Online Energy Utility Platform

Student: Miu Daria
Group: 30442
Profesor: Cristina Pop

Overview of project

This project aims to design and implement an online platform to manage users, their associated smart energy metering devices, and the monitored data from each device.
The system can be accessed by two types of users after a login process: administrator (manager), and clients. The administrator can perform CRUD (Create-Read-Update-Delete) operations on user accounts (defined by ID, name, role: admin/client), registered smart energy metering devices (determined by ID, description, address, maximum hourly energy consumption), and on the mapping of users to devices (each user can own one or more smart devices in different locations). After the mapping is done, for each device the energy consumption is stored hourly as tuples of the form in the database.

The application is implemented using Java. As a framework, Spring Boot has been used. The front end was developed using React framework. The IDE used for developing the project will be IntelliJ IDEA. The database is a MySql relational database, the database is handled in DataGrip and MySql Workbench using MySQL.

The architecture of the system

The system will have a layered architecture in which each layer communicates with the layer directly below or above it (hierarchical structure).

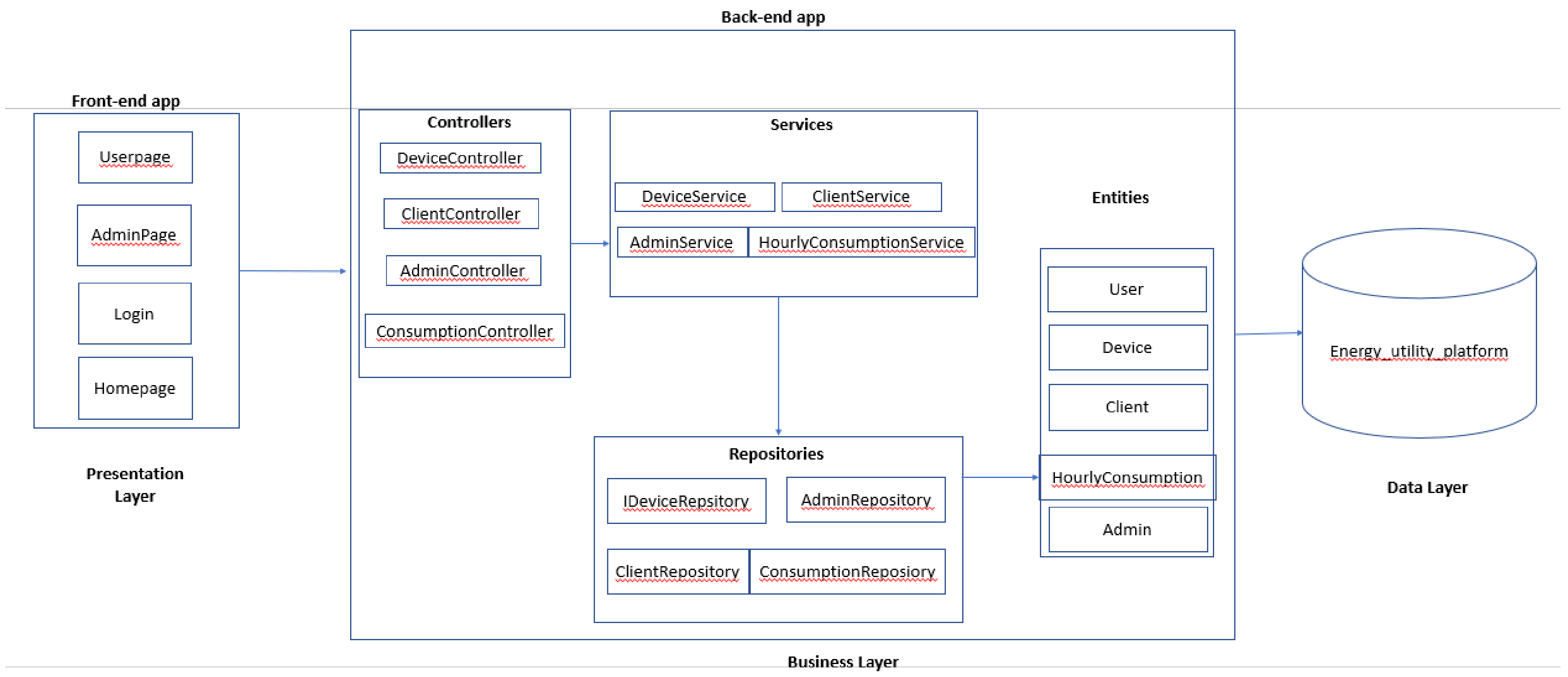
Presentation Layer: The presentation layer handles the HTTP requests, translates the JSON parameter to an object, authenticates the request, and transfers it to the business layer through **controllers**. In short, it consists of **views**.

