# Elektronski fakultet Niš

Servisno-orijentisane arhitekture

Mainflux

# Mentor: Student:

# Prof. dr Dragan Stojanović Julije Kostov 15680

**Sadržaj**

1. Opis 3

2. Arhitektura 4

2.1. Multiprokotolarni servis za razmenu poruka 5

2.2. Sistem menadžer 5

2.3. Menadžer za korisnike i aplikacije 6

2.4. Storage engine koji radi sa vremenskim serijama podataka 6

2.5. Kompleksno procesiranje događaja 7

3. Upravljanje događajima 8

4. SenML 8

4.1. Mainfluk poruke 8

4.1. Normalizacija 9

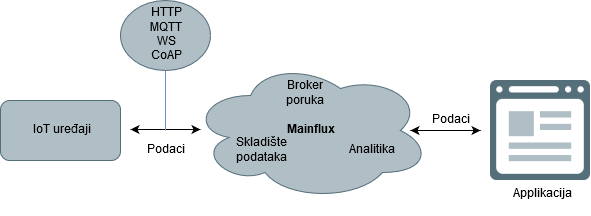
5. Bezbednost 9

6. Literatura 11

1. **Opis**

Mainflux je moderna, skalabilna, visoko osigurana, open source IoT cloud platforma napisana u programskom jeziku Go koja je zasnovana na arhitekturi mikroservisa. Omogućava povezivanje IoT uređaja, korisnika i aplikacija preko raznih mrežnih protokola, kao što su HTTP, MQTT, WebSocket i CoAP. Takođe pruža i distribuiranu bazu podataka za rad sa vremenskim serijama podataka.

Koristi se kao IoT middleware za izgradnju kompleksnih IoT rešenja. Pruža API za povezivanje IoT uređaja i aplikacija kao i razmenu poruka između njih. Omogućava reupotrebljivost tako da je za razvoj IoT sistema potrebno razviti aplikaciju za krajnjeg korisnika kao i IoT uređaje.



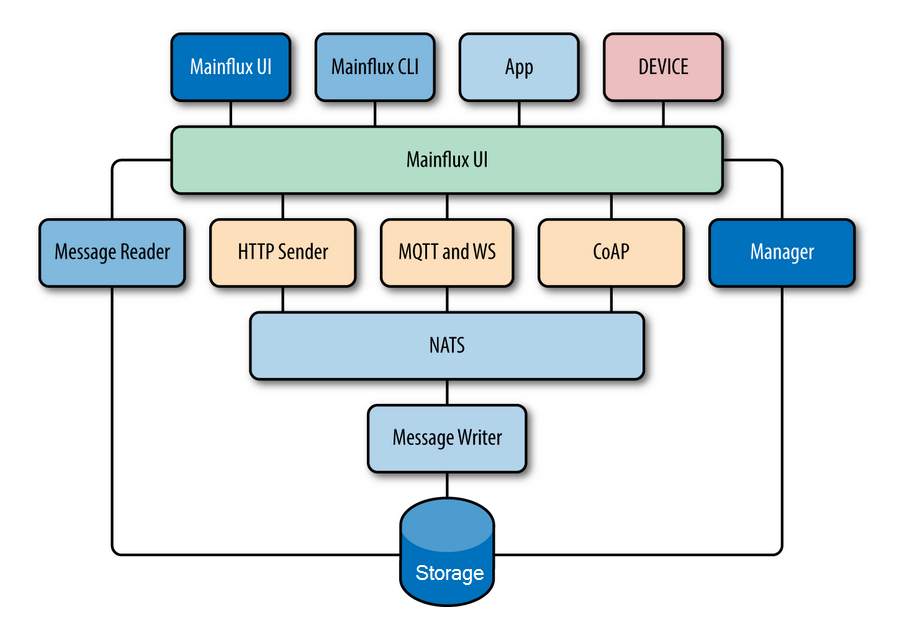
Sl. 1. Lokacija mainflux platforme u sistemu.

Karakteristike:

* Multiprotocol komunikacija
* Omogućava uravljanje uređajima
* Podrška za različite baze podataka (Cassandra, InfluxDB, MongoDB)
* Dizajn arhitekture vođen domenom
* Testabilnost
* Deployment pomoću docker-a

1. **Arhitektura**

Mainflux se koristi kao middleware koji se sastoji od više servera koji pružaju različite funkcionalnosti i servise koji se koriste prilikom razvoja IoT sistema. Mainflux koristi arhitekturu mikroservisa.



