



REPUBLIKA SLOVENIJA
MINISTRSTVO ZA JAVNO UPRAVO



Financira
Evropska unija
NextGenerationEU



Analiza stanja

Naziv projekta	Idejna zasnova in postavitve pilota ekosistema interneta stvari z algoritmičnimi orodji
Status dokumenta	Prva verzija
Datum izdelave dokumenta	11.4.2023
Datum zadnje spremembe	11.05.2023
Verzija dokumenta	1.0
Avtor dokumenta	Marko Bajec, Jernej Cvek, Matjaž Pančur, Gregor Burger, Franc Drobnič, Andrej Kos



Kazalo vsebine

Kazalo vsebine	2
1. Zahteve	4
1.1 Zahteve naročnika	4
1.2 Zahteve končnega uporabnika	6
2. Standardi	8
3. Analiza zahtev	9
3.1 Sestavni deli interneta stvari	9
3.1.1 Prehod na robu	10
3.1.2 Omrežje	10
3.1.3 Platforma IoT	11
3.1.4 Varnost	12
3.2 FIWARE generični omogočevalci kot gradniki platforme IoT	20
3.2.1 Pregled generičnih omogočevalcev FIWARE	20
3.2.2 Ravni avtorizacije FIWARE	21
3.3 Alternativne možnosti gradnikov platforme IoT	22
3.3.1 Prehod API namesto povratnega posredniškega strežnika PEP	23
3.3.2 Alternativne rešitve za upravljanje identitet in dostopov	24
3.3.3 Vizualizacija podatkov	25
4 Predlog načina izvedbe pilotne platforme IoT	28
4.1 Visokonivojski pogled arhitekture	28
4.2 Senzorika	29
4.3 Podprti prehodi IoT na robu	30
4.3.1 Arhitektura OPC UA	30
4.4 Južni sestav platforme IoT	31
4.4.1 Vstopna točka (prehod API)	31
4.4.2 Posrednik sporočil MQTT	31
4.4.3 Omrežni strežnik LoRaWAN	32
4.4.4 Agenti IoT	33
4.5 Osrednji sestav platforme IoT	33
4.5.1 Fiware Context Broker	33
4.6 Severni sestav platforme IoT	34
4.6.1 QuantumLeap, CrateDB, Grafana	34
4.6.2 Thingsboard	34
4.7 Varnost platforme IoT	36
4.7.1 Kong	36
4.7.2 Keycloak	37
5. Namestitev v Kubernetes (K8s)	39
5.1 Uvod v Kubernetes	39
5.2 Možnosti namestitev s pomočjo K8s	40
5.3 Izzivi pri uporabi Kubernetesa	40
5.4 Kubernetes Fiware Foundation referenčna postavitve	41
5.5 Načrt namestitve pilotne platforme	41

1. Zahteve

1.1 Zahteve naročnika

Zahteve naročnika se nahajajo v dokumentu tehničnih specifikacij naročila. Spodaj ležeči tekst podpoglavja vsebuje zahteve naročnika za idejno zasnovo in postavitev pilota ekosistema interneta stvari z algoritmičnimi orodji, kot so bile navedene v Tehničnih specifikacijah naročila.

Splošne zahteve

Z namenom enostavnejše vzpostavitve enotnega evropskega digitalnega trga je Evropska Komisija s (so)financiranjem fundacije FIWARE iz javnih sredstev omogočila razvoj cele vrste odprtokodnih digitalnih gradnikov in standardov. Uporabo teh gradnikov in standardov Evropska Komisija priporoča in spodbuja, saj je smiselna tako iz vidika smotrosti porabe javnih sredstev kot tudi zagotavljanja čim višje ravni interoperabilnosti med sorodnimi sestavi. To je razlog, da se daje uporabi teh digitalnih gradnikov, standardov in specifikacij prednost pred drugimi rešitvami. Sem sodita zlasti:

- Standardizirani informacijski model in aplikacijski programski vmesnik NGSI-LD (ali vsaj NGSI-v2).
- CEF digitalni gradnik Context Broker. Uporaba odprtokodnega digitalnega gradnika je brezplačna, zagotovljeno je tudi dolgoročno vzdrževanje ter podpora pri uvajanju s strani Evropske komisije oz. CEF.

Pri oblikovanju pilotne rešitve se smiselno upoštevajo obvezna navodila za izdelavo informacijskih sistemov Ministrstva za javno upravo, Direktorata za informatiko, ki se nahajajo v dokumentu »Generične Tehnološke Zahteve« na povezavi

<https://nio.gov.si/nio/asset/dokument+genericne+tehnoloske+zahteve+gtz-743>.

Osrednji sestav - Splošne tehnične zahteve za pilota

Osrednji sestav – skladnost s standardi

Z namenom zagotavljanja čim večje splošnosti, standardizacije, interoperabilnosti in odpornosti na izzive, ki jih prinaša prihodnost, se zahteva skladnost s specifikacijami NGSI-LD (ali vsaj NGSI-v2), t.j. standardiziranim informacijskim modelom in aplikacijskim programskim vmesnikom za objavlanje, poizvedovanje in naročanje na informacije o kontekstu, katerega namen je olajšati odprto izmenjavo in souporabo strukturiranih informacij med različnimi zainteresiranimi stranmi.