Business Layer: The business layer handles all the business logic. It consists of **service** classes and uses **repositories** provided by data access layers to get the data from the database and send it to the database. It also uses **dtos** to communicate with the presentation layer.

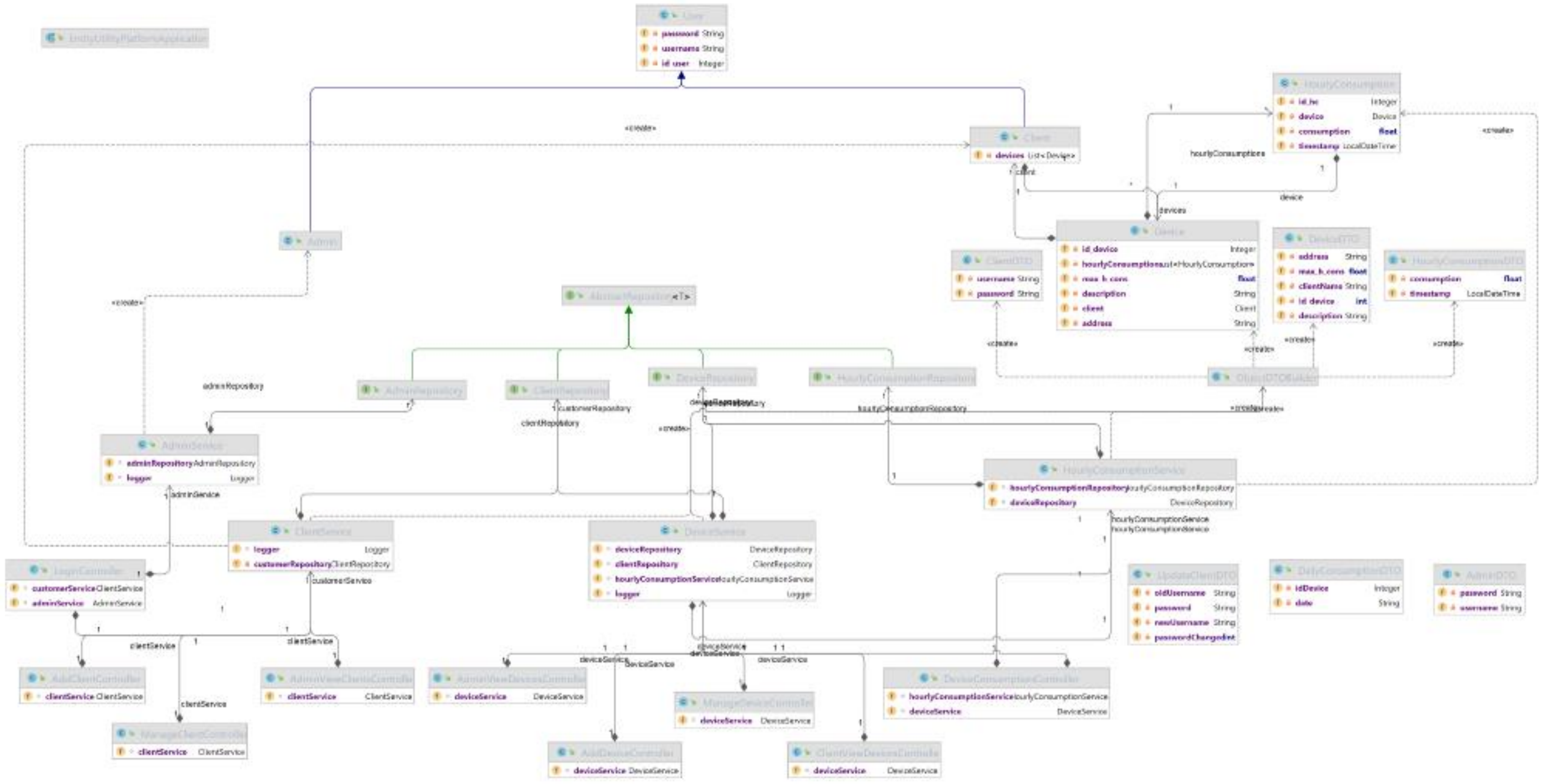
Persistence Layer: The persistence layer contains all the storage logic and translates business objects from and to database rows through **repositories**.

