Distributed Systems Assignment 3 Documentation

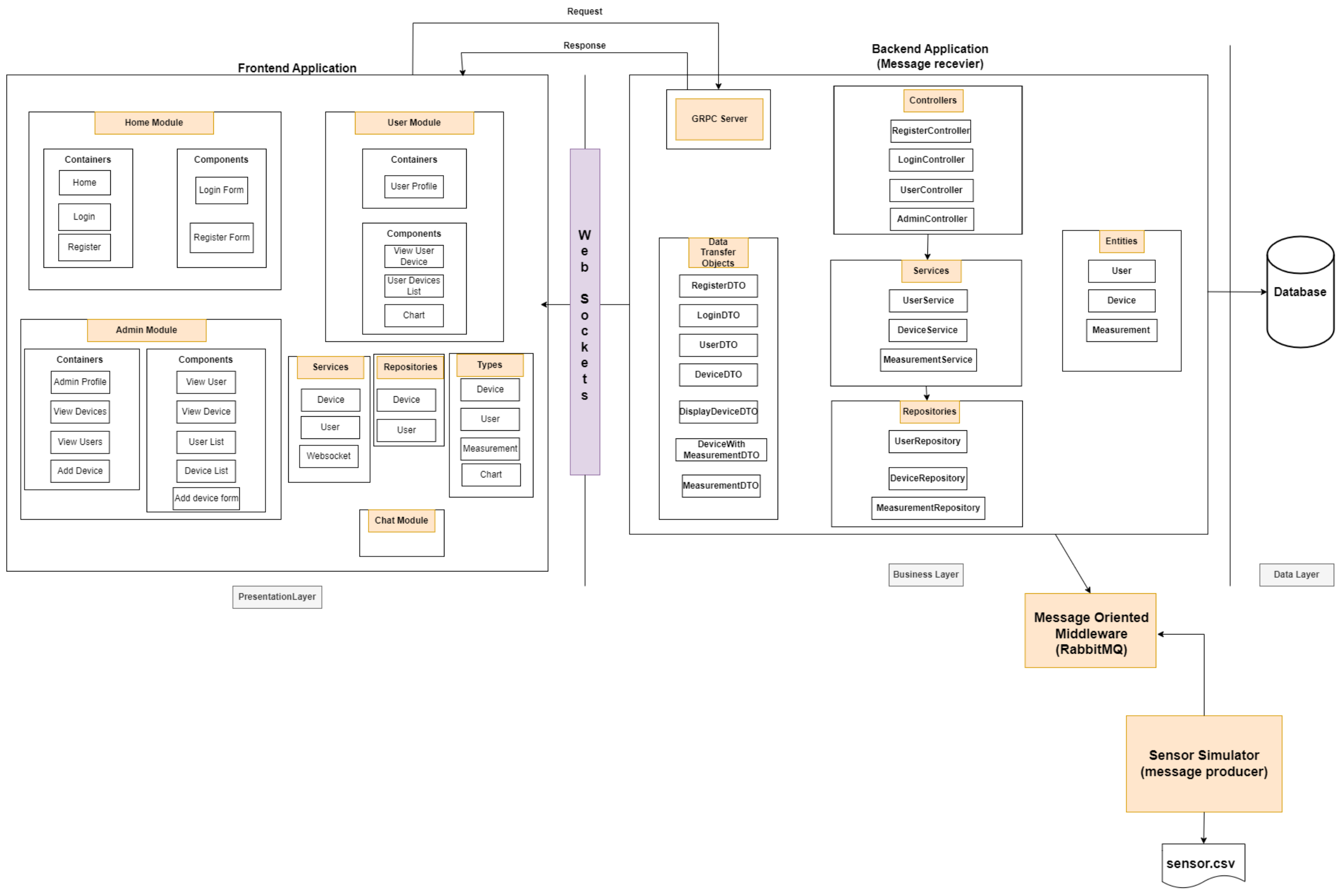
Online Energy Utility Platform

Student: Moisa Oana Miruna

Group: 30442

Table of contents

Conceptual architecture of the online platform



The Data Layer consists of the database.

