-Projekat 1:

Za samo pokretanje projekta potrebno je imati skinut VS 2019. Potrebno je u okviru Data\_Service-a >Service>SmartHomeRepository u 19-oj liniji koda promeniti putanju do mongo baze jer ce se na drugom racunaru pokrenuti pod drugim imenom. Posle toga se pokretanjem preko Docker Compose-a se sve automatski pokrece.

Projekat se sastoji od 2 mikroservisa:

1. Sensor Device Service

2. Data Service

Za proveru svih ruta postoji exportovan postman fajl koji treba da se importuje, medjutim kada se aplikacija pokrene na drugom racunaru doci ce do promene lokalnih portova koji se mapiraju na portove docker kontejnera tako da je potrebno da se komandom "docker ps -a" posle pokretanja aplikacije vidi koji se lokalni portovi koriste za pokretanje aplikacije i da se ti portovi upisu u odgovarajuce rute u postman-u.

Pored postman-a moguce je rute ispitati i pomocu swagger UI-a koji ce automatski da se pokrene za Data\_Service mikroservis i samo je potrebno da se vidi lokalni port koji se koristi za pokretanje Sensor\_Device\_Service mikroservisa i na taj nacin je moguce ispitati rute i za drugi mikroservis pomocu swagger UI-a.

**Sensor\_Device\_Service:**

Cita podatke iz Exel-a I poziva POST Metodu Sensor Device Servisa koja zatim upisuje u MongoDB

Takodje sadrzi I sledece metode:

**PUT /api/smartHome/interval** - Za promenu brzine kojom ce da se salju na drugi mikroservis u sekundama.

**POST /api/smartHome/sensorType -** Za promenu vrste podataka koji se salju drugom mikroservisu.

1 – Svi podaci

2 – Samo podaci vezani za proizvodnju I potrosnju struje

3 - Ostali podaci: temperature, vazdusni pritisak…

**GET /api/smartHome/interval -** Za dobijanja informacije na koliko sekundi se salju podaci.

**GET /api/smartHome/sensorType -** Za dobijanja informacije koji podaci se salju drugom mikroservisu

**Data\_Service:**

Prima podatke Od Sensor\_device\_service I upisuje ih u MongoDB

**POST /api/smartHomeData** - Za upis podataka sa senzora u bazi

**GET ​/api​/smartHomeData​/usage** - Za dobijanje informacija o potrosnji u odnosu na neku vrednost. Navode se 2 parametra u ruti use I grSmUse, ukoliko je grSmUse prazno traze se svi podaci ciji property use ima istu vrednost kao I use parametar iz rute, ukoliko se za grSmUse navede gr onda se traze vece vrednosti u odnosu na use a ako se navede sm traze se manje vrednosti.

**GET /api/smartHomeData/generated** - Za dobijanje informacija o generisanoj struji u odnosu na neku vrednost. Navode se 2 parametra u ruti gen I grSmGen, ukoliko je grSmGen prazno traze se svi podaci ciji property gen ima istu vrednost kao I gen parametar iz rute, ukoliko se za grSmGen navede gr onda se traze vece vrednosti u odnosu na gen a ako se navede sm traze se manje vrednosti.

**GET /api/smartHomeData/temperature** - Za dobijanje informacija o temperaturi u odnosu na neku vrednost. Navode se 2 parametra u ruti temp I grSmTemp, ukoliko je grSmTemp prazno traze se svi podaci ciji property temp ima istu vrednost kao I temp parametar iz rute, ukoliko se za grSmTemp navede gr onda se traze vece vrednosti u odnosu na temp a ako se navede sm traze se manje vrednosti.

**GET /api/smartHomeData/allData**  - Za dobijanje svih podataka iz MongoDB-a. Navode se 2 parametra u ruti from I to tj. koliko podataka da se procita.

-Projekat 2:

Sto se tice pokretanja projekta od kada su dodati servisi koji su zahtevani u drugom projektu potrebno je da se odrade svi koraci navedeni iznad koji se ticu prvog projekta pa onda dodatni koraci:

- Potrebno je u okviru Analytics\_Service >Service>AnalyticsRepository u 18-oj liniji koda promeniti putanju do mongo baze jer ce se na drugom racunaru pokrenuti pod drugim imenom.

