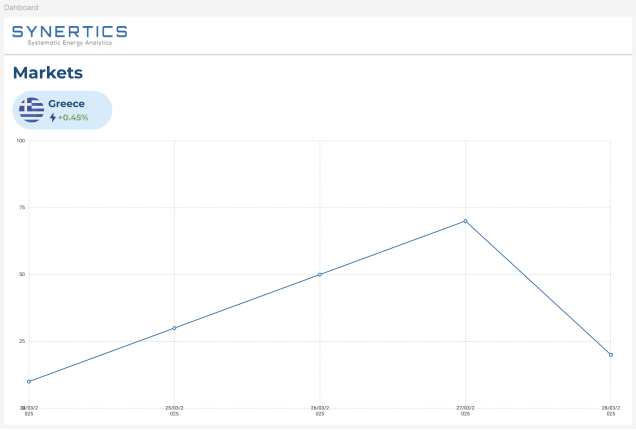
Greek electricity futures prices dashboard

Created by André Morais, last updated on 31/03/2025

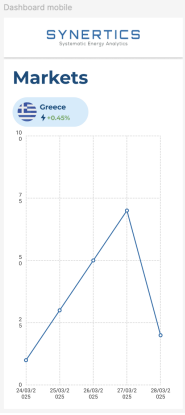
Background and strategy

Design

Desktop



Mobile



Source: Synertics' Figma file

Concept

In this challenge, you will develop a data visualization dashboard for Greek energy futures market data using Django (Python). The application should display market trends similar to what you see in the provided mockups, showing how energy futures prices fluctuate over time.

The Greek energy market is a dynamic environment where prices can change significantly based on various factors including supply, demand, regulatory changes, and regional events. Your challenge is to create both frontend and backend components that work together to deliver an intuitive visualization of this data.

Review the mockups provided to understand the design requirements. Then, develop a plan for how you'll approach both the Django templates implementation and backend data collection. Remember to consider how to implement scheduled tasks in Django for daily web scraping.

SYNERTICS
Systematic Energy Analytics

Requirements

#	User Story	Priority	Notes
1	As a user, I want to view energy market trends in a line chart so I can analyze futures performance over time.	Must have	-
2	As a user, I want a responsive design that adapts between desktop and mobile views so I can access market data on any device.	Must have	If it is implemented as a Progressive Web App (PWA), it's a bonus.
3	As a user, I want to view the variation between today's price and yesterday's price.	Must have	-
4	As a system administrator, I want the application to automatically scrape futures data from the ENEX Group website **daily** so that users always have access to the most recent market information.	Must have	It is obtainable through the link: https://www.enexgroup.gr/documents/20126/314344/{yyyy}{mm}{dd}_DER_DOL_EN_v01.xlsx in excel format.
5	As a developer, I want all scraped futures data to be stored in a SQLite database with proper timestamps so that we maintain a historical record of market trends.	Must have	If it is implemented in PostgreSQL, it's a bonus.
6	As a developer, I want a REST API endpoint that returns properly formatted chart data so that I can easily integrate it with the frontend components.	Must have	-
7	As a developer, I want the API to provide the percentage change, so that the frontend can display derived metrics without additional processing.	Should have	-
8	As a developer, I want all the code well documented so that I can understand it in the future.	Should have	-
9	As a system administrator, I want to receive notifications if the web scraping process fails so that I can take immediate corrective action.	A bonus	-
10	As a system administrator, I want to see all the logs for the scraping process.	A bonus	If well configured in Django, the package *logging* can be used.

Notes

You don’t need to worry about authentication. You can use Django’s default authentication system, but the dashboard must be accessible to everyone.

Trade Day	Instrument	Number of Orders
28/03/25	GREBM0425	7
28/03/25	GREBM0525	6
28/03/25	GREBM0625	
28/03/25	GREBQ325	2
28/03/25	GREBQ425	
28/03/25	GREBY26	2
28/03/25	GREPM0425	6
28/03/25	GREPM0525	
28/03/25	GREPM0625	
28/03/25	GREPQ325	2
28/03/25	GREPQ425	
28/03/25	GREPY26	2

When performing the automatic daily scrape, you’ll obtain a table like the one shown below. The “instrument” column represents the futures product name, which is structured as follows:

- “GRE” indicates Greece.
- “P” or “B” denotes peakload or baseload futures, respectively.
- “M,” “Q,” or “Y” signifies the contract period—month, quarter, or year.
- The final part identifies the specific month, quarter, or year of the contract.

Examples:

- GREBM0425 - Baseload futures for April 2025.
- GREPQ325 - Peakload futures for the third quarter of 2025 (starting July 1st).
- GREBY26 - Baseload futures for 2026.

In the chart, you should display the values from the “Average Price of Orders” column, specifically for yearly baseload futures. In this example, that means only the daily values for the instrument “GREBY26.”.

Although only one column and one instrument are displayed in the chart, you may want to store all products and values from the table in the database.

This challenge is a small example of Synertics’ workload.

SYNERTICS
Systematic Energy Analytics

Questions

1. What frameworks do I need to use in the visualization part? - You are free to use any chart framework.
2. How frequently is the ENEX Group server updated? - The server is updated daily around 6pm.
3. What happens if the scraper fails to collect data on a specific day due to a lack of available data? - You can skip the day.
4. Can we manually trigger the scraper outside of the scheduled time? - We should be able to manually call the scraper function.
5. How far back can we view historical data in the charts? - The chart must show the last 7 days of available data.
6. What authentication is required to access the API? - To make it easier, it can be unrestricted.
7. How is the data formatted in the API response? - Ideally in a JSON formatted string, but its opened to better options.
8. Can we display multiple countries on the same chart? - No, but the real software includes many available countries.
9. How are weekends and holidays handled in the data visualization? - They aren't shown because the market hasn't trades in the weekend/holidays.
10. Do I need to implement the system in Docker? - It's a bonus and it's easier if you plan to use PostgreSQL. If you plan to use Celery and Redis Docker can also be useful.
11. How do I deliver the challenge? You can deliver it in Git repository or in a Zip file.

Good luck :)