Dokumentacija SOA projekta

Uvod

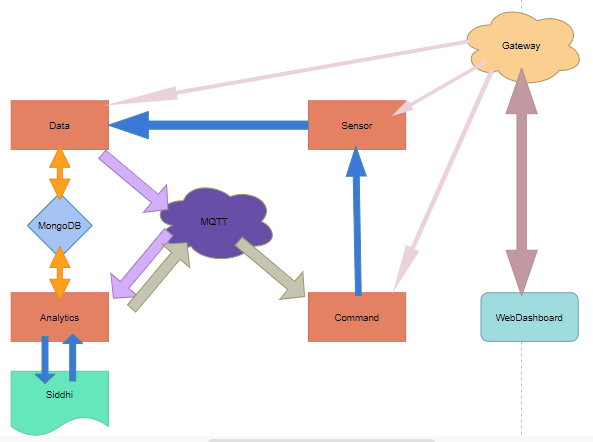
U ovom projektu kreirana je mikroservisna aplikacija za detektovanje tornadoa I informacija o njima. Virtuelni senzor koristi podatke sa podrucja SAD-a u periodu od 1950. do 2019. godine. Sistem je projektovan pomocu mikroservisne arhitekture koja se koristi za razvoj IoT aplikacija.

Projekat se sastoji od 9 mikroservisa:

1. Sensor microservice
2. Data microservice
3. Analytics microservice
4. Siddhi microservice
5. Command microservice
6. Gateway microservice
7. Mongo microservice
8. MQTT microservice
9. Web dashboard

Za back-end servise koriscen je Node.Js framework Moleculerjs a za front-end koriscen je React. Za komunikaciju izmedju backend servisa koriscen je MQTT, za komunikaciju izmedju Analytics mikroservisa I Siddhi mikroservisa koriscen je HTTP. Dodatno Command mikroservis salje notifikacije WebDahsboard-u preko soketa koriscenjem modula Socket.io. Kao baza koriscen je MongoDB od strane Data I Analytics mikroservisa.

Arhitektura



1. Sensor microservice

Ovaj mikroservis sastoji se od senzora I virtuelnog aktuatora. Sensor simulira realan senzor tako sto periodicno cita podatke iz fajla I prosledjuje ih Data servisu. Konfiguracija senzora I aktuatora se pamti u sensor.config,json fajlu. Moguce je menjati parametre ocitavanja senzora.

**Endpointi:**

GET /api/sensor/metaData => sensor.getMetaData – Za pribavljanje meta podataka o senzoru

POST /api/sensor/metaData => sensor.postMetaData – Za promenu meta podataka o senzoru

PUT /api/sensor/metaData => sensor.putMetaData – Za promenu meta podataka o senzoru

POST /api/sensor/executeCommand => sensor.executeCommand – Za izvrsenje komande na virtuelnom aktuatoru

Sensor servis periodicno prosledjuje podatke Data servisu preko REST POST pozivanjem data.postData akcije.

1. Data microservice

Ovaj mikroservis prihvata ocitane podatke od Sensor mikroservisa i pamti ih u bazi podataka.

**Endpointi:**

GET /api/data => data.list

POST /api/data => data.create

GET /api/data/:id => data.get

PUT /api/data/:id => data.update

DELETE /api/data/:id => data.remove

POST /api/data/Data => data.postData –Za upis podataka u bazu

GET /api/data/Data => data.getData –Za pretrazivanje baze podataka po razlicitim kriterijumima

Data servis prosledjuje podatke Analytics servisu preko MQTT tako sto podatke publish-uje na analytics.getData.

1. Analytics microservice

Ovaj mikroservis prihvata podatke od Data servisa, prosledjuje ih eksternom Siddhi servisu, prihvata rezultate obrade od Siddhi-ja I u zavisnosti od njih publish-uje odredjenu komandu Command mikroservisu.

**Endpointi:**

GET /api/analytics => analytics.list

POST /api/analytics => analytics.create

GET /api/analytics/:id => analytics.get

PUT /api/analytics/:id => analytics.update

DELETE /api/analytics/:id => analytics.remove

POST /api/analytics/CEPResponse => analytics.CEPResponse – Za prihvatanje rezultata od Siddhi-ja

Analytics servis prosledjuje podatke Command servisu preko MQTT tako sto podatke publish-uje na injured-alert, fatalities-alert, loss-alert i damage-alert u zavisnosti od detektovanih dogadjaja.

