicting a percentage can be between 0 and 1.
    - Accepted values: This is another kind of restriction. For example, for the currency, which is a string, it could only be accepted the following values: € and $
    - Example: An example with 1 possible value for this variable, but only if it is a simple one or an array of simple ones.
    - Comments: Just in case something else needs to be indicated.

For the outputs the same explanation and structure as with the inputs is followed considering that case the results of the service.

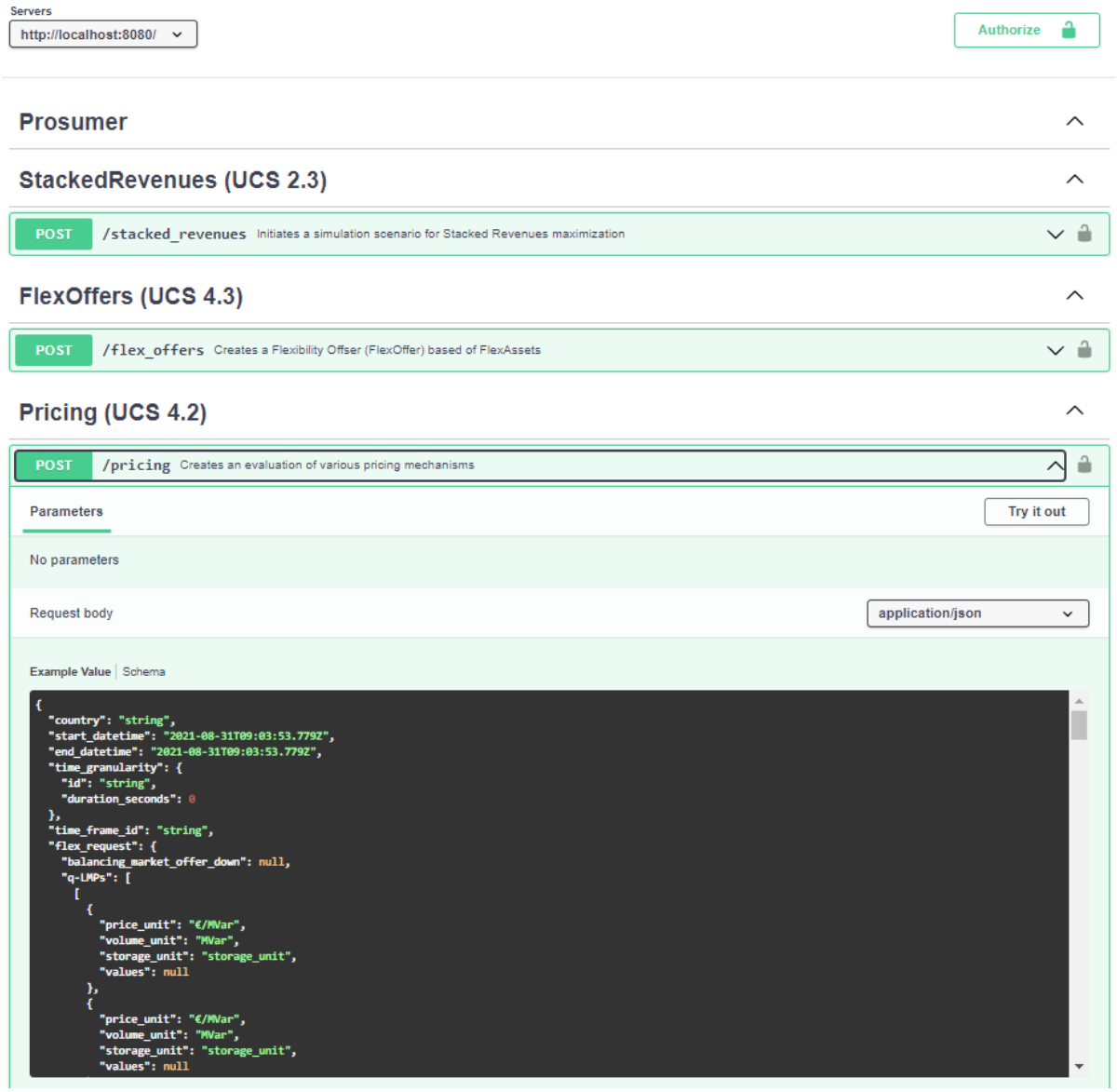
4.4 API Swagger prototype

As defined in D2.2 the application design is based on a REST architecture approach. REST defines six architectural constraints, which make any web service a true RESTful API. For the FLEXGRID ATP the following services are going to be used to fully cover all needs:

- POST: creates a new resource at the specified URI. The body of the request message provides the details of the new resource. Note that POST can also be used to trigger operations that don't actually create resources.
- GET: retrieves a representation of the resource at the specified URI. The body of the response message contains the details of the requested resource.
- PUT (*Update/Replace:*): either creates or replaces the resource at the specified URI. The body of the request message specifies the resource to be created or updated.

- PATCH (*Update/Modify*): performs a partial update of a resource. The request body specifies the set of changes to apply to the resource.
- DELETE (*Delete*): removes the resource at the specified URI.

Currently the service implemented for the integration of the UCS 2.3 is only the POST one as it is showed in the following picture.



This service allows to represent the calculations of the algorithm in the ATP for the users of the platform to see the results. Once the other UCS will be implemented and the central Data Base will be operative the other services will be used to fully use all the options that the ATP and the FLEXGRID algorithms provide.

5 Indicative ATP prototype results and developer user's manual

5.1 Indicative FLEXGRID ATP GUIs from a FLEXGRID service operation

The first algorithm that has been fully integrated in the FLEXGRID ATP was UCS 2.3 algorithm. In UCS 2.3, we consider the business case in which a profit-seeker ESP owns a set of Battery Storage Units (BSUs) located at various nodes/areas of a Distribution Network (DN). The ESP participates in 4 markets:

i) Day-Ahead Energy Market (DAM) operated by the Market Operator (MO).
ii) Day-Ahead Reserve Market (RM) operated by the TSO.
iii) Day-Ahead Distribution Level Flexibility Market (DLFM) operated by the Flexibility Market Operator (FMO).
iv) Near-real-time Balancing Market (BM) operated by the TSO.