Sl. 2. Arhitektura Mainflux-a sa servisima

Lista servisa:

* Servis za razmenu poruka (HTTP Sender, MQTT, WS i CoAP) – prosleđivanje poruka između aplikacija i uređaja
* Sistem menadžer (Manager) – nadgledanje i upravljanje sistemom
* Menadžer aplikacija (App) – upravljanje konekcijama sa aplikacijama
* Menadžer korisnika (Mainflux UI, Mainflux CLI) – upravljanje korisnicima aplikacija
* Storage engine koji radi sa vremenskim serijama podataka (Message Reader, Message Writer i Storage)
* Kompleksno procesiranje događaja (NATS) – koristi vremenske serija podataka za automatsko izvršenje akcija na osnovu definisanih pravila.
  1. **Multiprokotolarni servis za razmenu poruka**

Mainflux se koristi za razmenu poruka između aplikacija i uređaja. I aplikacije i uređaji mogu da generišu i konzumiraju poruke. Na primer, jedan uređaj može biti temperaturni senzor (obezbeđuje merenje temperature) i šalje merenja Mainflux-u, koji ih prenosi aplikaciji koja generiše grafik na osnovu očitavanja. U nekim drugom scenariju, uređaj može biti aktuator - na primer, u slučaju pametne brave sa elektromotorom za zaključavanje / otključavanje vrata. Aplikacija koja komanduje zaključavanjem bi u tom slučaju generisala poruke. Postoje, naravno, scenariji u kojima aplikacija može biti i provajder konteksta i potrošač, ili scenariji u kojima dve ili više aplikacija (ili uređaja) razmjenjuju poruke između njih.

Protokoli koje Mainflux koristi za rezmenu poruka:

* HTTP
* WebSocker
* MQTT
* CoAP
  1. **Sistem menadžer**

Da bi se opisala uloga system menadžera potrebno je predstaviti entitete koje koristi ovaj sistem.

**Korisnici**

Korisnici Mainflux platforme kao što su administrator, programmer itd. moraju da imaju svoj nalog na platformi. Za korisnika se vezuju ostali eniteti sistema (čiiji je vlasnik korisnik) tako da je prvo potrebno kreirati nalog korisniku. Kao vlasnik entiteta korinik može da upravlja svojim uređajima, aplikacijama i kanalima.

**Klijenti**

Klijenti koriste mainflux platform. Postoje dva tipa klijenata: uređaji i aplikacije.

**Uređaji**

Uređaji su obično mikrokontroleri sa ograničenim hardverom. Zbog ograničene memorije oni obično ne podržavaju “heavyweight“ protokole.. Zbog toga se tu koriste MQTT i CoAP - jer su oni “lightweight” protokoli dizajnirani za M2M(machine to machine) komunikaciju. Takođe, na strani sigurnosti, Za sigurnost se koristi DTLS protocol umesto TLS protokola.

Za povezivanje sa ovakvim uređajima Mainflux pruža MQTT I CoAP API.

Za bolje uređaje kao što je RaspberryPi Mainflux pruža standardni HTTP i WebSocket API.

**Aplikacije**

Aplikacije kreirane nezavisno od Mainflux-a od strane nekog programera mogu se povezati na Mainflux platform pomoću svih protokola koje pruža Mainflux ali najčešće se koriste HTTP I WebSocket protokoli. Mainflux omogućava da se pomoću WebSocket-a promene šalju aplikaciji u realnom vremenu.

**Kanali**

Kanali predstavljaju model komunikacionog kanala. Na njemu različiti uređaji ili aplikacije mogu da objavluju ili da se pretplate. Sve što se pošalje kroz kanal se čuva u bazi podataka.

**Menadžer uređaja**

Mainflux u bazi čuva podatke svakog povezanog uređaja koji predstavljaju trenutno stanje tog uređaja. Ove podatke uređaj može da menja, tako da prilikom promene podataka uređaj može da pošalje zahtev za ažuriranje. Imajući u vidu da Mainflux čuva sve te podatke o uređajima, može se koristiti za upravljanje uređajima kao i za dobijanje različitih vrsta informacija o njima kao što su: koliko je uređaja povezano, gde se nalaze, koji je status baterije itd.

