TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA
ROMANIA

Faculty of Automation
and Computer Science

Energy Utility Platform

-Distributed Systems project-

Fazakas Borbála
Group 30441
2023

Table of Contents

Overview

The energy utility platform is an application intended to facilitate monitoring the energy utilisation of the devices in any environment.

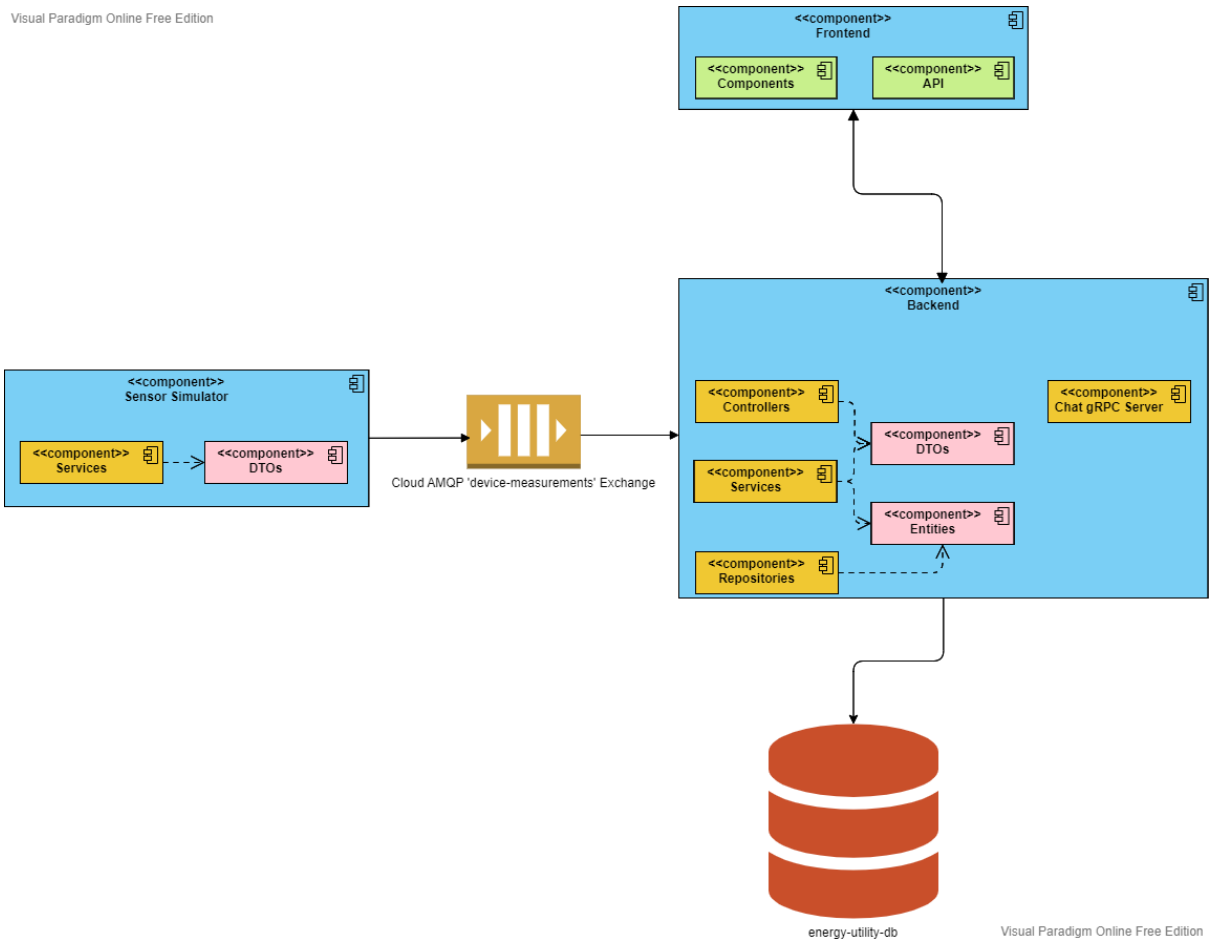
It supports two types of users: admin and clients, and the following functionalities:

- admins can
    - register new clients and admins
    - login
    - view the data of all clients and admins
    - update the data of existing clients or admins
    - delete any clients or admins (except themselves)
    - register new devices (includes specifying the client who owns the device)
    - view the data of all the existing devices
    - update the data of existing devices
    - delete any devices
    - chat with clients, to assist them
- clients can
    - register themselves
    - login
    - view their associated devices
    - view the plot of the energy consumption of any of their associated devices, for any day
    - see the logs of their devices' consumption updates
    - get notified if one of their devices' consumption passes over the hourly consumption threshold
    - chat with the admin, if they need help

Additionally, the platform collects the consumption measurements from all the devices, via a message broker middleware.

