# Online Energy Utility Platform

Technical University of Cluj-Napoca

Computer Science Department

**Distributed Systems – Prof. Cristina Pop**


Coordinated by:

**Andreea Vesa**

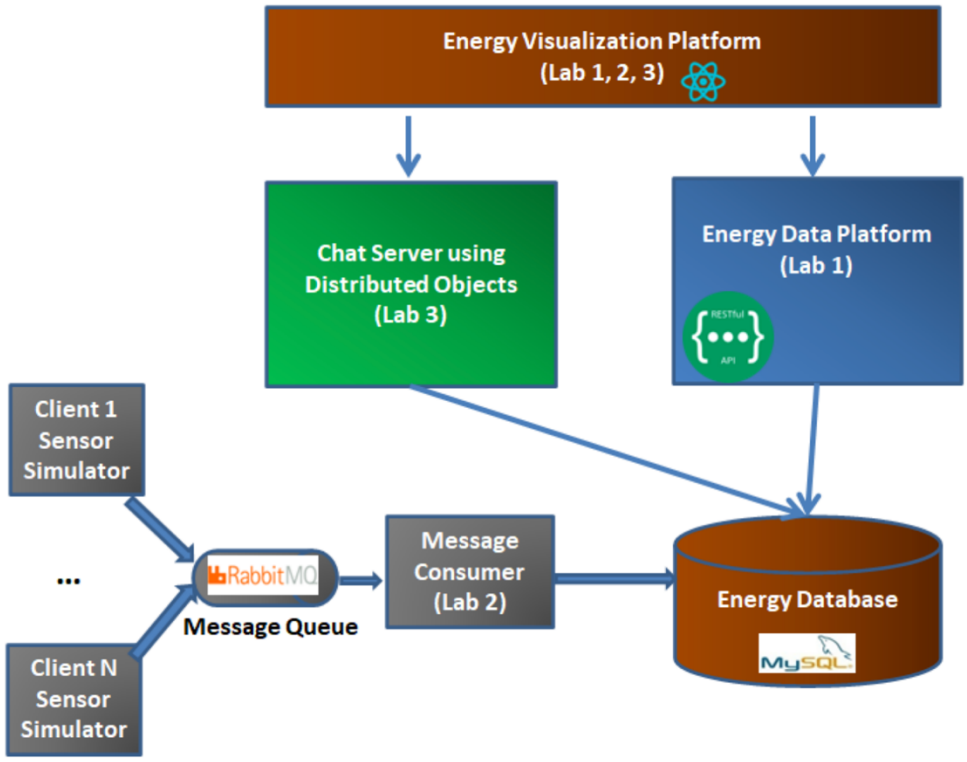
Created by:

**Titus Boga**

# Conceptual architecture of the online platform

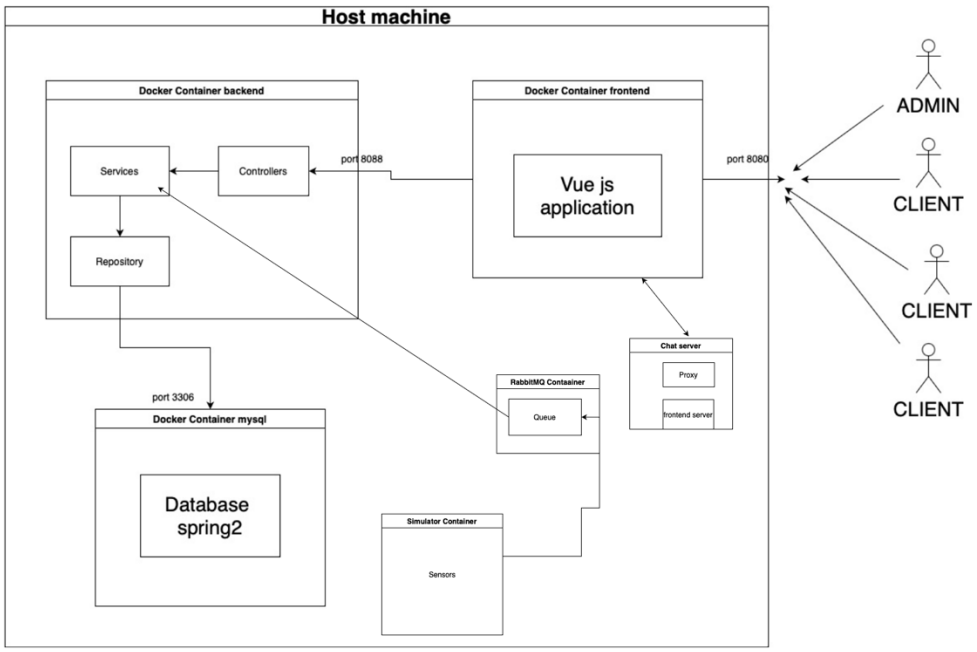
The platform is designed to provide the necessary functionalities for the 2 types of users that can be using the application. The design is structured in such manner that the administrator can login and will be welcomed by a page where he could edit all the needed information regarding all the users the applications has stored. By clicking a certain button, the administrator will be redirected to another page that will offer the possibility to add, delete or edit and link devices.

The client part which is not yet implemented should allow the user connected with a client account to have a visualization of the devices linked to its account. The user should also be prompted with a chart containing the consumption of the energy for the current day and the user should be able to choose from a shown calendar the day on which the consumption should be shown. This of course should be done dynamically for all the devices.




# UML Deployment Diagram

This application is thought to run on a machine with an ip accessible by all the user (could be a public ip or an ip found inside an infrastructure). By accessing the ip of the host machine and the port 8080 from the browser, the frontend part of the application will be displayed. Because the application is divided in 3 services which are running isolated in 3 different containers we had to make sure that they can communicate with each other by taking care that their ports are accessible. (8088 for the spring <backend> and 3306 for the mysql <database>).




When a user accesses the frontend through their browser they are asked to log in or sign up, which, just like in the most cases, triggers a command send to an endpoint provided by the spring application inside the controllers using the REST API. The controllers then access the services which deal with most of the logic which we don't want to include inside the controllers for various reasons. After processing everything needed inside the service, the next step is to work with the data that we have in order to deliver a response to the user. In this way the flow goes to the Repository part where, in most cases, triggers a command to the MySql database in order to have a check, to save some information, to edit it or to delete it (of course based on the action triggered by the user). The Repository represents basically the connection between the services and the database (the persisted data).

The previously presented information can be more easily understood by taking a look at the diagram below, where the flow of the application is shown by using a UML Deployment diagram consisted of the most important parts of the application.

# Building and execution considerations

Building and deploying the frontend and the backend is being done in the same manner as for the assignment 1. Moreover for the Simulator we have to use a more specific Dockerfile that can be found on the repository which include using certain .jar files that make the running of RabbitMQ functionalities. Everything can be found on the Simulator repository.