V okviru pilotnega projekta mora biti na vseh ravneh upravljanje celotnega življenjskega cikla informacij o kontekstu, vključno s posodobitvami, poizvedbami, registracijami in naročninami poenoteno in skladno s specifikacijami NGSI-LD (ali vsaj NGSI-v2).

Osrednji sestav – povezljivost s končnimi napravami interneta stvari – prenosni in podatkovni protokoli in smer podatkovnega toka

Osrednji sestav mora omogočati povezovanje s končnimi napravami interneta stvari po naslednjih prenosnih in podatkovnih protokolih:

- HTTP – Ultralight,
- HTTP – JSON,
- MQTT – Ultralight,
- MQTT – JSON,
- LoRaWAN - Cayenne LPP,
- LoRaWAN – CBOR,
- LoRaWAN – možnost določitve dekoderske funkcije za lastniški podatkovni format.

Osrednji sestav mora omogočati razširitve z dodatnimi prenosnimi in podatkovnimi protokoli ter povezovanje končnih naprav interneta stvari (to so viri podatkov (npr. senzorji), ponori podatkov (npr. aktuatorji) in naprav, ki so tako viri kot tudi ponori podatkov (npr. v zgradbah nameščeni koncentratorji)). Zagotavljati mora dvosmeren podatkovni tok.

Osrednji sestav – varnostne zahteve

Osrednji sestav mora biti navzven zavarovan z uporabo mehanizmov avtentikacije in granularne avtorizacije ter z uporabo kriptografskih postopkov pri prenosu podatkov navzven, kar omogoča zavarovanje na vseh ravneh. Tudi komunikacija med osrednjim sestavom in končnimi napravami interneta stvari mora biti, razen v izjemnih primerih, ko je to tehnično neizvedljivo, zavarovana z uporabo mehanizmov avtentikacije in avtorizacije ter z uporabo kriptografskih postopkov pri prenosu podatkov.

Osrednji sestav – skalabilnost

Sestav mora biti zasnovan na način, ki z naraščajočo obremenitvijo sestava omogoča strojno in programsko skalabilnost.

Osrednji sestav – uporabniki

Sestav mora omogočati večorganizacijsko, večuporabniško in večnajemniško delovanje.

Osrednji sestav – obdelava osebnih podatkov

Sestav ne obdeluje osebnih podatkov.

Osrednji sestav – pravni vidiki

Potreben je pregled in potrditev ustreznosti licenčnih pogojev vseh posameznih gradnikov sestava za uporabo v okolju državne uprave.

Osrednji sestav – finančni vidiki

Potrebna je analiza finančnih učinkov sestava za uporabo v okolju državne uprave.

Osrednji sestav - Specifične funkcionalne zahteve za (energetski) nadzor in vodenje stavb za prvega uporabnika DSP

Funkcionalne zahteve v okviru pilota za uporabnika DSP

Pilotni sestav mora omogočati spremljanje porabe energije v stavbah, tako agregirane kot tudi po posameznih porabnikih, zaradi zahtevanega energetskega knjigovodstva in spremljanja učinkovitosti rabe energije. Zagotovljeno mora biti:

- zbiranje podatkov o meritvah porabe električne energije,
- zbiranje podatkov o meritvah porabe toplotne energije,
- zbiranje podatkov o meritvah porabe plina,
- zbiranje podatkov o meritvah porabe vode.

Funkcionalne zahteve izven okvira pilotnega projekta

Pilotni projekt mora biti zasnovan tako, da bodo v prihodnosti mogoče njegove nadgradnje in razširitve. V nadaljevanju in izven okvira pilotnega projekta se predvideva vzpostavitev energetskega vodenja stavb z namenom energetske-ekonomske optimizacije procesov, ki so v stavbah povezani s porabo energije. Pilotni projekt mora biti zasnovan na način, da bo za potrebe energetskega vodenja zgradb omogočal dvosmerni podatkovni prenos med osrednjim sestavom in končnimi napravami interneta stvari v zgradbi (senzorji, števcji, ventili, aktuatorji itd.). Osrednji sestav mora omogočati povezavo z ločenim programskim gradnikom, ki bo razvit izven okvira projekta in bo udeležan krmilno-regulacijske funkcionalnosti, ki so potrebne za energetske vodenje zgradb.

Predvideni uporabniki pilota:

- MJU / Direktorat za informatiko.
- MJU / Direktorat za stvarno premoženje.

1.2 Zahteve končnega uporabnika

Zajem senzorskih podatkov in upravljanje aktuatorskih naprav

Direktorat za stvarno premoženje želi spoznati objekte z energetskega vidika. Zato je želja z IoT senzorji opremiti zgradbe in zbirati podatke o energetske porabi in energetske učinkovitosti prostorov. Zbrani podatki bodo služili za bodoče analize, in uporabljeni za boljše načrtovanje energetske potreb ter varčno porabo energije v prostih pod upravljanjem Direktorata za stvarno premoženje.