Conceptual Architecture

Overview



The application has a standard 3-Tier architecture, with:

- a Microsoft SQL Server, for persistent data storage
- a Spring Boot backend, for the business logic and the data access
- a ReactJs frontend, for the presentation

Additionally, a Sensor Simulator is implemented, which simulates the reading of the sensors for each device. This communicates with the backend via a message broker (a RabbitMq exchange).

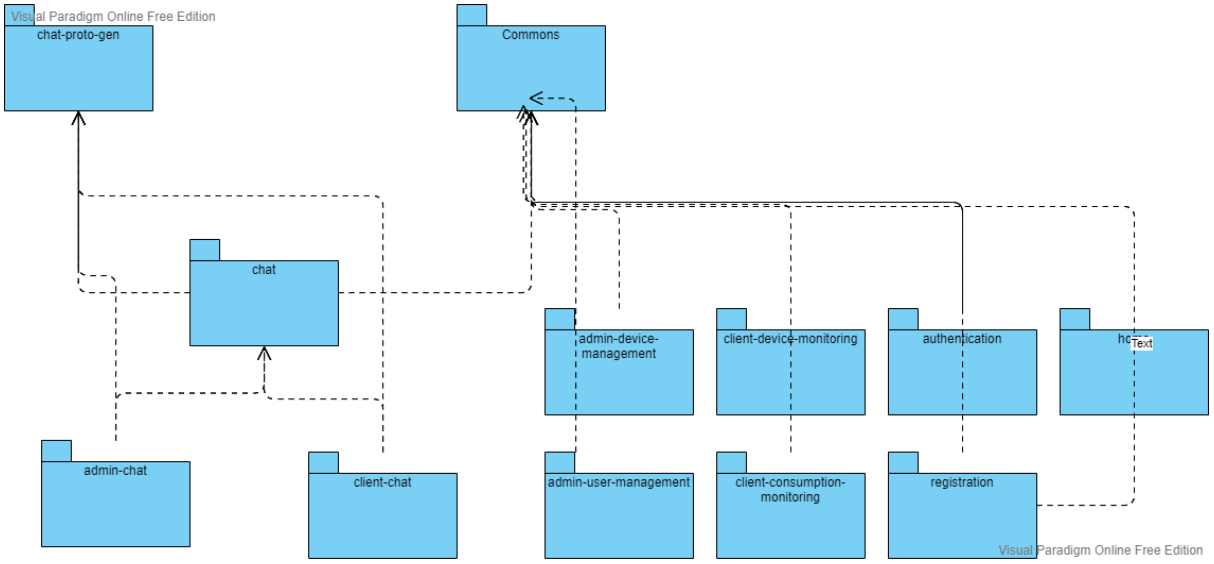
The backend

- offers a Rest API for accessing its services. The frontend uses this API to perform all the data retrievals and updates, through HTTP requests
- sends notifications to the frontend via websockets
- offers a gRPC Service for the chat functionalities

Front End

As specified above, the front end is implemented with react and it is responsible for providing an easy-to-use user interface for the end users. Thus, it offers pages for

- home (the presentation of the app)
- registering
- login
- managing the devices, for the admins (CRUD operations)
- managing the users, for the admins (CRUD operations)
- visualising the client's devices, for clients
- visualising each device's energy consumption for a given day, for clients
- showing the energy consumption logs fo the devices of a client
- chat, for clients
- chat, for admins



Back End

The back end is where all the business logic is implemented.