Below, we present some indicative screenshots from the FLEXGRID ATP in order to explain how an ESP user can exploit the FLEXGRID ATP services. As a first step, the ESP user fills in his/her credentials (i.e. username and password) logs in the FLEXGRID ATP (see Figure 67 below).

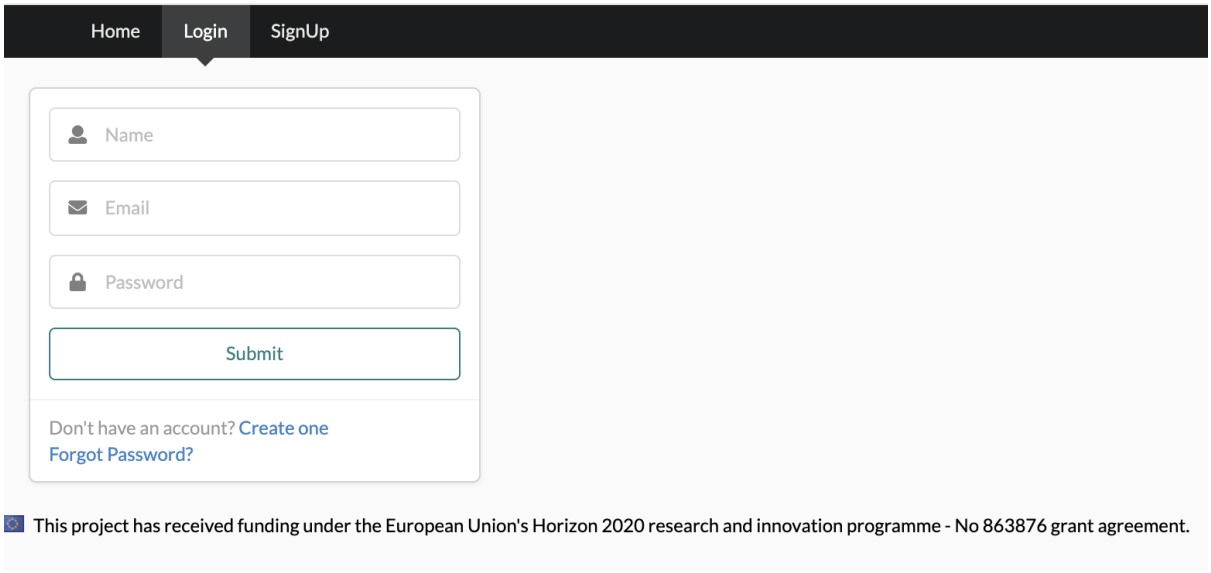


Figure 67: The ESP user fills in his/her credentials and logs in the FLEXGRID ATP

Via a global authentication process, the ESP user is redirected to the ESP GUI, where s/he can visualize his/her portfolio and select one of the 3 UCS of HLUC_02. The ESP user selects UCS 2.3 ("Maximize ESP's stacked revenues") and is directed to the main UCS 2.3 GUI.

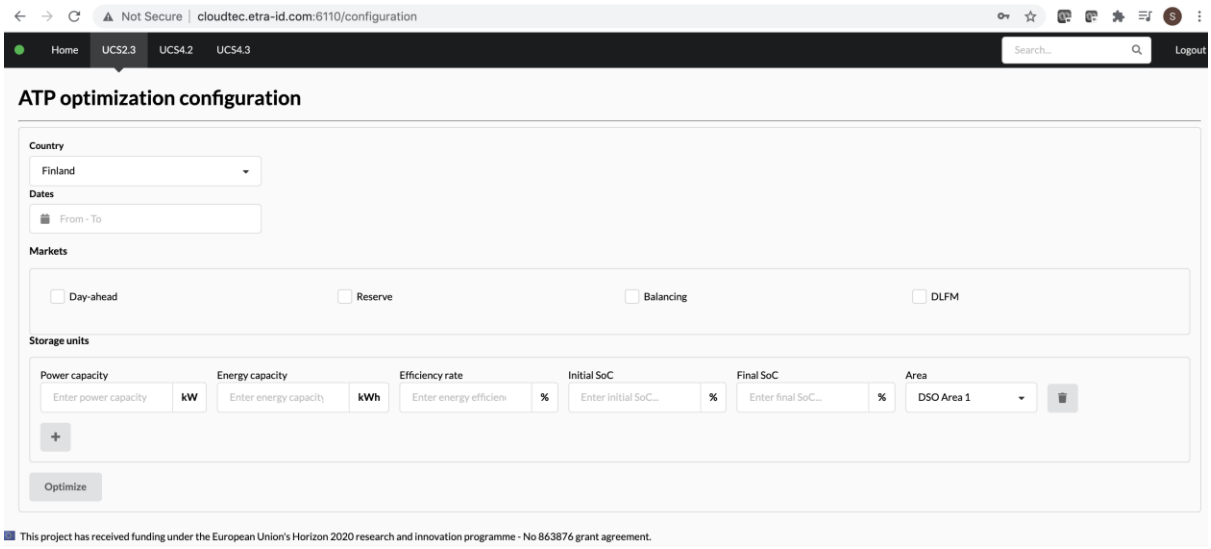


Figure 68: The ESP user selects UCS 2.3 and is ready to fill in the input parameters

Then, the ESP user fills in the input parameters in the GUI. In the example shown in the Figure 69 below, we assume that the ESP owns a Battery Storage Unit (BSU) that resides in Finland and wants to run a "what-if" simulation scenario for market participation on March 3rd 2021 in the day-ahead energy market. We assume that the BSU's power capacity is 100 kW and its energy capacity is 400 kWh. The BSU's efficiency rate is 99%, while the initial and final SoC rate is identical and equal to 50%. We also assume that the BSU resides at DSO area #1.

