# siberian weather station

Mikroservisna aplikacija za posmatranje meteoroloskog stanja neke oblasti. U ovom konkretnom slucaju je aplikacija iskoriscena u sibiru.

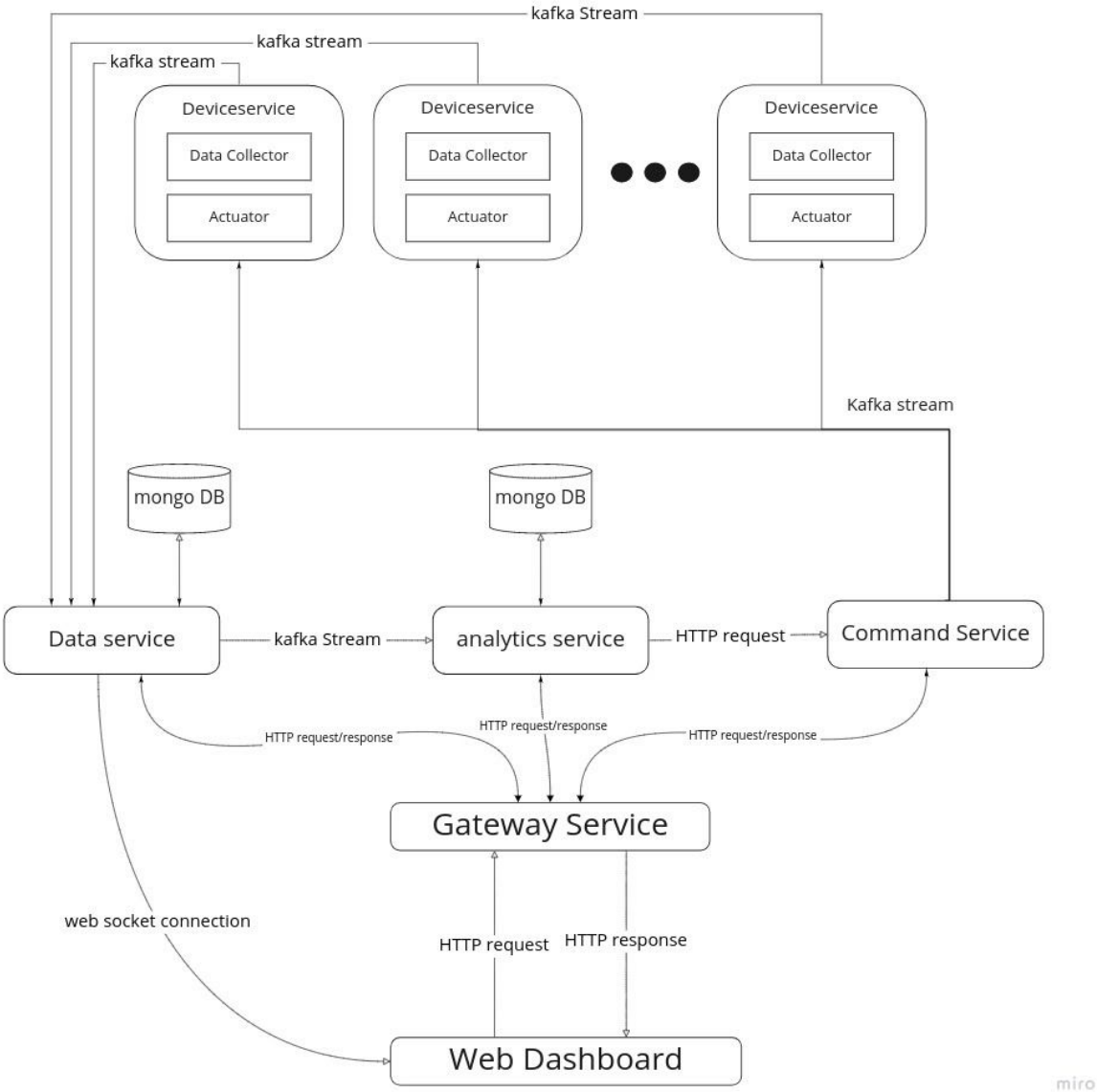
## Arhitektura

Ceo sistem je podeljen na 5 backend mikroservisa napisani u java spring frameworku :

- Device Service
- Data Service
- Analytics Service
- Command service
- Gateway service

I jedne frontend aplikacije napisane pomocu angular frameworka :

- Web dashboard


kafka Stream
kafka stream
kafka stream
Deviceservice
Data Collector
Actuator
Deviceservice
Data Collector
Actuator
Deviceservice
Data Collector
Actuator
Kafka stream
mongo DB
mongo DB
Data service
kafka Stream
analytics service
HTTP request
Command Service
HTTP request/response
HTTP request/response
HTTP request/response
Gateway Service
web socket connection
HTTP request
HTTP response
Web Dashboard
miro

## Device Service

Ovaj servis u sebi sadrzi dve komponente:

- Data Collector
    - Prima podatke sa jednog senzora i salje ih kroz kafka stream data mikroservisu
    - Podaci se citaju iz fajla umesto sa pravog senzora
- Actuator
    - Jedan aktuator koj moze da prima komande preko http post ili preko kafka streama

Servis iz envirament variabla cita koju meteorolosku vrednost meri( temperatura, pritisak, jacina vetra …), i o kom tipu aktuatora se radi. Moguce je imati vise tipova aktuatora ali je za sada implementirana samo jedan, sirena za uzbunu.