- Pre pokretanja aplikacije u Visual Studio-u potrebno je da se pokrene u docker-u aplikacija koja je kreirana pomocu “Siddhi” platforme, potrebno je u konzoli izvrsiti sledece komande :

- **docker pull siddhiio/siddhi-runner-alpine:latest**

- **docker run -it -p 8006:8006 -v <Apsolutna putanja/ TemperatureApp.siddhi >:/siddhi-runner/TemperatureApp.siddhi siddhiio/siddhi-runner-alpine -Dapps=/siddhi-runner/TemperatureApp.siddhi**

- primer apsolutne putanje: **C:\Users\Goran\Servisno\_Orijentisane\_Arhitekture\_Projekti\SOA\SOA\SOA\_PRO JEKAT\_1\SOA\_PROJEKAT\_1\Analytics\_Service\Siddhi\TemperatureApp.siddhi**

Kada se pokrene ova aplikacija potrebno je samo u okviru projekta u Visual Studio-u u okviru mikroservisa “Analytics\_Service” u klasi “DataReader”promeniti u 85-oj liniji koda URL koji se koristi za pozivanje Siddhi aplikacije, potrebno je staviti konkretnu adresu IPv4 za datu masinu a koju je moguce pronaci kucanjem komande u konzoli “ipconfig”. Kada se izvrse prethodni koraci samo treba pokrenuti aplikaciju u Visual Studio-u pomocu Docker Compose-a.

-Projekat 3:

Pre pokretanja trebaju se ispostovati svi koraci navedeni u odeljcima iznad za projekat 1 I 2 I aplikacija se moze pokrenuti preko Docker Compose-a. Posle pokretanja aplikacije potrebno je pokrenuti dashboard koji je zahtevan u ovom projektu. Pre pokretanja dashboard-a potrebno je u WebDashboard>src>app>API>api.ts promeniti port na kome je pokrenut API\_Gateway\_Service. Dashboard je kreiran pomocu Angular-a tako da je za pokretanje potrebno pozicioniradi se u konzoli do foldera gde se on nalazi I izvrsiti komande “npm install” I posle toga “npm run start”. Pomocu dashboard-a mozemo aktivirati odredjene akcije I nadgledati podatke.

Za proveru svih ruta postoji exportovan Postman fajl koji treba da se importuje, medjutim kada se aplikacija pokrene na drugom racunaru doci ce do promene lokalnih portova koji se mapiraju na portove docker kontejnera tako da je potrebno da se komandom "docker ps -a" posle pokretanja aplikacije vidi koji se lokalni portovi koriste za pokretanje aplikacije i da se ti portovi upisu u odgovarajuce rute u postman-u.

Pored postman-a moguce je rute ispitati i pomocu swagger UI-a koji ce automatski da se pokrene za Data\_Service mikroservis i samo je potrebno da se vidi lokalni port koji se koristi za pokretanje API\_Gateway\_Service mikroservisa i na taj nacin je moguce ispitati rute I pomocu swagger UI-a.

Sad ace biti opisane rute koje je moguce korostiti.

**API\_Gateway\_Service:**

**POST api/dataGateway** - Za dodavanje podataka u bazi.

**GET api/dataGateway/usage?use=1&grSmUse=gr -** Za dobijanje informacija o potrosnji u odnosu na neku vrednost. Navode se 2 parametra u ruti use I grSmUse, ukoliko je grSmUse prazno traze se svi podaci ciji property use ima istu vrednost kao I use parametar iz rute, ukoliko se za grSmUse navede gr onda se traze vece vrednosti u odnosu na use a ako se navede sm traze se manje vrednosti.

**GET api/dataGateway/generated?gen=1&grSmUse=gr -** Za dobijanje informacija o generisanoj struji u odnosu na neku vrednost. Navode se 2 parametra u ruti gen I grSmGen, ukoliko je grSmGen prazno traze se svi podaci ciji property gen ima istu vrednost kao I gen parametar iz rute, ukoliko se za grSmGen navede gr onda se traze vece vrednosti u odnosu na gen a ako se navede sm traze se manje vrednosti.

**GET api/dataGateway/temperature?temp=1&grSmUse=gr -** Za dobijanje informacija o temperaturi u odnosu na neku vrednost. Navode se 2 parametra u ruti temp I grSmTemp, ukoliko je grSmTemp prazno traze se svi podaci ciji property temp ima istu vrednost kao I temp parametar iz rute, ukoliko se za grSmTemp navede gr onda se traze vece vrednosti u odnosu na temp a ako se navede sm traze se manje vrednosti.

**GET api/dataGateway/allData?from=0&to=5 -** Za dobijanje svih podataka iz MongoDB-a. Navode se 2 parametra u ruti from I to tj. koliko podataka da se procita.

**POST api/commandGateway/setTimeInterval -**  Za promenu brzine kojom ce da se salju podaci sa senzora u sekundama.

**GET api/commandGateway/getTimeInterval -** Za dobijanja informacije na koliko sekundi se salju podaci sa senzora.

**GET api/commandGateway/getLastAction -** Vraca nam poslednju akciju koja je zadata sa Command mikroservisa Sensor\_Device mikroservisu.