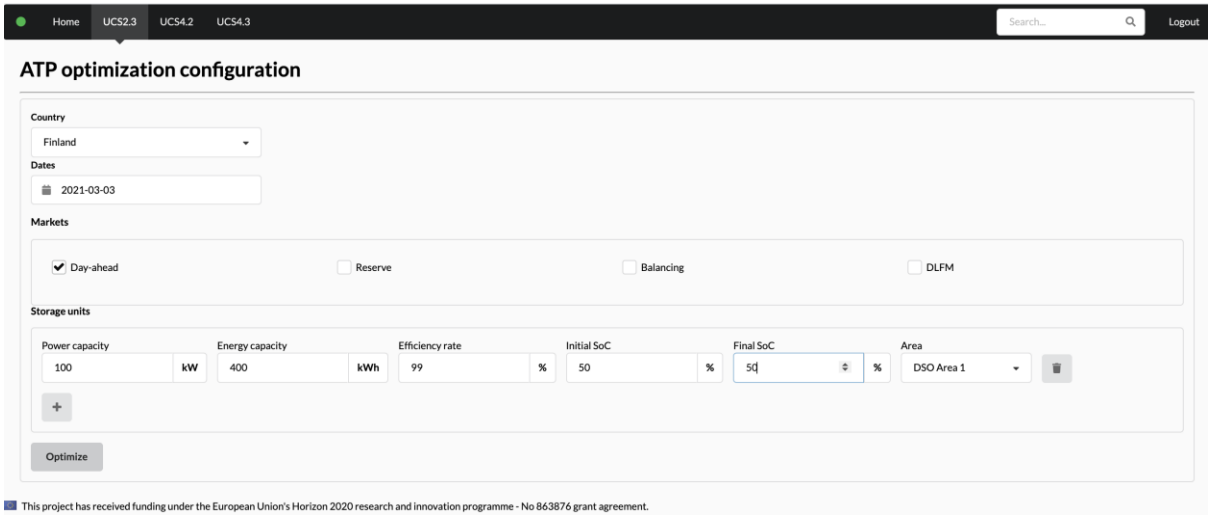


Figure 69: The ESP user fills in the input parameters and executes the UCS 2.3 algorithm (i.e. presses the "Optimize" button)

After pressing the "Optimize" button, the UCS 2.3 algorithm runs in the FST backend for a few seconds and the results are returned back to the ESP user as shown in Figure 70 and Figure 71. During the algorithm's run, the FST backend requests for the required day-ahead energy market data. This data is automatically retrieved from the Nord Pool market in real time, so that there is no need to store data in the FLEXGRID database. In case that there is no public API from a given Market Operator and/or country, the historical market price data can be easily stored and retrieved in the FLEXGRID database. In the figure below, the Energy and Flexibility Offers (one per hourly timeslot) are visualized. For example, for the 17:00-18:00

timeslot, the ESP's day-ahead energy market offer is 97 kWh, while there is no other offer in any other market. Right after this figure, the ESP's revenues are visualized for the given timeframe (i.e. one whole day, which is divided in 24 hourly timeslots). For instance, the ESP is expected to earn 13.74 euros for one single day by participating only in the day-ahead energy market operated by Nord Pool. As expected, the revenues from the other three markets are zero, because the ESP user did not select any other market during the simulation setup.

