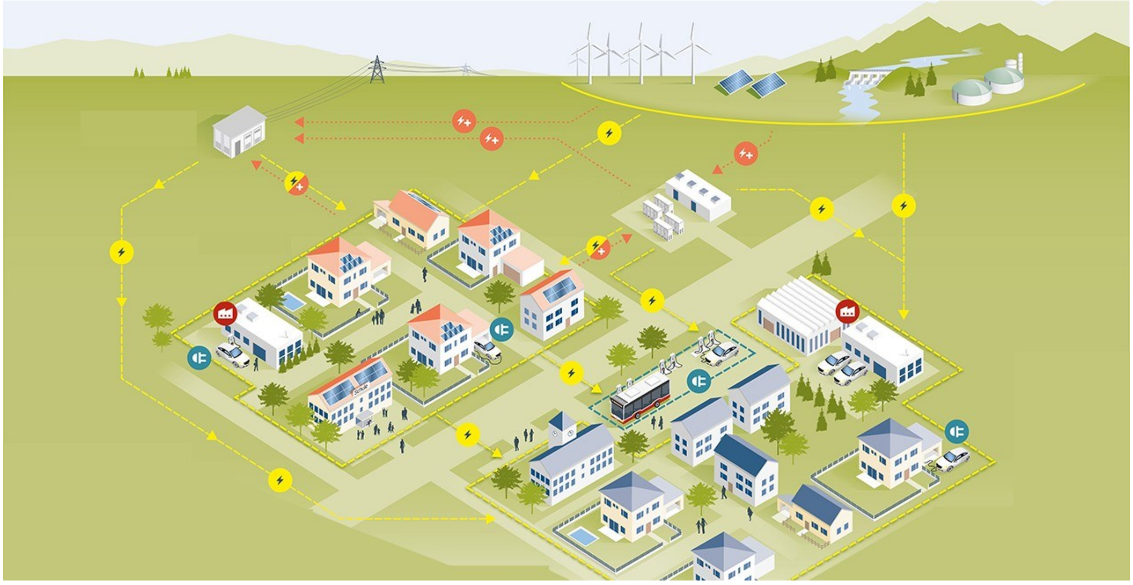


^

Specification Semester Project

Energy communities

An energy community is an association of at least two participants for the joint production and utilization of energy. There are community producers and users and grid producers and users. You should write a system, consisting of multiple components, that shows the current energy distribution and usage.

Idea

In the center of the system is a message queue that receives energy production and usage messages. Based on these updates, a service should calculate the current community and grid usages. If a community user wants energy, the community energy pool will be used first. Otherwise the grid will deliver the energy.

After the usage is calculated, another service calculates the percentage for the current hour based on the usage.

You can monitor the current distribution of energy on a Graphical User Interface (GUI). You can also ask for historical data.

Components

You have to develop 6 components for this project. **Every component is its own application that can be started independently form the other applications.**

Community Energy Producer

A community energy producer sends the following message to the queue:

- type: PRODUCER
- association: COMMUNITY
- kwh: *the kWh produced in a minute* (e.g. 0.003)
- datetime: *the datetime of the energy production* (e.g. 2025-01-10T14:33:00)

The Energy Producer should send a message every couple of seconds with a semi random (but plausible) amount of kWh. Incorporate a Weather API to make sure more energy is produced when the sun is shining.

Community Energy User

A energy user sends the following message to the queue:

- type: USER
- association: COMMUNITY
- kwh: *the kWh used in a minute* (e.g. 0.001)
- datetime: *the datetime of the energy usage* (e.g. 2025-01-10T14:34:00)

The Energy User should send a message every couple of seconds with a semi random (but plausible) amount of kWh. Incorporate the time of day to make sure more energy is needed in peak hours in the morning and in the evening.

Usage Service

Every time a new PRODUCER or USER messages comes in, the database is updated. The data from individual minutes is accumulated into the corresponding hours, e.g.:

Database table before the new USER message:

hour	community_produced	community_used	grid_used
2025-01-10T14:00:00	18.05	18.02	1.056
2025-01-10T13:00:00	15.015	14.033	2.049

A new messages is processed by the queue:

- type: USER
- association: COMMUNITY
- kwh: 0.05
- datetime: 2025-01-10T14:34:00

Database table after the message:

hour	community_produced	community_used	grid_used
2025-01-10T14:00:00	18.05	18.05	1.076

hour	community_produced	community_used	grid_used
2025-01-10T13:00:00	15.015	14.033	2.049

As the community user required more energy than was available in the community production pool, grid usage also increased.