Database Layer: In the database layer, **CRUD** (create, retrieve, update, delete) operations are performed.

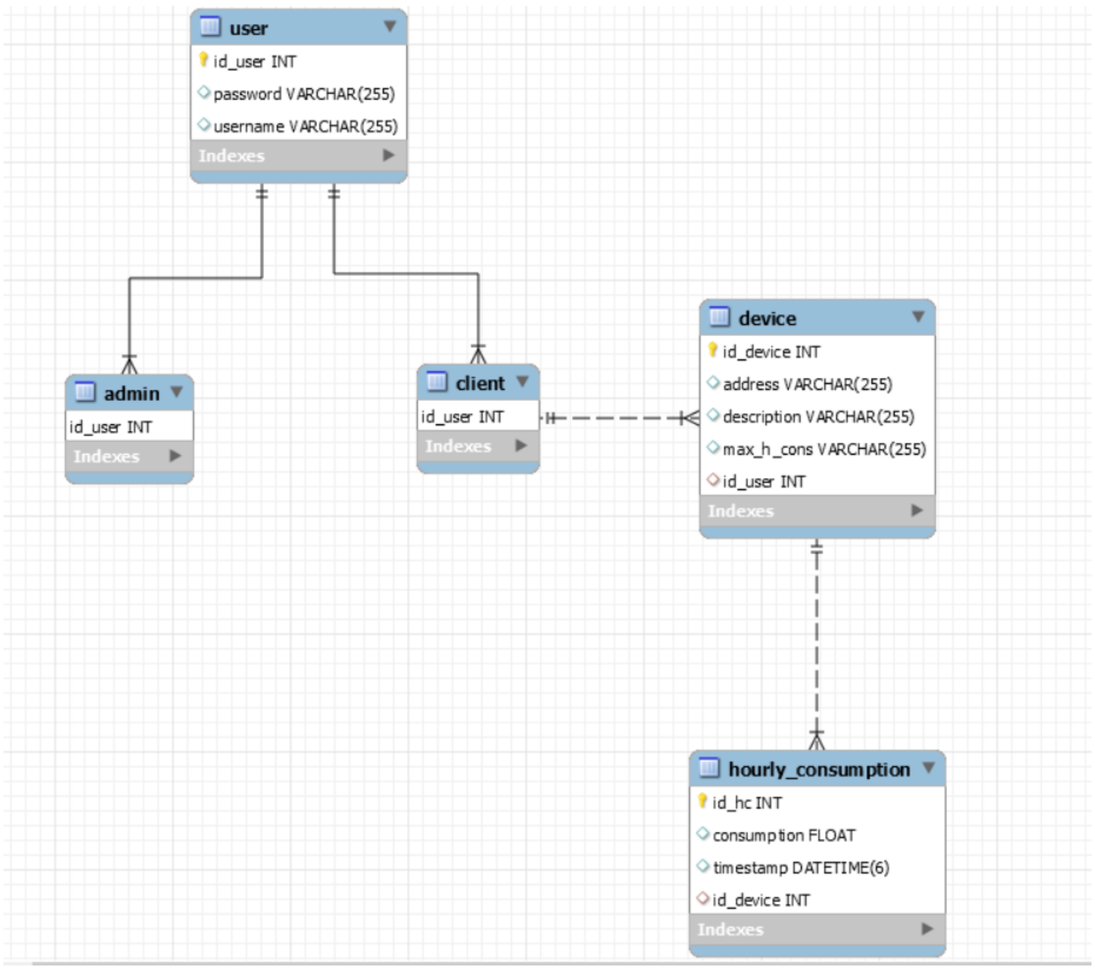
System Architecture Diagram



Class Diagram

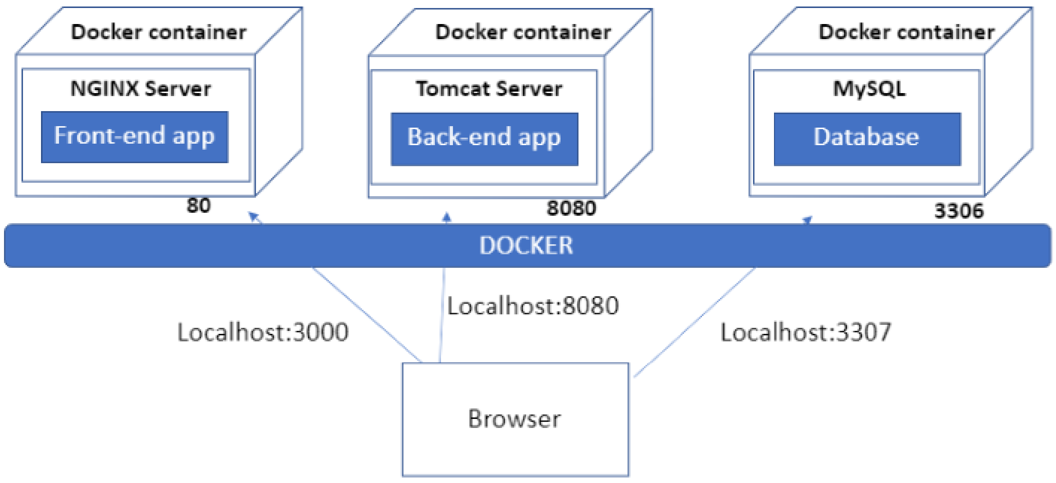


Database Diagram



Deployment

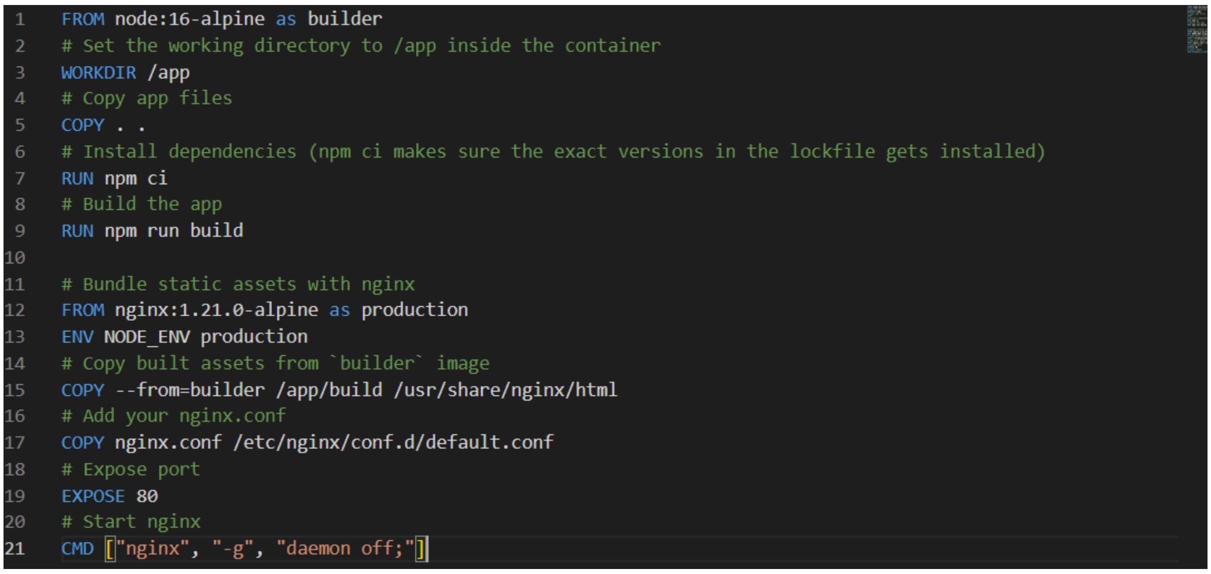
All the components of the application are deployed locally on docker containers. Docker is a container-based technology where containers are running as processes in the user space of the operating system. The React frontend part is deployed in one container and the Mysql database and the Spring Boot Java application are together in another container.
Below is a generalized diagram of the docker architecture of the deployment.