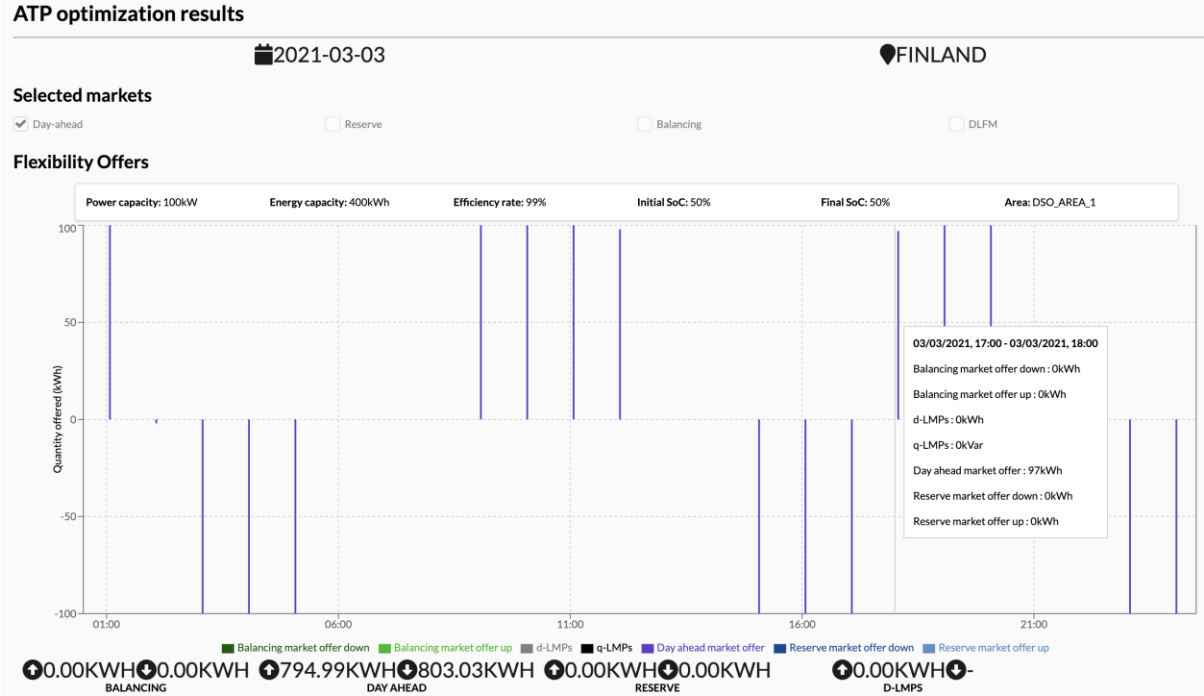
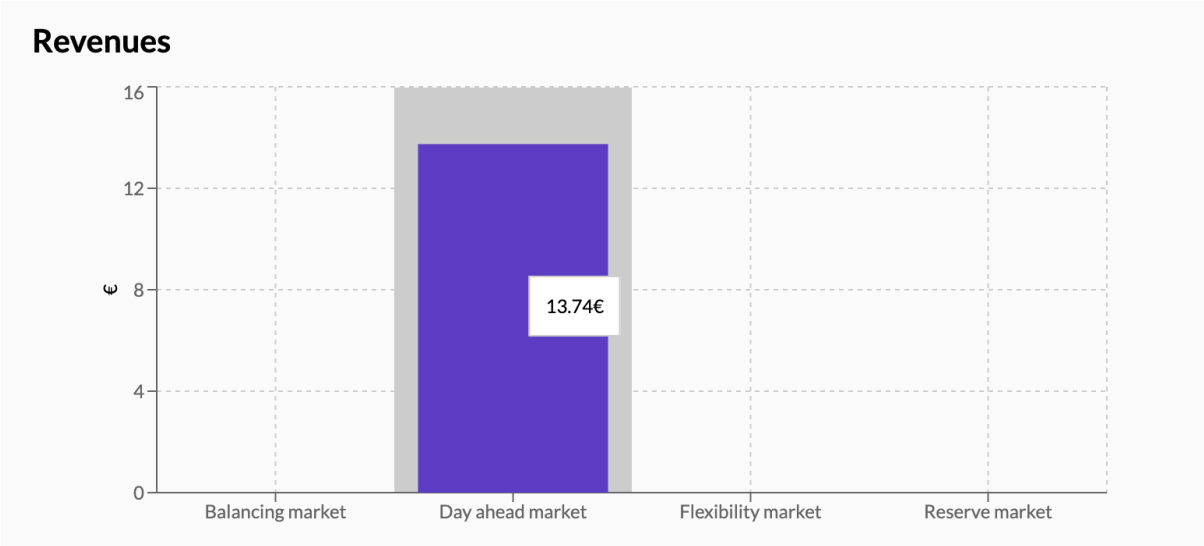


Figure 70: The Flexibility offers are returned back as results to the ESP GUI



This project has received funding under the European Union's Horizon 2020 research and innovation programme - No 863876 grant agreement.

Figure 71: Revenue results (in euros) are returned back to ESP's GUI

Assuming that the ESP user wants to run a “what-if” simulation scenario regarding its participation in all four available markets. Therefore, the ESP user can “tick” the respective four boxes as shown in Figure 72. From the drop-down menu “Area”, the “DSO Area 2” is selected, because DSO area 1 did not have a particular distribution network problem in this specific date. All the other input parameters are the same.

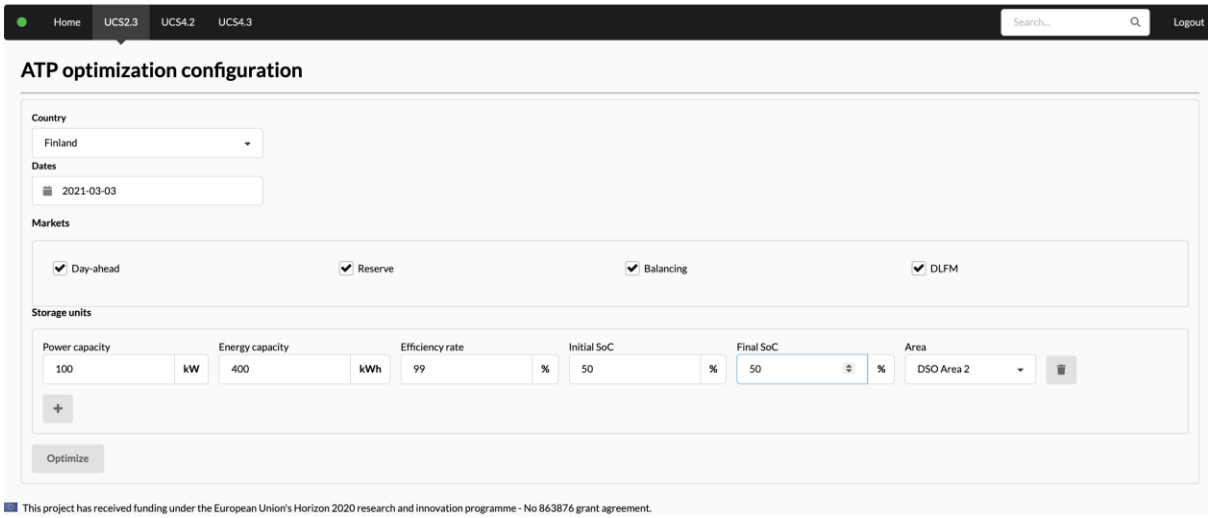


Figure 72: The ESP runs a simulation scenario assuming participation in all four available markets

After pressing the “Optimize” button, the UCS 2.3 algorithm runs at the FST backend, and the respective results are shown in the two figures below. The various bars with the different colours represent the energy/flexibility offers of the ESP for each hourly timeslot. For example, the two green bars represent the balancing energy up (i.e. light green) and balancing energy down (i.e. dark green) offers. For example, at the 16:00-17:00 timeslot, both offers are zero, which means that the stacked revenue maximization algorithm did not select balancing energy market participation for this given timeslot. On the other hand, the day-ahead energy market offer is -100 kWh (see purple bar), which means that the BSU wants to charge (i.e. buy) this amount of energy. At the same time, the reserve market up (light blue) offer is 200 KW for 16:00-17:00 (1 hour) and the reserve market down (dark blue) offer is zero. Finally, the active power reserve quantity offer (see grey bar for d-LMP) and the reactive power reserve offer (see black bar for q-LMP) is zero.