Current Percentage Service

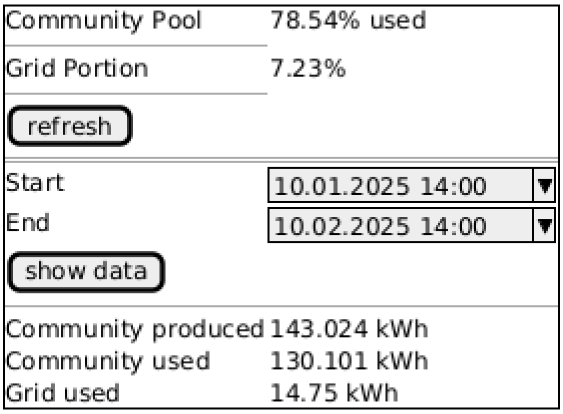
Because the usage changed, a new percentage has to be calculated.

hour	community_depleted	grid_portion
2025-01-10T14:00:00	100.00	5.63

This means the community pool is 100% depleted and the grid portion of the total energy was 5.63% . The table only hold the information of the current hour.

GUI

This information needs to be displayed somewhere. Use JavaFX to create a GUI that can display the current percentage data and historical data based on a time filter.



Community Pool	78.54% used
Grid Portion	7.23%

Start	10.01.2025 14:00
End	10.02.2025 14:00

Community produced	143.024 kWh
Community used	130.101 kWh
Grid used	14.75 kWh

Important: The GUI is not directly connected to the database. The GUI uses a REST API to fetch the data.

REST API

Use Spring Boot to create a REST API with two endpoints:

- GET /energy/current
    - returns the percentage of the current hour
- GET /energy/historical?start=...&end=...
    - returns the usage date for a given time period (start to end)

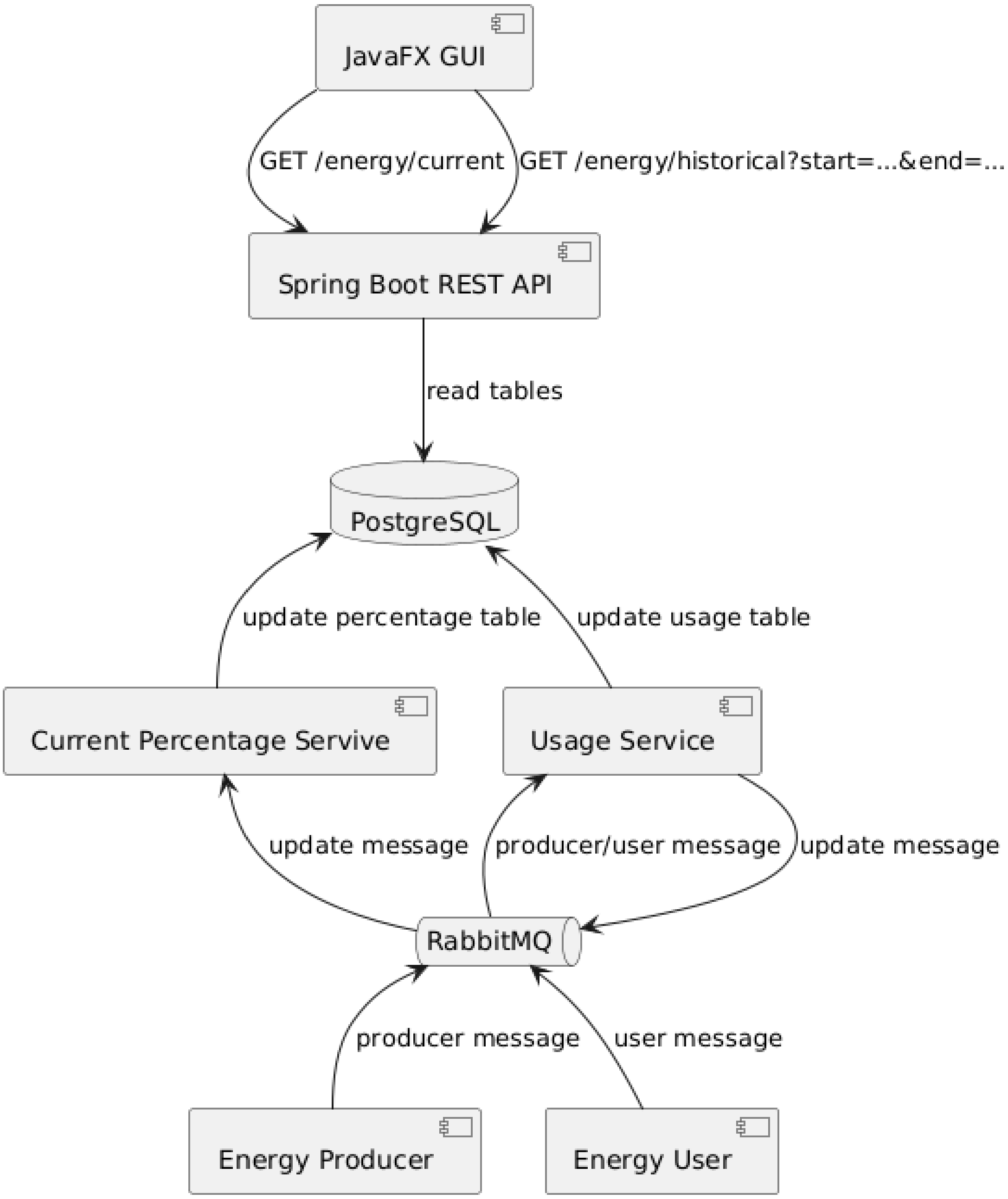
The Spring Boot Application is connected to the database, but can only read the information from the tables.