Deploying the frontend React application

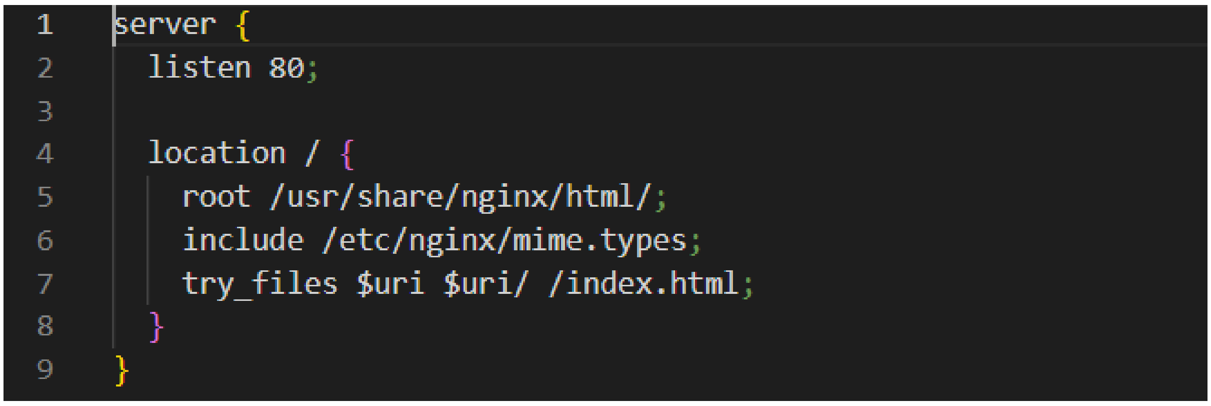
The first step I did was creating the Dockerfile and the nginx.conf file in the root folder of my react project. After that, I wrote the contents of the files as below:

Dokerfile



```
FROM node:16-alpine as builder
# Set the working directory to /app inside the container
WORKDIR /app
# Copy app files
COPY . .
# Install dependencies (npm ci makes sure the exact versions in the lockfile gets installed)
RUN npm ci
# Build the app
RUN npm run build

# Bundle static assets with nginx
FROM nginx:1.21.0-alpine as production
ENV NODE_ENV production
# Copy built assets from `builder` image
COPY --from=builder /app/build /usr/share/nginx/html
# Add your nginx.conf
COPY nginx.conf /etc/nginx/conf.d/default.conf
# Expose port
EXPOSE 80
# Start nginx
CMD ["nginx", "-g", "daemon off;"]
```

nginx.conf

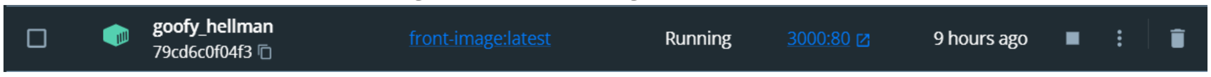


```
server {
  listen 80;

  location / {
    root /usr/share/nginx/html/;
    include /etc/nginx/mime.types;
    try_files $uri $uri/ /index.html;
  }
}
```

After the files were created, I run the following commands:

- docker build . -t image-front
- docker run -p 3000:80 -d image-front

And the container from the image started running as seen below

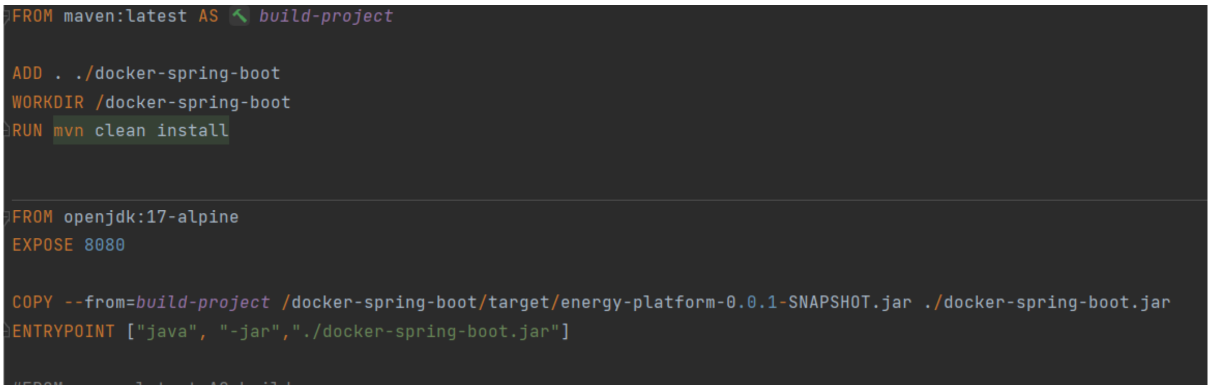


```
goofy_hellman    front-image:latest    Running    3000:80    9 hours ago
79cd6c0f04f3
```

Deploying the database and the Spring Boot app

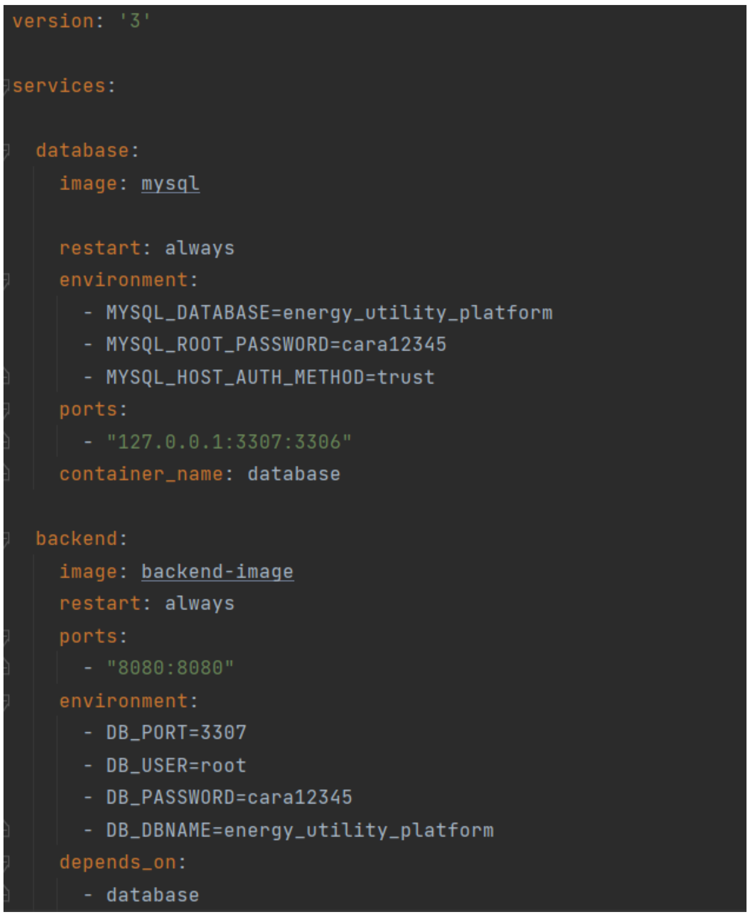
The first step I did was to create the Dockerfile and the docker-compose.yml file in the root folder of my Spring Boot application. The contents of the files can be seen below.