Figure 73: All the energy/flexibility offers are returned back as results to the ESP GUI

Figure 74 depicts the aggregated quantity values for all markets together with the aggregated revenues (in euros) per market. For example, the day-ahead energy market revenues are 30.69 euros (see purple bar), which are much more than the 13.74 euros from the previous simulation scenario. This is easily explained by the fact that the BSU can now participate in three more markets at the same time and thus the ESP has more opportunities (or else degrees of freedom) for arbitrage (i.e. revenue maximization). For instance, by providing active and reactive power reserves in the Distribution Level Flexibility Market (DLFM) especially in the morning hours during which the DSO requested for flexibility, the ESP earned 52.53 euros (see black bar). The blue bar represents the reserve market revenues (i.e. 16.48 euros) and the green bar represents the balancing energy market revenues (i.e. 26.57 euros). Much more and in-depth technical details and explanations about the UCS 2.3 algorithmic operation are provided in FLEXGRID D4.3.

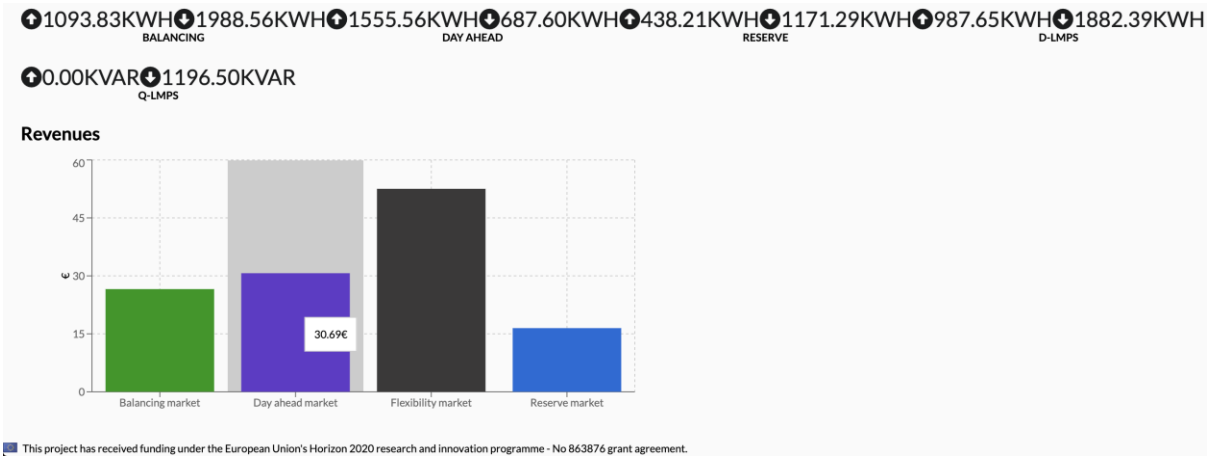


Figure 74: Revenue results (in euros) for each market together with aggregated quantity values per market are returned back to ESP's GUI

Following up the same process, the ESP user can run exhaustively many "what-if" simulation scenarios as follows:

- Run the UCS 2.3 algorithm for a different date or a bunch of days, weeks or months.
- Add more BSUs from the ESP's portfolio that may reside in different distribution network areas.
- Select a different set of input parameters and compare the respective results (when data from a new country is available, this data can be stored in FLEXGRID central database or a new API for automatically retrieving the historical market prices can be easily deployed by following the instructions in the developer user's manual).

All the revenue values presented in the figures above come up from the multiplication of the hourly quantity value per market times the respective hourly market price. It should be noted that these historical market prices are requested and retrieved on demand and in real time by the FST. More specifically, for the above-mentioned simulation scenarios, once the ESP user presses the "Optimize" button, the FST communicates with the Nord Pool trading API - https://www.nordpoolgroup.com/trading/api/ (for the day-ahead energy market) and the FINGRID public API - https://data.fingrid.fi/en/pages/apis (for the frequency containment reserve market and the balancing energy market). In the figures below, there are indicative APIs (json format) for automatically retrieving market price data from Nord Pool and FINGRID to FLEXGRID ATP. For example, for Nord Pool API, we can see the hourly market prices for March 3rd 2021 that are retrieved for the day-ahead energy market. For instance, in the first timeslot (i.e. 00:00-01:00), the market price is 25.06 euros/MWh, while in the second and third timeslot the prices are 24.35 and 24.84 euros/MWh respectively.

