Distributed Systems

Assignment 2

Asynchronous Communication

Sensor Monitoring System and Real-Time Notification

Ghise Nicolae-Mihai, Group 30443

1. Problem description

Implement a component for Assignment 1 application based on a message broker middleware that gathers data from the smart metering devices, pre-processes the data to compute the hourly energy consumption and stores it in the database.

A smart Metering Device Simulator module will be the Message Producer. It will simulate a sensor by reading energy data from a csv file given to us by the professor. The simulation required data from the file every 10 minutes and at the end of the hour to see if the consumption of that device during that hour exceeds its maximum energy consumption. If so, a notification will be pushed on the front end.

2. Technologies and solutions

The message broker middleware that I will be using is RabbitMQ, and in order to be able to see that the application works, I will redesign the requirement which states every 10 minutes and when the hour ends to every 10 seconds and when the minute ends. So basically, I will do the simulation in a smaller timeframe, and I will be reading that faster.

Every time I read a new value from the file; I verify if the current timestamp's minute is different from the last value's timestamp's minute. If so, I will do some calculations using two variables, one in which I store the current consumptions since start reading the file, and one variable where I store the consumption when it was the last change of minutes.

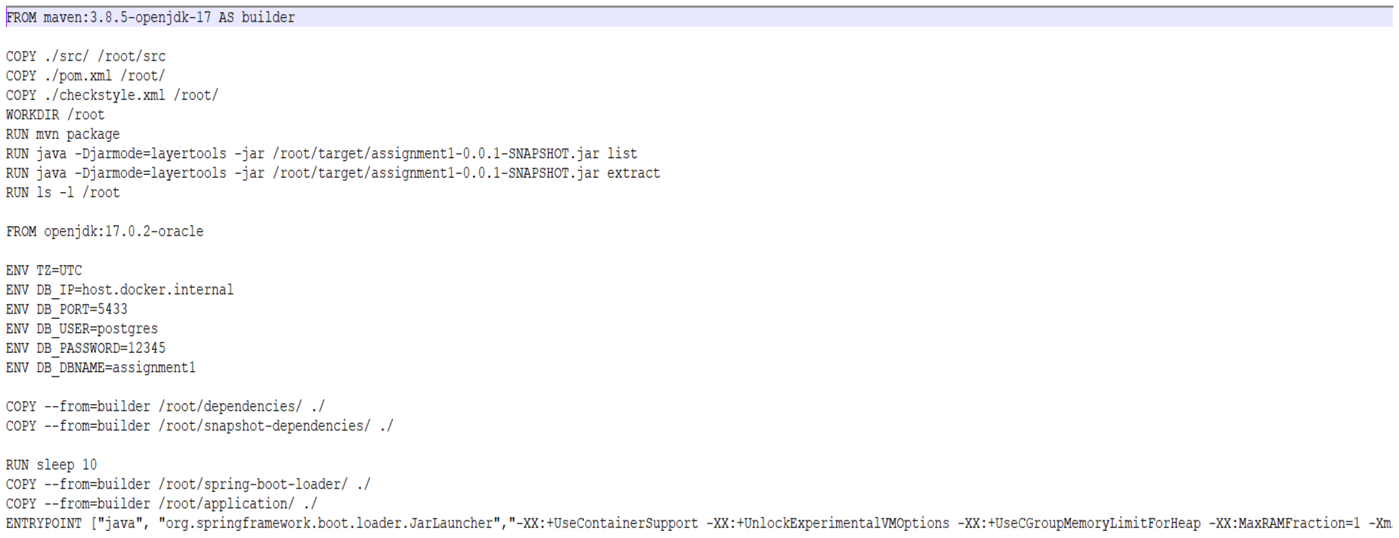
3. Functional requirements

In order to achieve the goals of the assignment, I used a sender which will read from the file and send the data to the queue, and a receiver for each device which will take the data from the queue and will verify it. In order to be able to connect the first application from the first assignment to the new application which contains RabbitMQ and the sender I used a connection factory. The receiver is obviously in the first application.