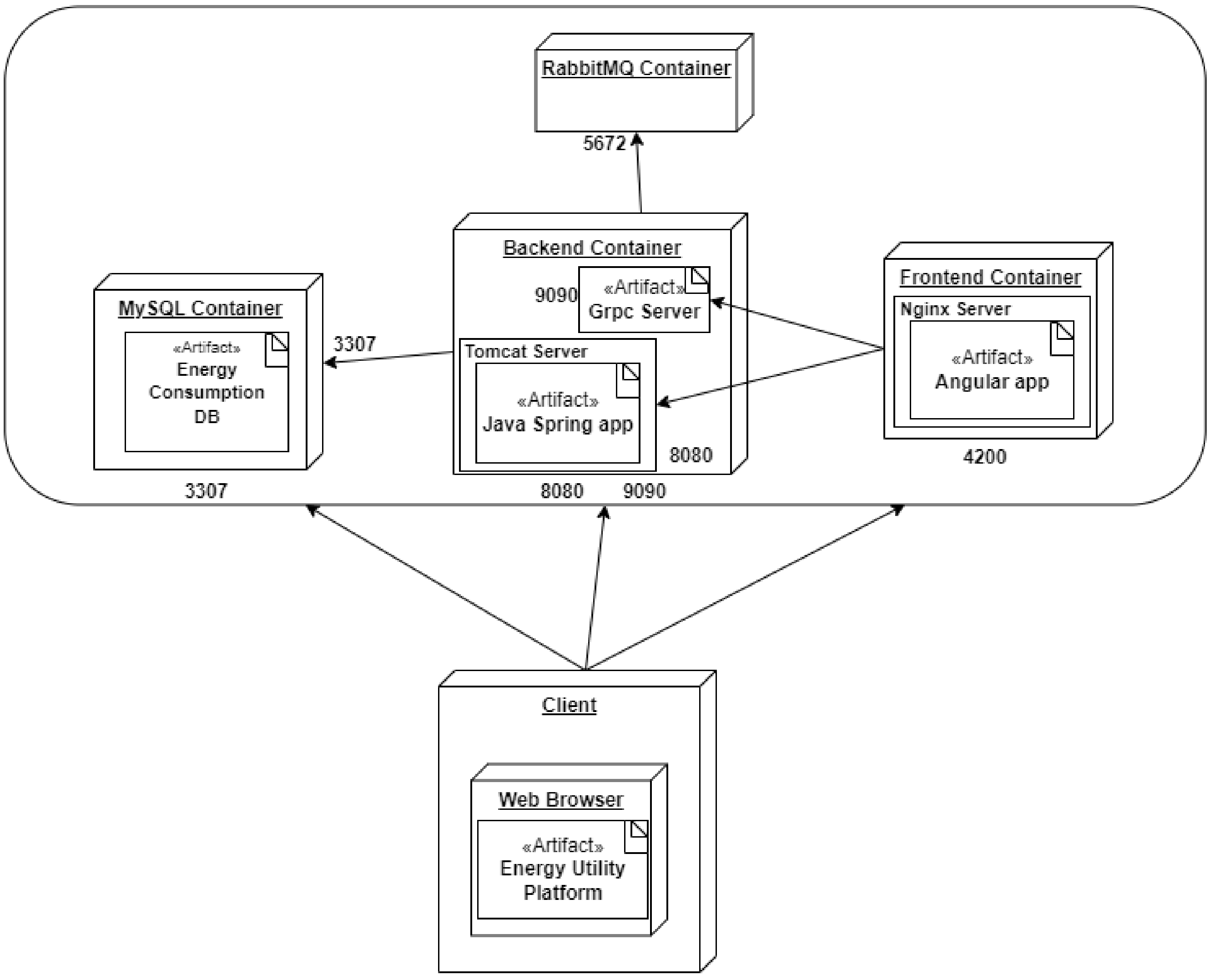
The Business Layer is composed of:

- Resources: Entities and DTOs. The entities correspond to the tables in the database: User, Device and Measurement. The DTOs are used to facilitate the communication between the frontend application and the backend application (between the business layer and the presentation layer). We use DTOs for API requests, to send or receive only the specific data we need. For example, we use the RegisterDTO to send the user data for creating an account to the backend. We use the LoginDTO also to send the user credentials (username and password) to be checked in the backend.
- Repositories: We have one repository class for each model class. Using the repository classes we can perform CRUD operations on the data from the database. For the repositories we use JPA Repository.
- Services: We also have one service class for every repository class. This layer can only access the repository layer below it. In this layer we will do the validations to the data we want to insert in the tables and other functionalities and then we will call methods from the persistence layer to do the actual operations to the database.
- Controllers: This layer will hold the classes that deal with the user interface. Each module from the frontend application will have one corresponding controller class in this package.

The frontend application was developed in the Angular framework, therefore the presentation layer is organized in modules. The module is a container for the different parts of an application. When I chose the modules, I thought of the main sections of the application: the Home module represents the login and register functionalities, the Admin and the User modules hold the functionalities specific to each type of user. Each module has a containers package and a components package. The containers are also called smart components because they can communicate with services to fetch and send data to the backend, they are the pages we usually route to. The components are also called dumb or presentational components because they don't have access to services, they can only communicate with the parent container and they are great for reuse. They can be small sections form a web page. One example is the flow of displaying a list of devices: we have a view-devices container that communicates with the Device Service to get the data from the API, this container sends the data to the devices-list component, which 'calls' the view-device component for each device from the list. The view-device component describes the template for displaying a single device.

We can also define services, repositories and types in the modules. The services and repositories work exactly like they do in the backend, with the exception that the repositories from the frontend make the api calls.

UML Deployment Diagram



The deployment diagram consists of 4 containers/nodes for the deployment of the database, backend application, frontend application and RabbitMQ server. The artifacts are concrete entities that are deployed on the nodes. The client can access the deployed application from a web browser on his/her own device.