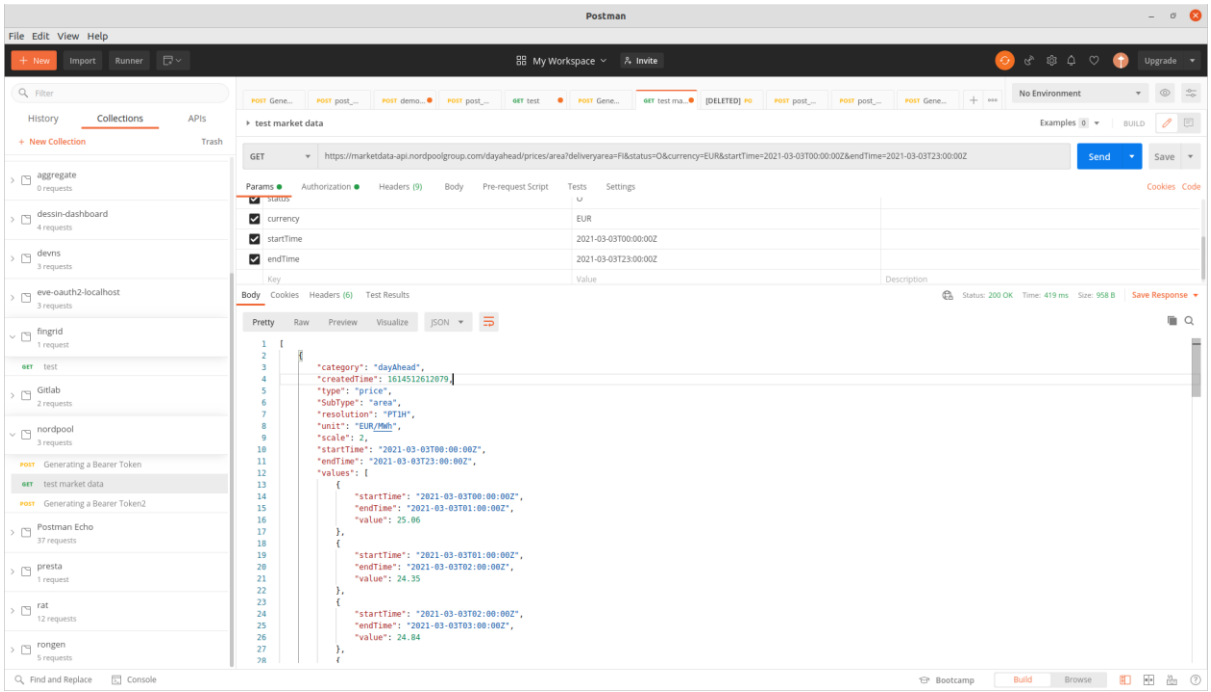


Figure 75: API (json format) for automatically retrieving day-ahead energy market price data from Nord Pool API to FLEXGRID ATP

In the same way, the two figures below depict the up- and down-regulation balancing market prices. For example, we can see that the up-regulation balancing market prices for the first six hourly timeslots are 25.06, 24.35, 24.84, 26.12, 39.11 and 42.08 respectively. As of the down-regulation balancing market prices for the first six hourly timeslots are 20.00, 20.00, 15.44, 15.44, 20.00 and 21.24 respectively.

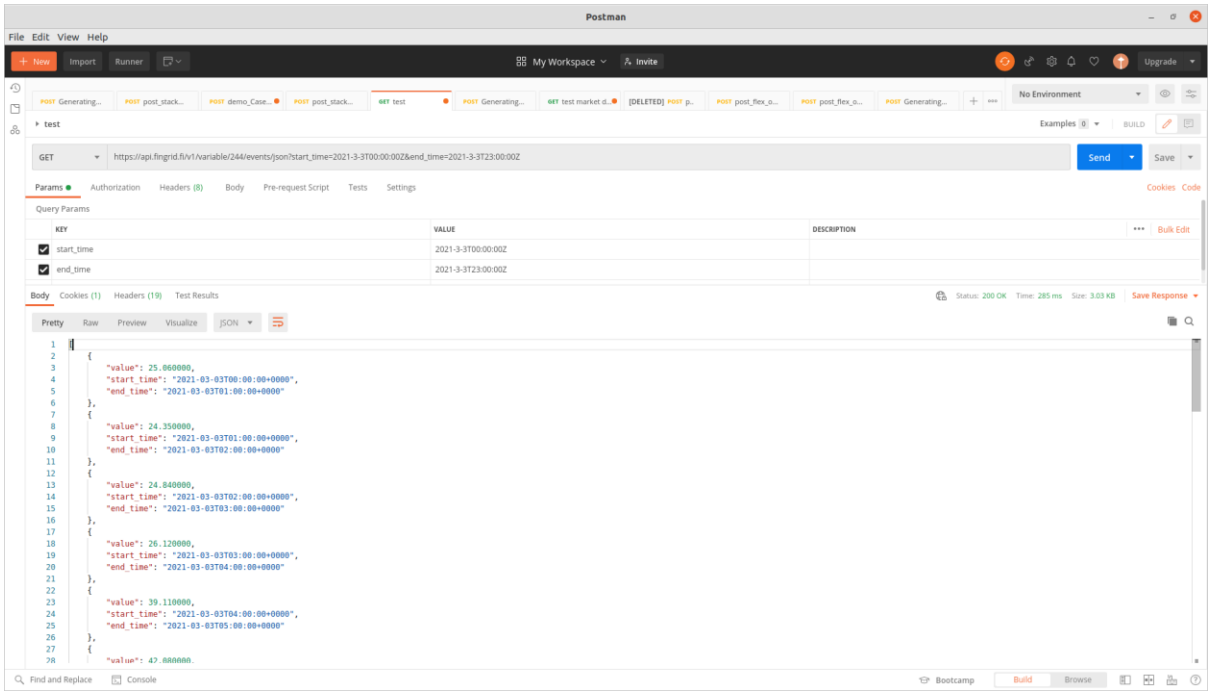


Figure 76: API (json format) for automatically retrieving up-regulation balancing market price data from FINGRID API to FLEXGRID ATP