4. Deployment on docker

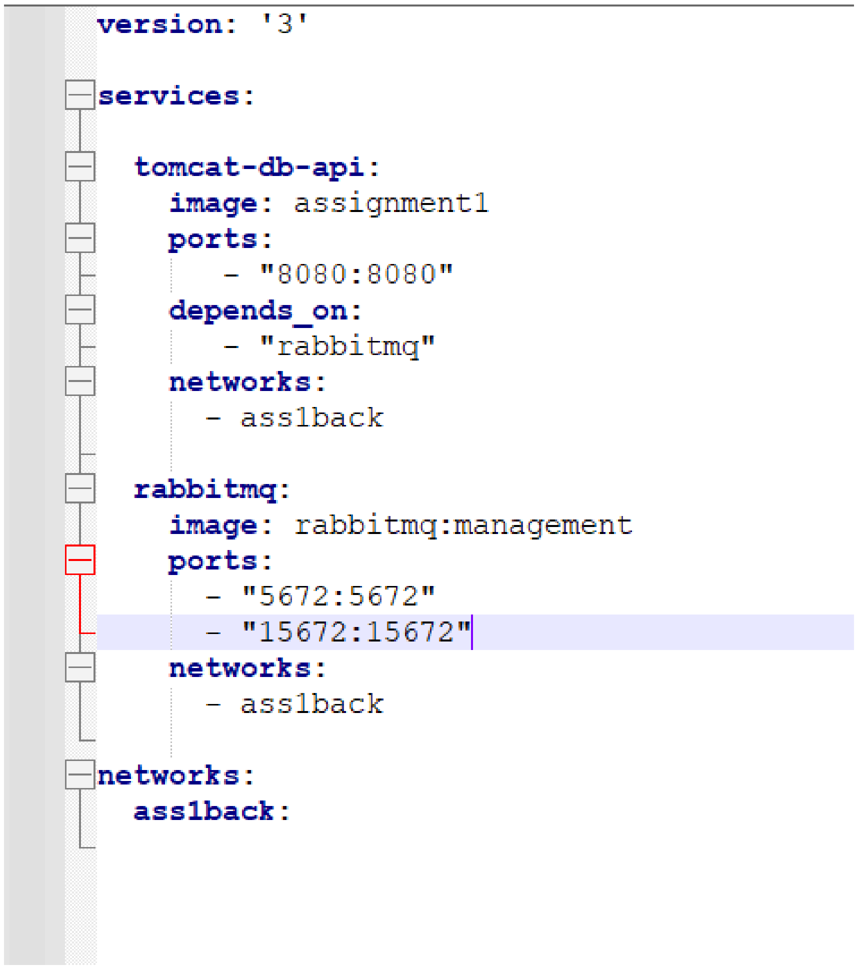
Deployment of RabbitMQ inside first assignment

Docker file (first assignment including receivers + RabbitMQ):