Takodje je moguce podesiti i frekfenciju ocitavanja podataka, slanjem zahteva na odgovarajuci endpoint.

Endpointi:

GET `/SensorMetaData`
Sluzi za dobijanje informacija o senzorima, kao sto su frekfencija ocitavanja i felicina bacha koja se salje data servicu

POST `/SensorMetaData`
Sluzi za postavnjenje parametara ocitavanja vrednosti sa senzora

GET `/ping`
Sluzi za postavnjenje parametara ocitavanja vrednosti sa senzora sluzi za proveru aktivnosti servisa

GET `/actuator`
Sluzi za dobijanje informacija o aktuatoru, kao sto su tip aktuatora, i lista podrzanih komandi koje aktuator moze da pokrene

POST `/actuator`
Sluzi za pokretanje jedne komande. Request body treba da sadrzi parametar commandName tipa string i parametar commandArgumetns koj je lista stringova

## Data Service

Prima podatke iz data service i cuva ih u mongo bazu podataka. Za svaku vrstu merenja postoji po jedan kafka stream( temperatura, pritisak, jacina, wetra…). Podaci se zatim salju dalje analitics servisu kroz drugi kafka stream.

Takodje ima otvoren i jedan web socket na kome se kontinualno salju podaci kako bi bili prikazani na web dashboardu.

Endpointi:

GET `/ping`
Sluzi za postavnjenje parametara ocitavanja vrednosti sa senzora sluzi za proveru aktivnosti servisa

## Analytics Service

Njegova uloga je da prima podatke sa data servisa, koji stizu preko kafka streama, i da ih analizira radi otkrivanja neuobicajnih vrednosti. Svaki put kada se otkriva odstupanje od normalne vrednosti salje se preko web soketa informacija web dashboardu da je doslo do odstupanja i takodje se salje signal command servisu da treba da odreaguje

On takodje poseduje svoju mongo bazu za belezenje svake neuobicajne vrednost. Moguce je pretraziti bazu kroz odgovarajuci endpoint

Enpoints:

GET `/ping`
Sluzi za postavnjenje parametara ocitavanja vrednosti sa senzora sluzi za proveru aktivnosti servisa

GET `/event`
Pribavlja listu svih eventova, svaki event sadrzi informacije o tome koj je tip izmerene vrednosti, koja je vrednost u pitanju i datum merenja

## Command Service

Uloga command servica je da salje commande aktivnim acutatorima u systemu.

Endpoint:

GET /`command`
Vraca informacije o dostupnim komandama u systemu, on to cini tako sto salje prvo zahtev gateway servisu da otkrije jedan od aktivnih aktuatora i onda tom aktuatoru salje http zahtev da pribavi listu njegoovih commandi

POST /`command`
Salje svim aktuatorima u systemu, pomocu kafka streama, signal da izvrse odredjenu komandu, ta komanda se dobija kao request body i sadrzi inforamcije o nazivu komande i o listi parametara

## Gateway service

Ovaj servis sluzi za pristup ostalim servisima i skoro svaki njegov endpoint poziva odgovarajuci endpoint na nekom drugom servisu

Takodje sluzi i kao naming servis, tj obezbedjuje ostalim servisima mogucnos da zatraze adresu nekog drugog servisa. Da bi to bilo ostvareno svaki servis mora sebe registrovti kada se pokrene, to cini tako sto salje poruku sa svojim informacijama, kao sto su ip adresa i port, na odgovarajuci kafka stream. Na drugom kraju kafka streama gatway servis slusa i belezi svaki novopridosli servis. Da bi se izbacivali servisi koji su se namerno ili nenamerno ugasili i time vise nisu dostupni, gateway servis periodicno proverava dali su svi servisi iz njegove mape servisa aktivni tako sto svakome salje HTTP GET zahtev na /`ping` adresi, ako servis odgovara sa “pong” znaci da je aktivan i spreman da obradjuje zahteve, u suprotnom se izbacuje iz registra svih aktivnih servisa.

Endpointi:

GET /`naming/actuatorInfo`
Vraca informacije o jednom aktivnom aktuatoru u sistemu, ovaj zahtev se prosledjuje actuator servisu i vraca se njegov odgovor

GET /`naming/services`
Vraca informacije o svim aktivnim servisima u systemu

GET /`naming/commandService`
Vraca informacije o command serviseu, ovaj endpoint se koristi od strane analytics serviisa kako bi znao gde da salje zahtev za pokretanje commande

GET /`naming/events`

Vraca informacije o svim eventima koje je analytics servis zabelezio, ovaj zahtev se prosledjuje analytics servisu

GET /naming/command
Vraca informacije o commandama koje acutatri systema mogu da izvrse, ovaj zahtev se prosledjuje command servisu

POST /naming/command
Prima, kao request body, jednu commandu i prosledjuje je command servisu

# Pokretanje i testiranje

Pokretanje svih servisa je moguce pomocu shell scripta koje se nalaze u folderu docerscripts. Prvo se mora pokrenuti skrpita pod imenom kafka_mongo_startup.sh a zatim i gateway_servic_startup.sh. Nakon toga se mogu pokrenuti i sve ostale skripte prozvoljnim redosledom

Testiranje se moze vrsiti kroz postman kolekciju koja se nalazi u root direktorijumu projetka i zove se SOA.postman_collection.json. Ova kolekcija se moze jednostavno iportovati kroz postman