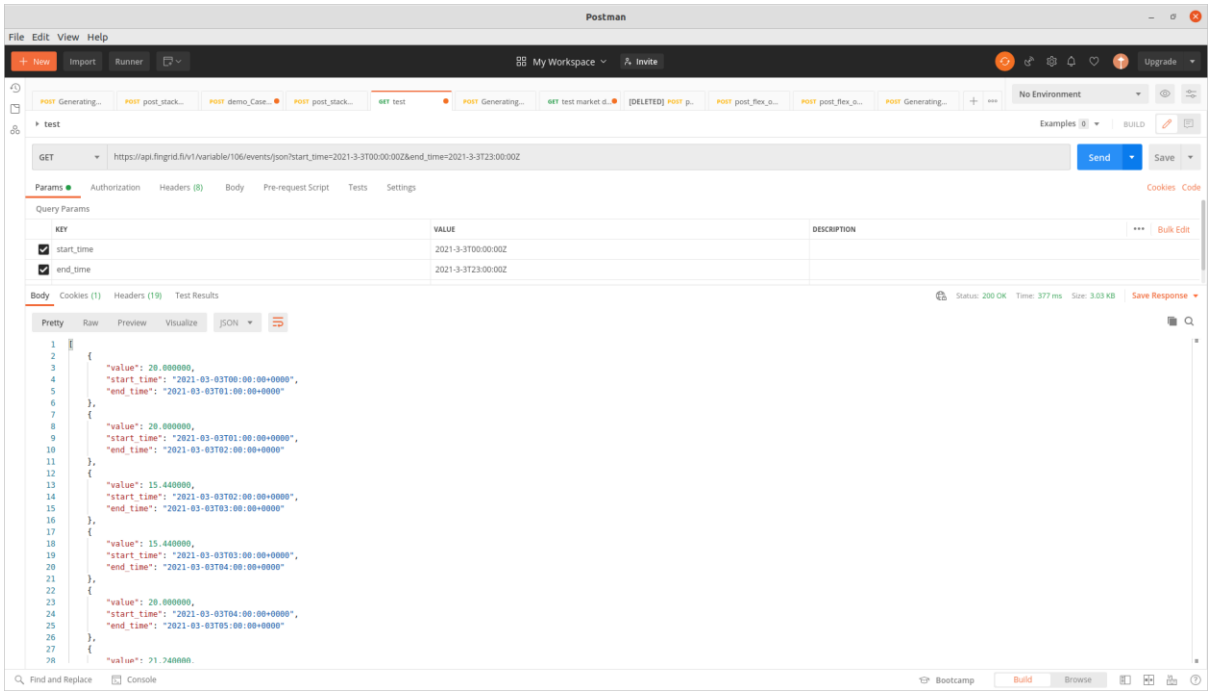


Figure 77: API (json format) for automatically retrieving down-regulation balancing market price data from FINGRID API to FLEXGRID ATP

Finally, in Figure 78, the frequency containment reserve market prices are depicted. For instance, for the first six hourly timeslots the prices are 14.00, 15.00, 15.00, 8.75, 6.50 and 8.00 respectively.

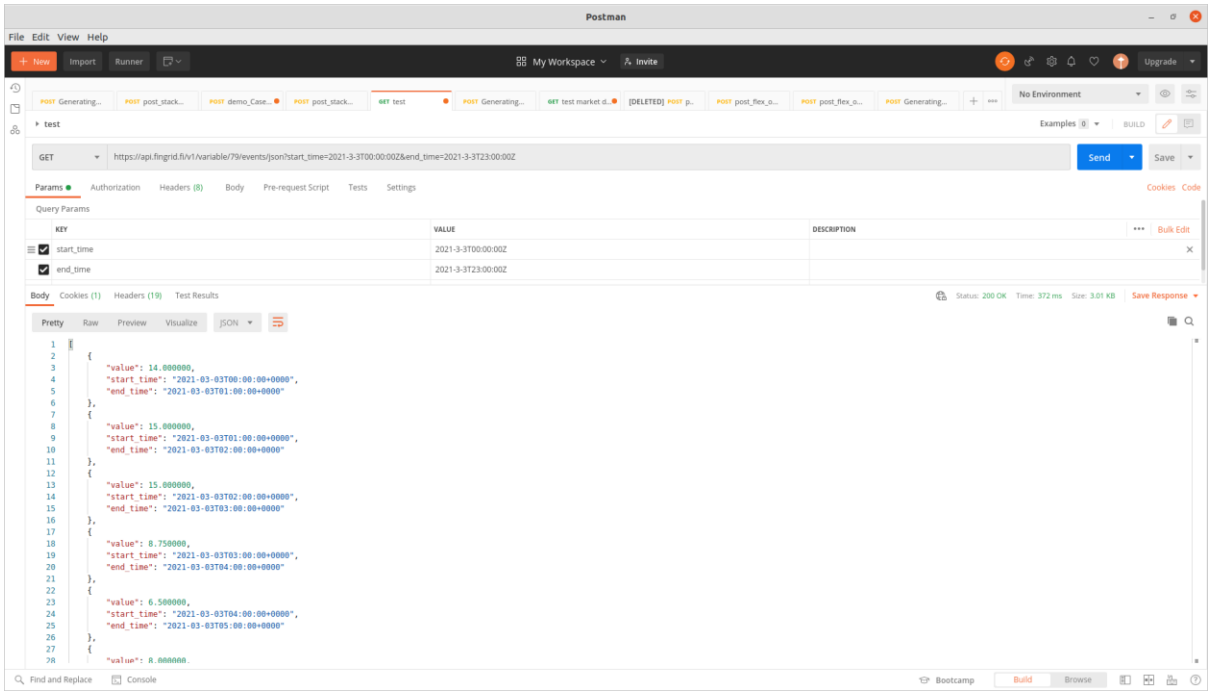


Figure 78: API (json format) for automatically retrieving frequency containment reserve (FCR) for normal operation market price data from FINGRID API to FLEXGRID ATP

5.2 Download, install and configure a FLEXGRID ATP service

This section explains the basic steps that an interested S/W developer should follow in order to be able to download, install and configure a FLEXGRID service in its own system. Due to the modular-by-design FLEXGRID ATP architecture, each FLEXGRID ATP service can be offered as a stand-alone service or as a part of a bunch of services according to the end customer's business preferences. FLEXGRID ATP deployment is based on open-source S/W tools and thus a basic (DEMO) version of FLEXGRID services are publicly available in the project's GitHub area. It should be noted that the final version of the FLEXGRID ATP (containing advanced functionalities tailored to specific customer segments) will be kept in closed access according to the FLEXGRID's exploitation plan.