First, the back end has 4 main packages:

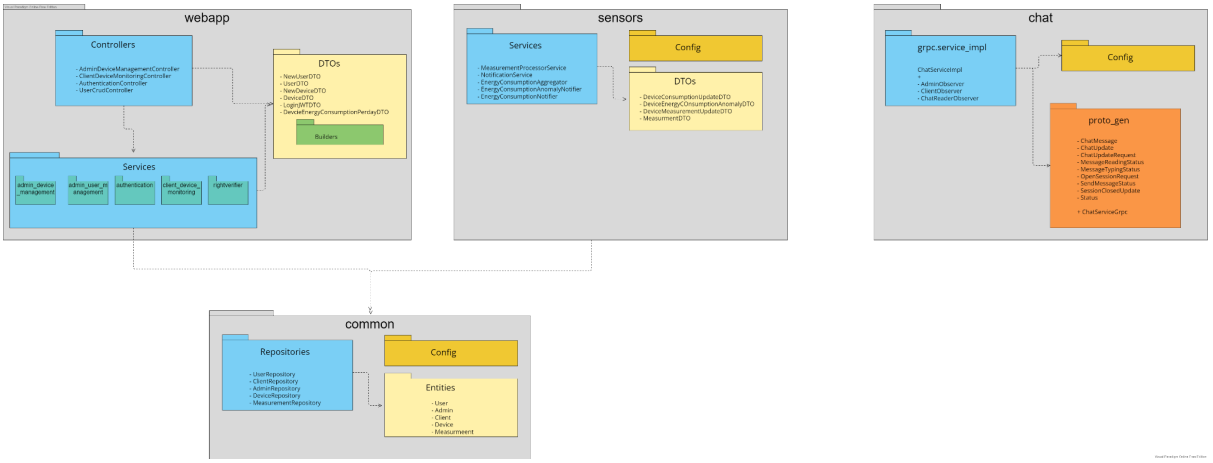
- common: defining the underlying entities, repositories and the shared configuration parameters
- webapp: defining the Rest API of the web application, for the clients
- sensors: handling the processing of the measurement messages from the devices and the notification of clients
- chat: defining the service for the chat functionality

Then, it is structured in layers itself, with

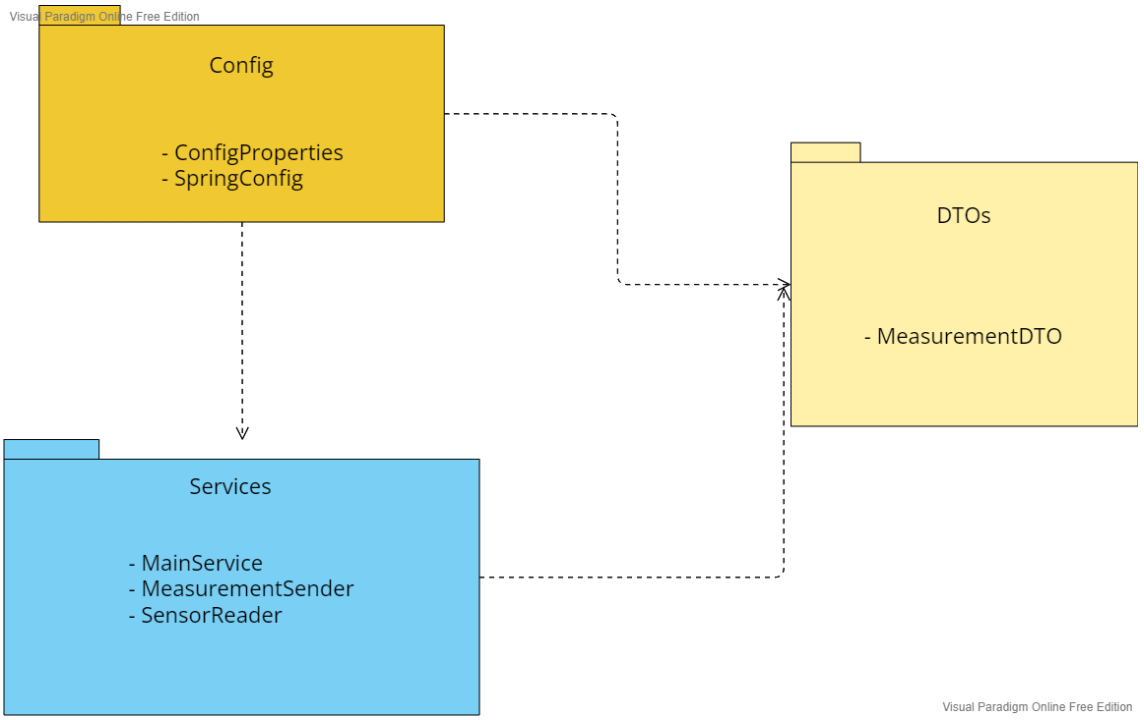
- the controllers being responsible for taking up and handling HTTP requests from the clients
- the services implementing the business rules

- the repositories creating the connection with the database

The controllers communicate via light-weight DTOs (data transfer objects) with the clients and the services, but the services convert the DTOs to entities, in order to mirror the data representation in the underlying database.