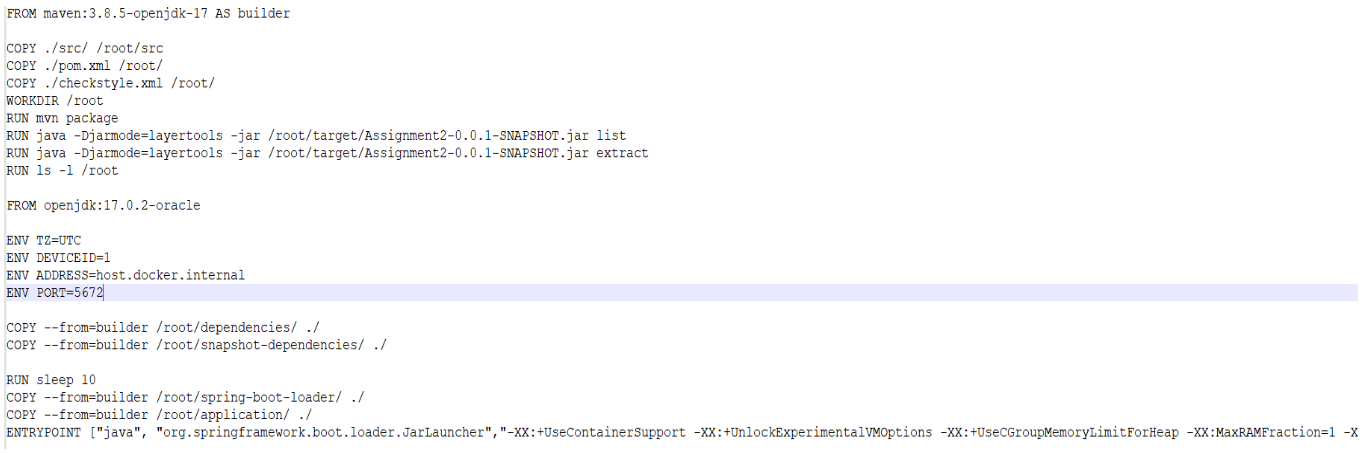
```
FROM maven:3.8.5-openjdk-17 AS builder

COPY ./src/ /root/src
COPY ./pom.xml /root/
COPY ./checkstyle.xml /root/
WORKDIR /root
RUN mvn package
RUN java -Djarmode=layertools -jar /root/target/assignment1-0.0.1-SNAPSHOT.jar list
RUN java -Djarmode=layertools -jar /root/target/assignment1-0.0.1-SNAPSHOT.jar extract
RUN ls -l /root

FROM openjdk:17.0.2-oracle

ENV TZ=UTC
ENV DB_IP=host.docker.internal
ENV DB_PORT=5433
ENV DB_USER=postgres
ENV DB_PASSWORD=12345
ENV DB_DBNAME=assignment1

COPY --from=builder /root/dependencies/ ./
COPY --from=builder /root/snapshot-dependencies/ ./

RUN sleep 10
COPY --from=builder /root/spring-boot-loader/ ./
COPY --from=builder /root/application/ ./
ENTRYPOINT ["java", "org.springframework.boot.loader.JarLauncher","-XX:+UseContainerSupport -XX:+UnlockExperimentalVMOptions -XX:+UseCGroupMemoryLimitForHeap -XX:MaxRAMFraction=1 -Xm
```

Docker-compose file: I had to use a network otherwise my connection to RabbitMQ would not be established



```
version: '3'

services:

  tomcat-db-api:
    image: assignment1
    ports:
        - "8080:8080"
    depends_on:
        - "rabbitmq"
    networks:
      - ass1back

  rabbitmq:
    image: rabbitmq:management
    ports:
      - "5672:5672"
      - "15672:15672"
    networks:
      - ass1back

networks:
  ass1back:
```

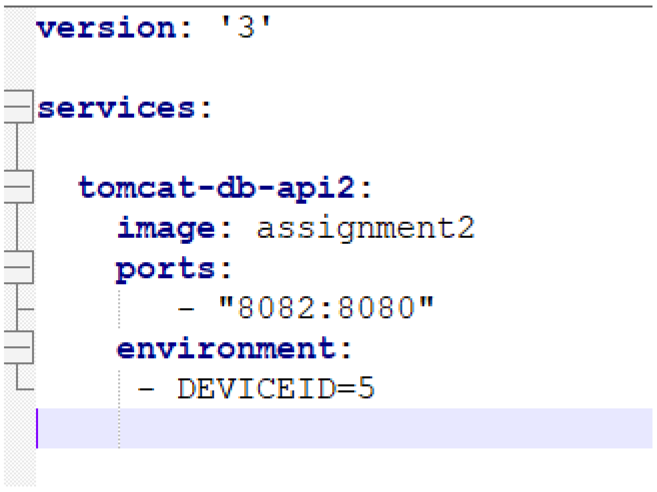
Deployment of the second assignment which contains the sender:



```
FROM maven:3.8.5-openjdk-17 AS builder

COPY ./src/ /root/src
COPY ./pom.xml /root/
COPY ./checkstyle.xml /root/
WORKDIR /root
RUN mvn package
RUN java -Djarmode=layertools -jar /root/target/Assignment2-0.0.1-SNAPSHOT.jar list
RUN java -Djarmode=layertools -jar /root/target/Assignment2-0.0.1-SNAPSHOT.jar extract
RUN ls -l /root

FROM openjdk:17.0.2-oracle

ENV TZ=UTC
ENV DEVICEID=1
ENV ADDRESS=host.docker.internal
ENV PORT=5672

COPY --from=builder /root/dependencies/ ./
COPY --from=builder /root/snapshot-dependencies/ ./

RUN sleep 10
COPY --from=builder /root/spring-boot-loader/ ./
COPY --from=builder /root/application/ ./
ENTRYPOINT ["java", "org.springframework.boot.loader.JarLauncher","-XX:+UseContainerSupport -XX:+UnlockExperimentalVMOptions -XX:+UseCGroupMemoryLimitForHeap -XX:MaxRAMFraction=1 -X
```