1. Siddhi microservice

Eksterni mikroservis slusa na adresi: <http://siddhi:8006/analytics> zahteve od Analytics servisa, vrsi obradu podataka I vraca generisane dogadjaje preko http zahteva Analytics mikroservisu.

**Dogadjaji koji se detektuju:** Injuries, fatalities, loss i damage.

1. Command microservice

Ovaj mikroservis salje notifikacije o detektovanim dogadjajima WebDashboard-u preko socketa (Socket.io). Takodje salje komande senzor servisu koji treba da ih izvrsi na virtuelnom aktuatoru preko REST POST zahteva. Moguce komande: injuried, fatalities, loss I damage. Takodje Command mikroservis salje komande senzoru kada stignu zahtevi za dobijanje I promenu meta podataka Sensor servisa.

**Endpointi:**

POST /api/command/injuredCommand => command.injuredCommand

POST /api/command/fatalitiesCommand => command.fatalitiesCommand

POST /api/command/lossCommand => command.lossCommand

POST /api/command/damageCommand => command.damageCommand

GET /api/command/getMetaDataCommand => command.getMetaDataCommand

POST /api/command/postMetaDataComand => command.postMetaDataCommand

PUT /api/command/putMetaDataCommand => command.putMetaDataCommand

1. Gateway microservice

Ovaj mikroservis koristi APIGateway servis. Koristi se za rutiranje zahteva od strane klijenta ka drugim servisima u sistemu.

**Endpoint:**

GET /api/gateway/list-aliases => gateway.listAliases

1. Mongo microservice

Ovaj mikroservis koristi MongoDB za skladistenje podataka I njemu pristupaju Data I Analytics mikroservisi.

1. MQTT microservice

Ovaj mikroservis koristi se za komunikaciju izmedju ostalih mikroservisa. Koristi MQTT protokol koji se zasniva na mosquitto mqtt transporteru. Pokrece se na portu 1883.

1. Web dashboard

Predstavlja klijenta I zasniva se na REACT tehnologiji. Pokrece se na portu 3000 ili 3001. Pokrece se u browseru I moze se koristiti za pretragu ocitanih podataka, dobijanje I promenu meta podataka Sensor servisa I za osluskivanje I prikaz detektovanih dogadjaja koriscenjem soketa.

Pokretanje I Testiranje

Sistem se moze pokrenuti lokalno, prvo back-end, pa Siddhi mikroservis pa zatim front-end, ili u okviru Docker kontejnera koriscenjem Docker-compose skripte.

**Lokalno pokretanje:**

Pre izvrsenja sledecih naredbi potrebno je I u back-end I front-end aplikaciji izvrsiti komandu npm install kako bi se prikupili svi neophodni paketi za rad aplikacija.

Pozicioniramo se u folder SOA\_Projekat I otvorimo terminal. Pokretanjem naredbe **npm run dev** pokrecu se svi bekend mikroservisi osim Siddhi. Da bi se pokrenuo Siddhi potrebno je skinuti Siddhi tooling program I pokrenuti tooling.bat skriptu iz terminala. Nakon pokretanja skripte Siddhi editoru se moze pristupiti na adresi **localhost:9390/editor.** U editoru importujemo Analytics.siddhi skriptu I pokrecemo je komandom run. Da bi se pokrenuo front-end servis potrebno je pozicionirati se u webdashboard folder I pokrenuti naredbu npm start. Front-end se moze koristiti iz browsera preko adrese localhost:3000( ili eksplicitnim podesavanjem porta u okviru start dela u package.json fajlu).

**Pokretanje iz Dockera:**

Pokretanje Dockera po prvi put vrsi se komandom **docker-compose up –build**. Ukoliko su vec kreirane slike, sistem se moze pokrenuti samo uz pomoc komande **docker-compose up**. Nakon pokretanja gateway servis je dostupan na adresi [**http://localhost:4000/api**](http://localhost:4000/api)**,**  siddhi je dostupan na adresi [**http://localhost:8006/analytics**](http://localhost:8006/analytics) a React aplikacija je dostupna na adresi [**http://localhost:3001**](http://localhost:3001).

**Testiranje servisa:**

Testiranje se moze vrsiti kroz postman kolekciju koja se nalazi u root direktorijumu projekta i

zove se SOA.postman\_collection.json. Ova kolekcija se moze jednostavno importovati kroz

postman.