Dockerfile

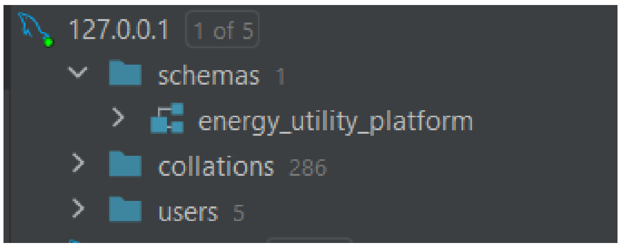


```
FROM maven:latest AS build-project

ADD . ./docker-spring-boot
WORKDIR /docker-spring-boot
RUN mvn clean install


FROM openjdk:17-alpine
EXPOSE 8080

COPY --from=build-project /docker-spring-boot/target/energy-platform-0.0.1-SNAPSHOT.jar ./docker-spring-boot.jar
ENTRYPOINT ["java", "-jar","./docker-spring-boot.jar"]
```

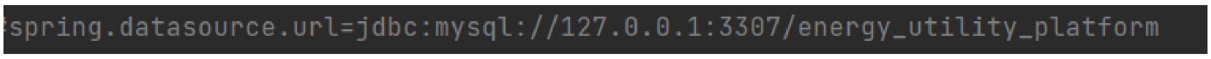
docker-compose.yml


```yaml
version: '3'

services:

  database:
    image: mysql

    restart: always
    environment:
      - MYSQL_DATABASE=energy_utility_platform
      - MYSQL_ROOT_PASSWORD=cara12345
      - MYSQL_HOST_AUTH_METHOD=trust
    ports:
      - "127.0.0.1:3307:3306"
    container_name: database

  backend:
    image: backend-image
    restart: always
    ports:
      - "8080:8080"
    environment:
      - DB_PORT=3307
      - DB_USER=root
      - DB_PASSWORD=cara12345
      - DB_DBNAME=energy_utility_platform
    depends_on:
      - database
```

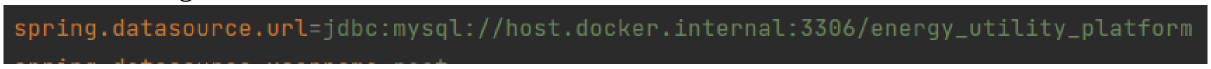
Then I created a new instance of data source in myqsl on port 3307 with host 127.0.0.1 through which the connection to the database will be made



Then I run the maven component to generate the jar of the application(the jar that is included in the Dockerfile). After generating the jar, I changed the data source connection in the app properties file from the local one :


```
spring.datasource.url=jdbc:mysql://127.0.0.1:3307/energy_utility_platform
```

to the following:

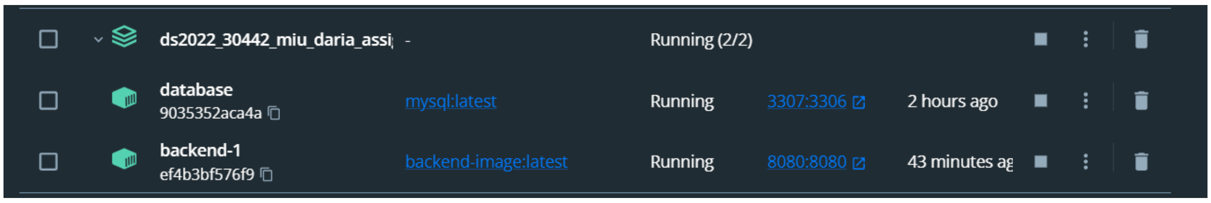

```
spring.datasource.url=jdbc:mysql://host.docker.internal:3306/energy_utility_platform
```

The next step was running the following commands in the terminal:

-docker-compose up database

```
-docker build . -t backend-image
-docker-compose up -d
```

And finally, the database and the backend image can be seen running in the docker container as below:



ds2022_30442_miu_daria_assi	-	Running (2/2)		
database 9035352aca4a	mysql:latest	Running	3307:3306	2 hours ago
backend-1 ef4b3bf576f9	backend-image:latest	Running	8080:8080	43 minutes ag