Docker-compose file for the sender: here I have to containers because I want to send data for two devices one with id 1 and one with id 5.

This is the docker compose file for the sender for device with id 5.



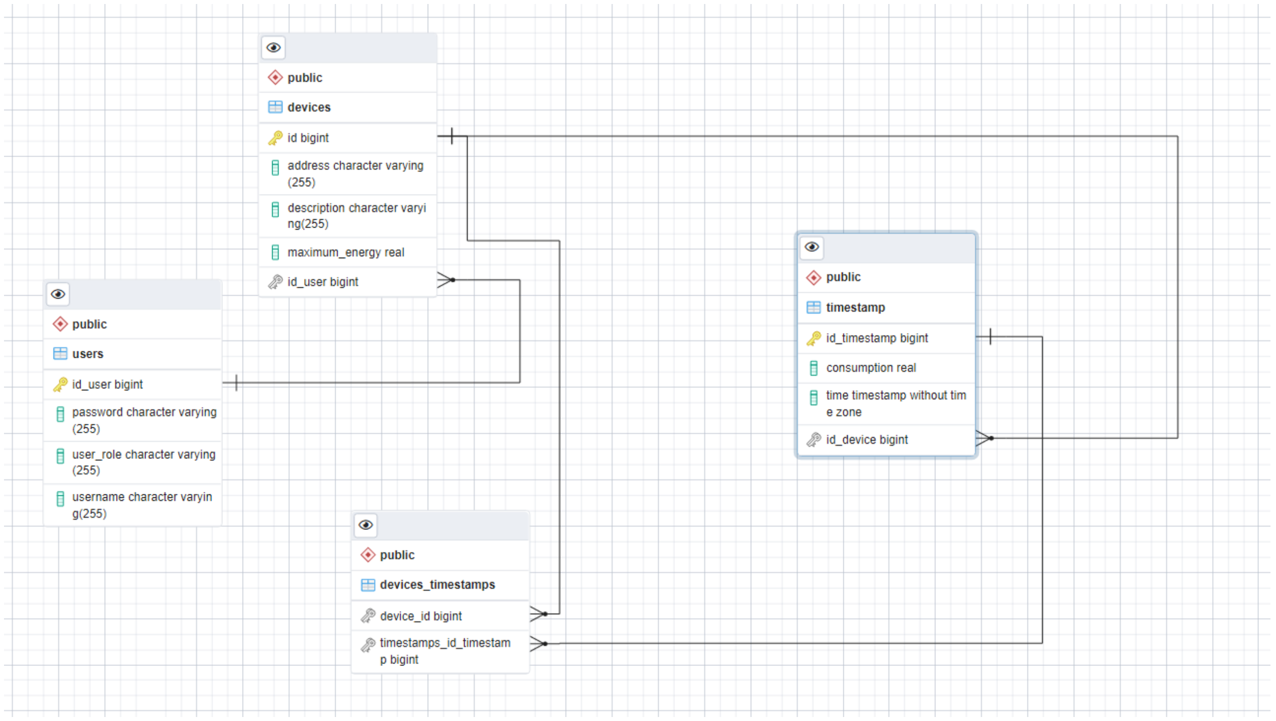
```
version: '3'

services:

  tomcat-db-api2:
    image: assignment2
    ports:
       - "8082:8080"
    environment:
     - DEVICEID=5
```