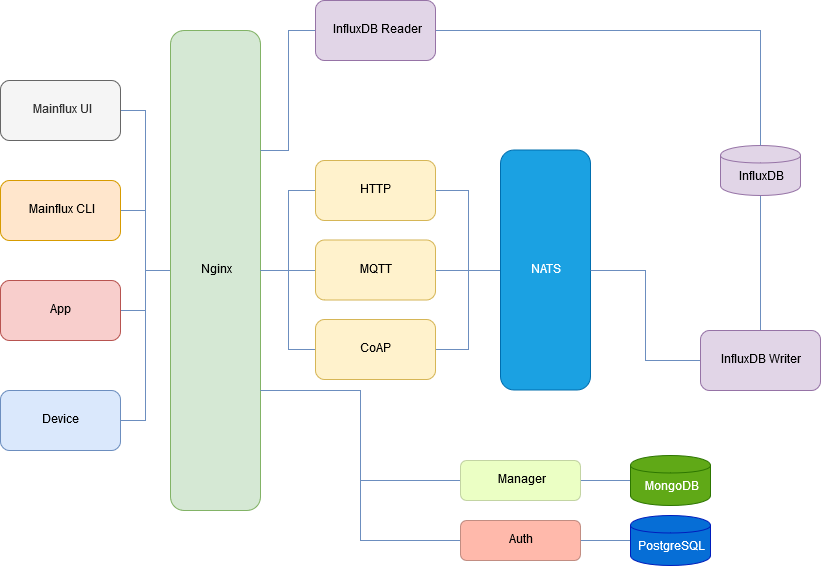
Takođe Mainflux se može koristiti za slanje različitih komandi uređajima kao deo procesa upravljanja uređajem: omogući/onemogući uređaj, ažuriranje softvera, promena privilegija pristupa itd.

* 1. **Menadžer za korisnike i aplikacije**

Mainfluk je višekorisnička platforma za upravljanje aplikacijama. Mainflux programeri mogu kreirati nove višekorisničke aplikacije na vrhu Mainflux-a bez potrebe da sami rukuju svojim krajnjim korisnicima - tj. Mainflux upravlja korisnicima za ove aplikacije. U tom smislu, Mainfluk je sličan Twitter-u ili Facebook-u - kreiranje Mainfluk aplikacije bi trebalo da bude proces sličan kreiranju Twitter aplikacije. Na primer:

* Korisnik kreira nalog programera na Mainflux-u.
* Korisnik kreira novu aplikaciju koristeći Mainfluk javni API.
* Korisnik razvija svoju aplikaciju koja poziva Mainfluk javni API za kreiranje korisnika, slanje poruka, upravljanje itd.
  1. **Storage engine koji radi sa vremenskim serijama podataka**

Mainfluk koristi NoSQL baze za rad sa vremenskim serijama podataka. Većina industrijskih IoT slučajeva su blisko povezani sa telemetrijom. Senzori izvode merenja i zatim ih postavljaju na Cloud. Ove baze podataka su prilagođene za rad sa telemetrijskim podacima i omogućavaju brze kompleksne upite o ovoj vrsti podataka.

****

Sl. 3. Arhitektura Mainflux-a

Mainflux omogućava skalabilnost baza podakata kao i korišćenje različitih baze za različite svrhe. Na slici 3. prikazana je arhitektura Mainflux-a gde se koriste baze MongoDB, PostgreSQL i InfluxDB. Za podatke o korisnicima i uređajima se koristi PostgreSQL, dok se za poruke koje razmenjuju uređaji mogu koristiti Cassandra DB, MongoDB, InfluxDB ili da nema baze ako nije potrebno skladištenje poruka.

* 1. **Kompleksno procesiranje događaja**

Kompleksna obrada događaja (Complex event processing) je metoda analize i obrade tokova informacija o određenim događajima i iz njih se izvodi zaključak. Mainflux koristi Apache Spark za izvršavanje transformacija nad podacima.

1. **Upravljanje događajima**

Mainfluk podsistem za razmenu poruka baziran je na asinhronim događajima, kao što su merenja koja dolaze od senzora, ili komande koje dolaze od aplikacija. Mainfluk koristi NATS broker za poruke. Važno je napomenuti da Mainfluk, koristeći prethodno pomenuti NATS broker za poruke, interno konstruiše takozvani „most protokola“. To znači da ako je klijent, na primer, pretplaćen na MQTT kanal on bi primio poruku objavljenu putem HTTP POST-a na isti kanal. Osim toga što pravi most između različitih protokola za komunikaciju, NATS broker prosljeđuje poruke mirkoservisu Message Writer, koji primljene poruke pretvara u odgovarajući format koji se potom smešta u bazu podataka. Očekuje se da poruke koje dolaze sa senzora budu formatirane pomoću SenML-ovog standarda za modeliranje poruka.