5.2.1 Step 1: Design your API using swagger editor

As a first step, one should use the online tool at https://editor.swagger.io/ to create the API definition. As a starting point, one can use the swagger file that is provided by FLEXGRID project. Thus, the developer can copy and paste the source code into the swagger online tool and then adapt the API to meet the goals of the developer's endpoint. There will be one swagger file per FLEXGRID service. For example, the swagger file for UCS 2.3 service is ready for use, while the residual swagger files will be available within the next months.

5.2.2 Step 2: Connect to FLEXGRID Central Database

The second step is for the developer's API endpoint to connect to the FLEXGRID Central Database authorization system. For this reason, the developer should contact the FLEXGRID ATP administrator and ask for client credentials for testing his/her API. Then, the developer will obtain: i) a client id, ii) a username, and iii) a password. After that, the developer will be able to obtain a token by posting a curl request and get a respective response, in which there

is the token that the developer needs to test his/her API service. For more technical details about the connection to the FLEXGRID central database, please check https://github.com/FlexGrid/FST-service-3-stacked-revenues-maximization

5.2.3 Step 3: Deploy, test and run your server locally

The third step is to deploy, test and run the server locally (i.e. localhost). This server contains all the source code (i.e. written in python language) that needs to be executed in order for the FLEXGRID service to be delivered. The developer should visit https://editor.swagger.io/, and from the top menu select "Generate Server". A zip file will be downloaded by the browser. Then, this file should be unzipped and saved in a directory. Python3 and pip3 applications should be downloaded and installed in the local PC in order for the FLEXGRID algorithm to be executed properly. After making a few changes in the controller files and configuring the server to validate the types of requests/responses, the server will be ready to run locally by using the token that has been acquired from the previous step. For more technical details about the deployment and testing of the local server, please check https://github.com/FlexGrid/FST-service-3-stacked-revenues-maximization

5.2.4 Step 4: Deploy the FLEXGRID application on your server

Now that the server is up and running, the next step is to deploy the FLEXGRID application (e.g. UCS 2.3) on this server. This procedure is based on:
https://www.digitalocean.com/community/tutorials/how-to-serve-flask-applications-with-uswgi-and-nginx-on-ubuntu-18-04.
It assumes that the operating system is Ubuntu 18.04, and outside facing web server is nginx. We will use uWSGi as the application server for our application, which will only be accessible through nginx. Several technical steps should be followed based on the FLEXGRID developer's manual and can be summarized as follows:

- Install required packages
- Clone the project repository and create a virtual environment for python
- Activate venv
- Add files for usgi deployment
- Test that the server can start with wsgi
- Deactivate the venv
- Create a uwsgi configuration file following the technical instructions
- Add system configuration to automatically run the service
- Create a nginx configuration and the relevant certificates with certbot
- Validate that the FLEXGRID service is working properly

5.2.5 Step 5: Implement the UCS 2.3 algorithm

The algorithm that has been imported for the FLEXGRID UCS 2.3 service can be found in this repository: https://github.com/FlexGrid/stacked_revenues. In order to integrate the algorithm, one can add your repository as a git submodule. Then, one can call the submodule code from the controller that was generated by codegen. It should be noted that the basic version of FLEXGRID algorithms is publicly available for further reuse, testing and exploitation by every interested party. In case an interested individual or legal entity wants to use the full version of FLEXGRID services, then this service should be purchased according to the FLEXGRID's exploitation plan.

5.2.6 Step 6: Using external data or data to further test and validate the algorithm operation

In order to use data from an external API, the user can create an adapter file by following the technical instructions in https://github.com/FlexGrid/FST-service-3-stacked-revenues-maximization. Until now, the source code for retrieving the market price data from Nord Pool and FINGRID is available. Following this FLEXGRID API example, it is easy to retrieve data from other Market Operators', FMOs', TSOs' and DSOs' public APIs, too.

Moreover, tests with sample data can take place. The advantage of this alternative is that there is no need to rely on any external services, which are provided by 3rd party servers and thus their 24/7 availability is not guaranteed. Sample input datasets for FLEXGRID UCS 2.3 service are provided here: https://github.com/FlexGrid/FST-service-3-stacked-revenues-maximization/tree/master/sample_data_xlsx

Finally, in case a researcher is interested in deeply comprehend the UCS 2.3 algorithm's operation and test it with even more real-life datasets, s/he could use the research datasets found here: https://github.com/FlexGrid/Battery_Stacked_Revenues. Respective performance evaluation results are provided in chapter 5 of FLEXGRID D4.2. In this way, any interested individual researcher and/or research group may easily replicate the FLEXGRID results and possibly enhance them in the future.

6 Conclusions

This Deliverable presents the work carried out until M18 of T6.2 “Design of APIs and S/W Development” and T6.3 “GUIs and integration activities”.

Related to T6.2, it has been defined a less technical methodology for the initial definition of the APIs for each module developed within WP3, WP4 and WP5, by filling a spreadsheet template that, once filled, can be easily translated into the .yaml files supported by Swagger for documenting APIs.

Linked with T6.3, the mockups for all the UCS that are being implemented have been designed. Also, it has been implemented the needed functionalities to run the algorithm for proving the UCS 2.3, which is the UCS that has been selected to be demonstrated during the 1st review of the project.

The actual outcome of both tasks can change during the second half of the project, mainly because the work that is being performed within WP3, WP4 and WP5 is still ongoing. The final version of these 2 tasks will be reported in M33 in D6.3 “Final version of FLEXGRID S/W prototype”.