For every container that I will create I will run the following commands:

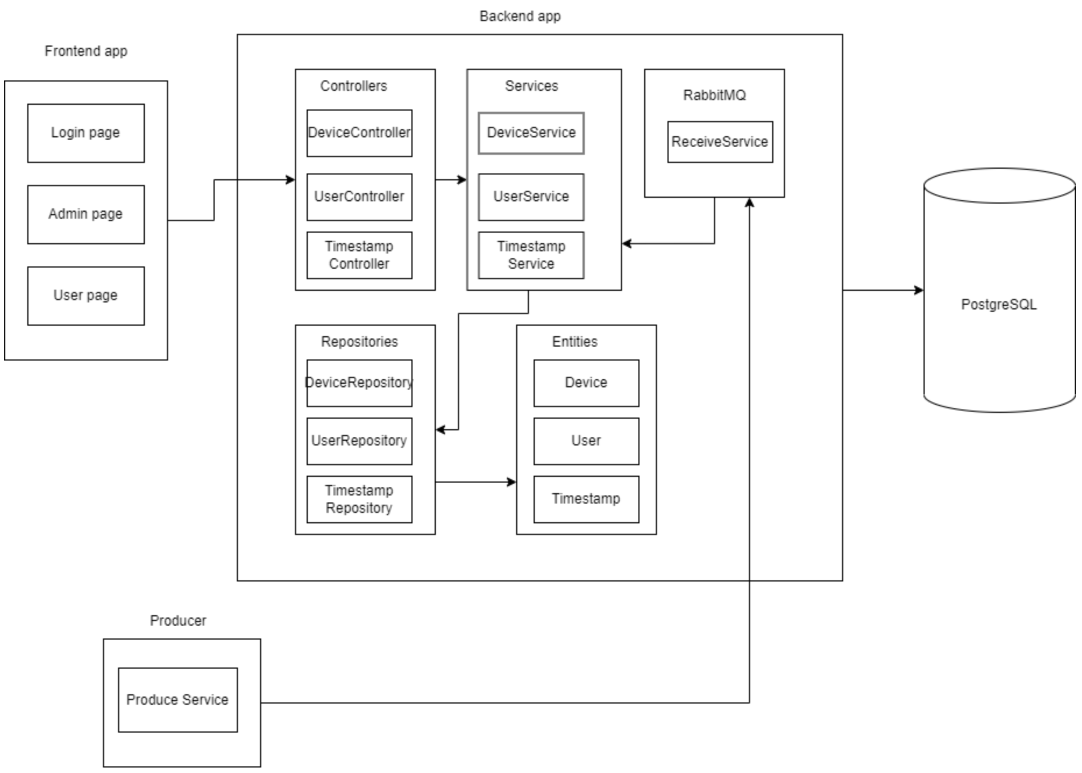
Docker build –t name .