1. **SenML**

Da bi se što više uskladio sa postojećim standardima, Mainfluk koristi SenML-ov standard kao model za poruke koji dolazi od klijenata. SenML je IETF standard koji pruža jednostavan model za podatke sa senzora i kontrolnih uređaja. On definiše minimalnu semantiku za podatke i metapodatke. Više o SenML formatu može se naći na ovde [SenML](https://tools.ietf.org/html/rfc8428). Primer poruke u SenML formatu:

[

{

"bn" : "urn:dev:ow:10e2073a0108006;"

"bt" : 1.276020076001e+09,

"bu":"A",

"bver" : 5,

"n" : "voltage",

"u" : "V",

"v" : 120.1

},

{"n" : "current", "t" : −5, "v" : 1.2},

{"n" : "current", "t" : −4, "v" : 1.3},

{"n" : "current", "t" : −3, "v" : 1.4},

{"n" : "current", "t" : −2, "v" : 1.5},

{"n" : "current", "t" : −1, "v" : 1.6},

{"n" : "current", "v" : 1.7}

]

* 1. **Mainfluk poruke**

Mainfluk koristi sledeći format da enkapsulira SenML poruke koje dolaze sa senzora u svoj interni format za poruke:

type RawMessage struct {

Channel string `json:"channel"`

Publisher string `json:"publisher"`

Protocol string `json:"protocol"`

ContentType string `json:"content\_type"`

Payload []byte `json:"payload"`

}

Prilikom kreiranja poruke ovakvog formata u polju Payload se stavlja SenML sadržaj koji dolazi sa senzora i dodaje metapodatke kao što je identifikator pošiljaoca, protokol korišćen za slanje poruke, itd.

* 1. **Normalizacija**

Message Writer mikroservis dobija poruke od NATS brokera koje su u prethodno definisanom internom format. Zatim on vrši analizu ove poruke, izdvaja korisne podatke iz SenML payload-a i to upisuje u bazu podataka. Poruka koja se skladišti u bazi podataka ima sledeći sadržaj:

type Message struct {

Channel string

Publisher string

Protocol string

Version int `json:"bver,omitempty"`

Name string `json:"n,omitempty"`

Unit string `json:"u,omitempty"`

Value float64 `json:"v,omitempty"`

StringValue string `json:"vs,omitempty"`

BoolValue bool `json:"vb,omitempty"`

DataValue string `json:"vd,omitempty"`

ValueSum float64 `json:"s,omitempty"`

Time float64 `json:"t,omitempty"`

UpdateTime float64 `json:"ut,omitempty"`

Link string `json:"l,omitempty"`

}

1. **Bezbednost**

Mainflux sistem koristi za bezbednost access liste i uloge klijenata. Takođe vrši enkripciju komunikacije koristeći standarde kao što je TLS. Isto tako, za šifrovanje CoAP komunikacije, koja je zasnovana na UDP transportnom protokolu, Mainfluk koristi DTLS. Glavnu ulogu u održavanju bezbednosti sistema su Menadžer servis i reverse-proxy Nginx.

Uloga Menadžera je da vrši autentifikaciju pomoću JSON Web Tokena (JWT), kao i autorizaciju pomoću access listi i uloge korisnika koje se skladište u internoj upravljačkoj bazi podataka.

Nginx ima sledeće uloge u sistemu:

* Reverse-proxy
* TLS enkripcija
* Raspoređivanje opterećenja
* Visoka dostupnost (skalabilnost)

Nginx kao reverse-proxy omogućava da se zaštite sve private rute u okviru sistema. Tako da sav spoljašnji saobraćaj ide preko njega a on ga prosleđuje određenom servisu. Takođe, treba napomenuti da se zbog jednostavnosti sistema TLS / DTLS saobraćaj prekida prilikom ulaska u sistem, pošto se primene Mainfluk-a obavljaju u bezbednoj privatnoj mreži iza Nginx-a.

1. **Literatura**
2. Mainflux documentation - <https://mainflux.readthedocs.io/en/latest/>
3. Mainflux github - <https://github.com/mainflux/mainflux>
4. Architecturing and securing IoT platforms - https://www.itnextsummit.com/wp-content/uploads/2017/11/0940\_Isidorovic.pdf
5. Introduction to Mainflux - <https://www.oreilly.com/library/view/scalable-architecture-for/9781492024132/ch04.html>
6. Mainflux open source IoT platform set up and usage - https://medium.com/mainflux-iot-platform/mainflux-open-source-iot-platform-set-up-and-usage-70bed698791a
7. SenML - https://tools.ietf.org/html/rfc8428