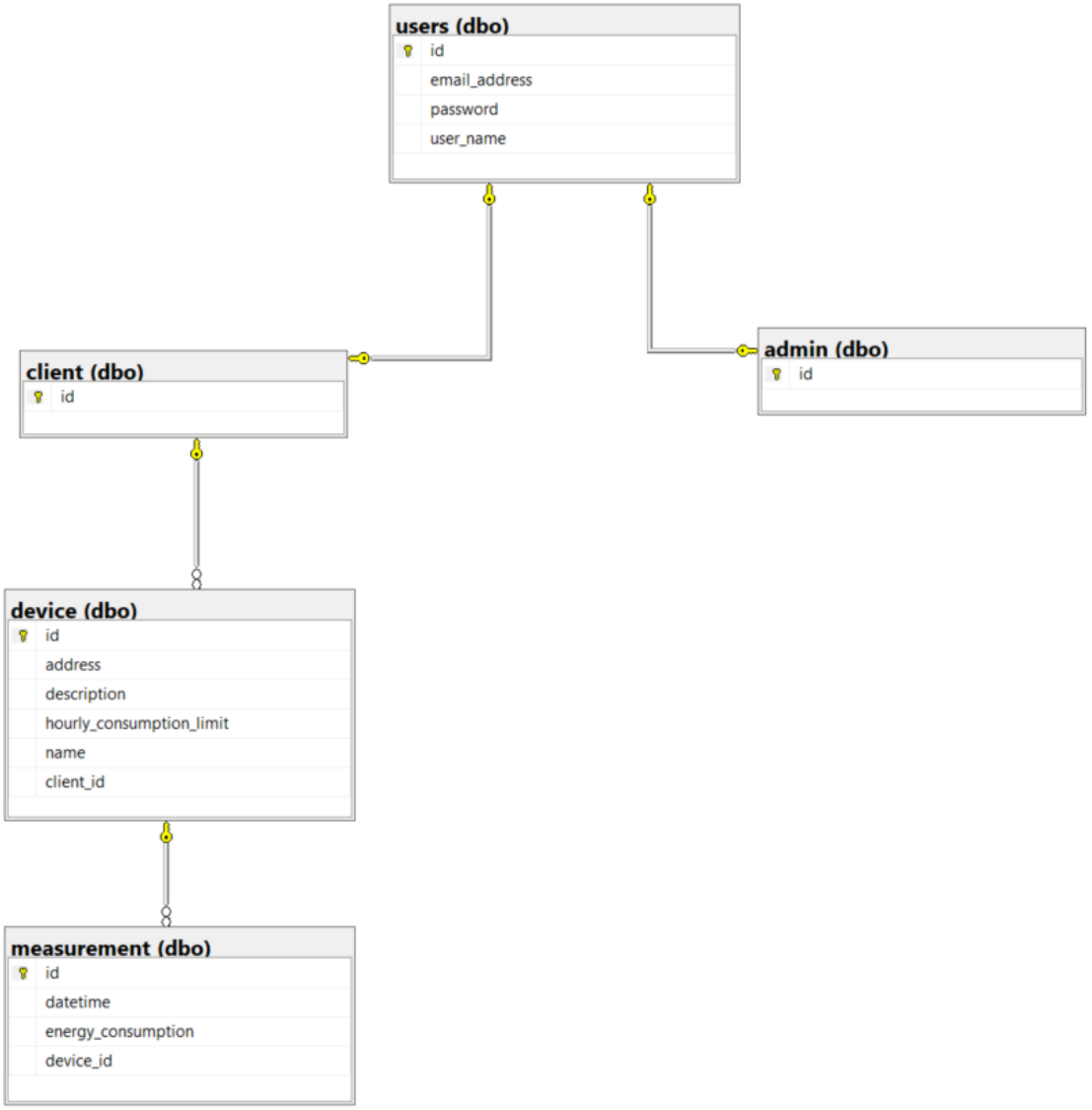


Sensor Simulator

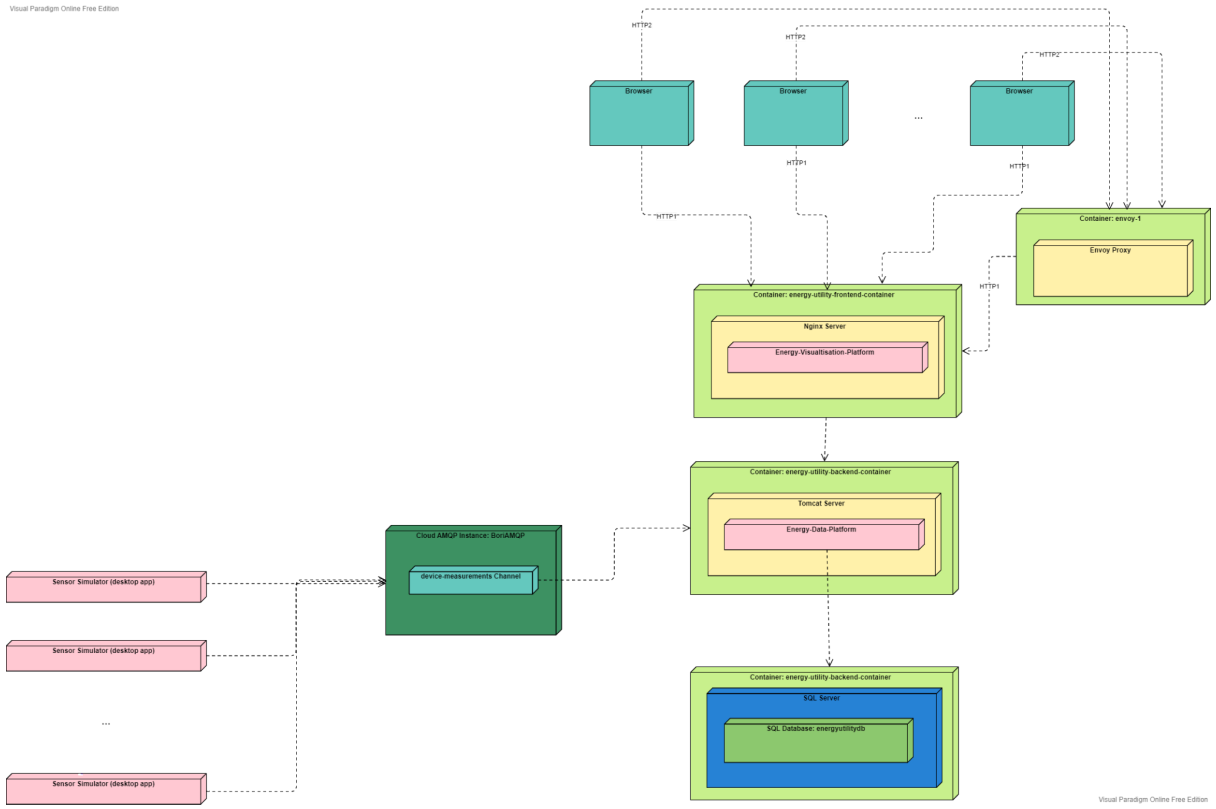


Each SensorSimulator process simulates reading the values from the device's sensor, by reading the values from a CSV file. The values are then sent to the backend, via the RabbitMQ Exchange, together with the timestamp and the device's id.

Database Design



UML Deployment diagram



Instructions for deploying and running the app

On the azure-deploy branch the app is prepared for deployment via a CI/CD pipeline in azure. Explaining the process for setting up this pipeline is out of scope here, but I'll highlight the primary steps:

1. clone the Github repositories for the backend and the frontend
    a. https://github.com/bori00/DS2022_30441_Fazakas_Borbala_Assignment_3_Backend
    b. https://github.com/bori00/DS2022_30441_Fazakas_Borbala_Assignment_3_Frontend
2. enter the folder of the backend project

```
cd DS2022_30441_Fazakas_Borbala_Assignment_2_Backend
```

3. build the docker image for the backend

```
docker build -t energy-utility-backend .
```

4. run the container for the backend

```
docker-compose up
```

5. enter the folder of the frontend project

```
cd DS2022_30441_Fazakas_Borbala_Assignment_3_Frontend
```

6. build the docker image for the frontend

```
docker build -t energy-utility-frontend .
```

7. run the container for the frontend

```
docker-compose up
```

8. access the app at *localhost*