Docker-compose up –d

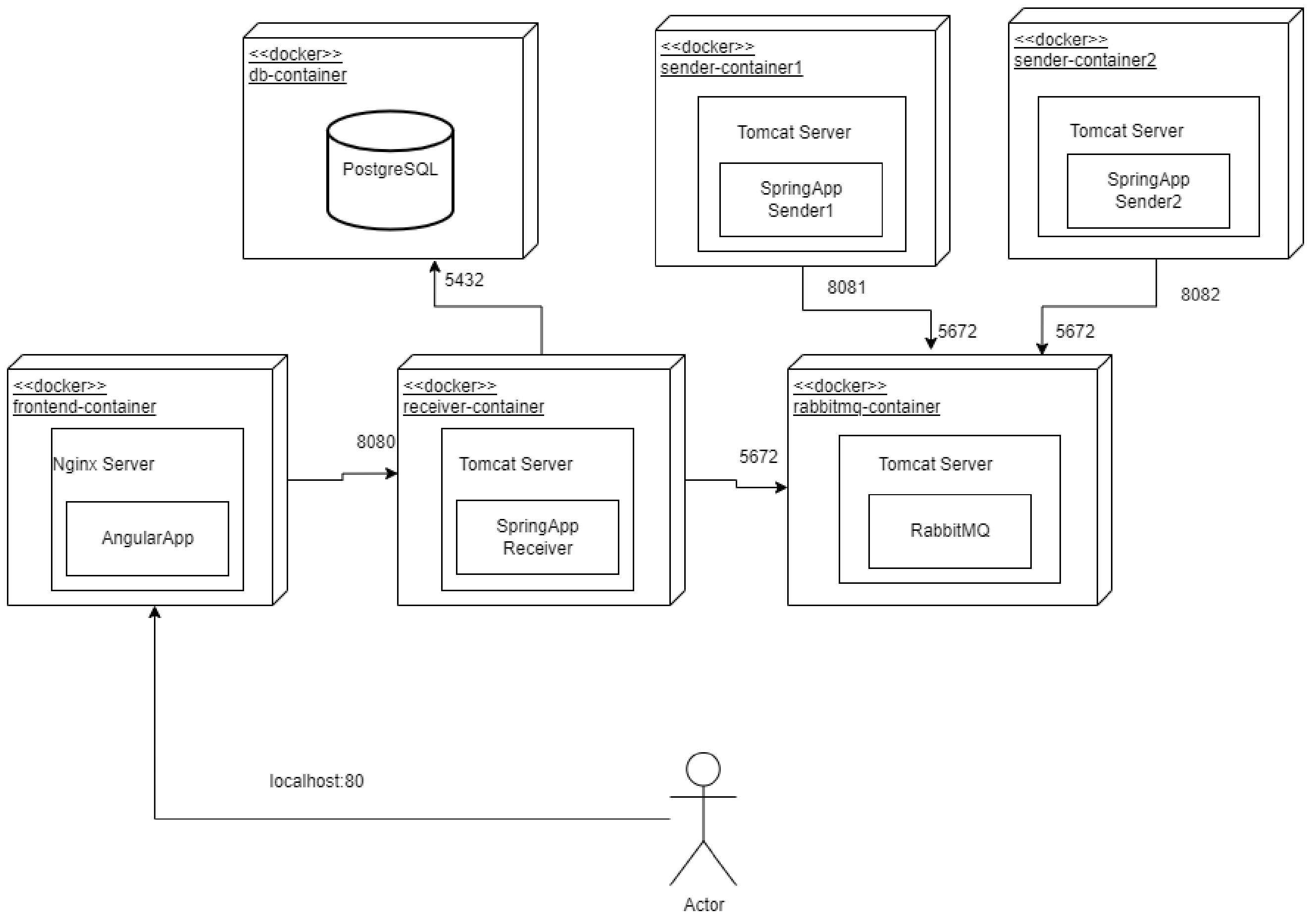
Database diagram:


public
devices
id bigint
address character varying (255)
description character varying(255)
maximum_energy real
id_user bigint
public
users
id_user bigint
password character varying (255)
user_role character varying (255)
username character varying(255)
public
timestamp
id_timestamp bigint
consumption real
time timestamp without time zone
id_device bigint
public
devices_timestamps
device_id bigint
timestamps_id_timestamp bigint

Conceptual diagram:



Deployment diagram:



<<docker>>
db-container
PostgreSQL
<<docker>>
sender-container1
Tomcat Server
SpringApp Sender1
<<docker>>
sender-container2
Tomcat Server
SpringApp Sender2
5432
8081
5672
8082
5672
<<docker>>
frontend-container
Nginx Server
AngularApp
8080
<<docker>>
receiver-container
Tomcat Server
SpringApp Receiver
5672
<<docker>>
rabbitmq-container
Tomcat Server
RabbitMQ
localhost:80
Actor