Example Timeline

1. Community Energy User sends minute usage data to queue

2. Usage Service picks up the minute data and updates the hour data in the database
3. Usage Service sends a message to the queue that new data is available
4. Current Percentage Service picks up the new data and saves the calculated percentage data to the database
5. GUI wants to refresh the current percentage und sends a GET request to the REST API
6. The REST API handles the request, reads the data from the database and returns the data to the GUI
7. The GUI displays the data to the user

Component Diagram



Project Grading Schema

Completion requirements

Milestone

A part of the system has to be finished by class 8. This will be graded by a Code Review.

Component	Description	Percent
Must Haves	• Every component can be started independently • System can be build and run with no errors • Spring Boot used for REST API • JavaFX used for GUI • **GitHub repository link in submission**	**mandatory** If this is not in the project, the milestone will be graded with 0 points
REST API	The REST API will be implemented using Spring Boot and designed to provide example data for testing and demonstration purposes. It will include two endpoints: one to retrieve the data of the current hour, and another to filter historic data. The API will focus solely on returning structured example data to simulate real-world usage scenarios without relying on a persistent data store or handling complex business logic.	50%
GUI	The GUI will be developed using JavaFX, providing an intuitive and interactive interface for users to interact with the REST API. The application will include buttons and input fields to send requests to the API, such as fetching the hour and historic energy data. Retrieved data will be displayed dynamically within the application using visual components like tables, labels, or text areas. The design will focus on simplicity and clarity, ensuring a seamless user experience while demonstrating the integration of JavaFX with a REST API for real-time data interaction.	50%

Final Submission

The final submission must be presented in class and the project will be graded by Code Review.

Component	Description	Percent
Must Haves	• Every component can be started independently • System can be build and run with no errors • Spring Boot used for REST API • JavaFX used for GUI • RabbitMQ used for communication between services • **GitHub repository link in submission**	**mandatory** If this is not in the project, the final project will be graded with 0 points

REST API	The Spring Boot App reads the data from the database instead of using static sample data.	10%
Energy Producer	Sends production message in random 1-5 second intervals to the message queue. Production message must include a random but sensible kWh value.	10%
Energy User	Sends usage message in random 1-5 second intervals to the message queue. Usage message must include a random but sensible kWh value.	10%
Usage Service	Receives the production/usage messages and updates the usage table correctly. Afterwards sends an update message to the queue.	40%
Current Percentage Service	Receives the update message and updates the current percentage table correctly.	30%

Points can be omitted for badly written code, unsufficient presentation or the failure to explain the system.