Vgrajeni števcji in naprave uporabljajo MODBUS protokol, podatki se zajamejo na 15 minutnih intervalih. Nekateri števcji, npr. števcji električne energije omogočajo shranjevanje podatkov tudi interno.

V stavbah so trenutno vgrajeni sledeči senzorji:

- števci električen energije,
- kalorimetri,
- temperaturni senzorji,
- dodatno še vremenska postaja (zunanja temperatura).

Aktuatorji (ventili in električna stikala) še niso vgrajeni na testnih objektih. Želja je v pilotno postavitev vključiti tudi vzorčni aktuator.

Pri ventilih je potrebno beležiti sledeča stanja:

- odprto,
- zaprto,
- pozicija ventila.

Pri stikalih je zahteva beleženja stanj:

- vklopljeno stanje,
- izklopljeno stanje,
- povratna informacija o realni izvedbi preklopa.

Obdelava in hranjenje podatkov

V prvi fazi izvedbe pilotnega projekta so pričakovanja prvega uporabnika prenos podatkov v podatkovno bazo, v kateri se podatki hranijo in se jih ne briše. Fokus je na pridobitvi podatkov v Context Broker, njihova vizualizacija in nadaljna obdelava sta drugem planu pilotne izvedbe.

Vizualizacija

Prvi naročnik ne pričakuje potrebe po preračunu veličin meritev. Podatki v bazi (Context Brokerju) morajo omogočati bodoče vizualizacije in podpirati možnost pretakanja podatkov v podatkovno skladišče ministrstva.

Omejevanje dostopa uporabnikov

Predvideva delitev pravic za različne uporabnike, predlagan je bil sledeči ključ

User (Uporabnik): lahko spremlja vse podatke na vseh stavbah. K uporabnikom dodamo še »zaposlenega«, ki si lahko pogleda podatke svojega delovnega okolja.

Admin: Dodeljuje pravice, vidi izbrane stavbe, vpisuje nove uporabnike, ima dostop vseh podatkov in nastavitev aktuatorjev na stavbi

Super Admin: Ima pravico dodeljevanja upravljanja nastavitev aktuatorjev na vseh stavbah

SuperSuper Admin: Upravlja še druge podatke (se določi v prihodnosti)

2. Standardi

Preučili smo standarde, ki se nam zdijo uporabni pri izgradnji platforme IoT z uporabo FIWARE komponent ter standarde, ki so pomembni za realizacijo prvega uporabniškega primera (področje stavb). V nadaljevanju so ti standardi naštet in na kratko opisani.

Gradniki, ki jih za podporo izmenjavi podatkov ponuja združenje FIWARE, so že v osnovi zgrajeni na podlagi standardov, tako normiranih, kot tudi *de facto*. V večini primerov gre tudi za že v praksi uveljavljene standarde in dobre prakse.

- Internetni protokoli IP, TCP, UDP
- Protokol HTTP in v nezaščitenih omrežjih njegova različica HTTPS
- Arhitektura REST in REST API
- Formati HTML, XML, JSON, JSON-LD
- Varnostne tehnologije PKI, JWT
- Tehnologije s področja Interneta stvari MQTT, LoRaWAN, CoAP

Združenje FIWARE kot osnovni standard za izmenjavo podatkov predlaga NGSI (Next Generation Service Interfaces), ki specificira podatkovni model in API. Je plod razvoja nekaj zadnjih desetletij, najprej v združenju mobilnih operaterjev OMA, kasneje pa v okviru partnerstva Future Internet Public-Private Partnership in s podporo Evropske komisije, ki je kasneje ustanovilo skupnost FIWARE. Formalno je NGSI standardiziral inštitut ETSI (European Telecommunications Standardization Institute) in v zadnji različici dodal še funkcionalnost grafov - Linked Data, zato se standard zdaj imenuje NGSI-LD¹.

Za učinkovito strojno izmenjavo podatkov morajo biti ti oblikovani na standarden način, čemur služijo podatkovni modeli. Znotraj FIWARE je nastala množica podatkovnih modelov, združljivih s standardom NGSI in NGSI-LD. Objavljeni so prosto na <https://smartdatamodels.org/> in v repozitoriju <https://github.com/smart-data-models>. Večina modelov izhaja iz drugih pobud, npr. <https://schema.org/>, SAREF ipd. Spodbujajo tudi izgradnjo novih modelov, če obstoječi ne pokrijejo potreb, in predstavljajo za to potrebno metodologijo. Podatkovni modeli so skupaj s konkretnimi podatki podlaga za tvorbo digitalnih dvojčkov (Digital Twin), ki v digitalni obliki odslikavajo predmete in pojme iz resničnega življenja.

Zaradi skladnosti z ostalimi napor standardizacije v državi predlagamo uporabo podatkovnih modelov, ki so zbrani in usklajeni v okviru GZS, in so tudi usklajeni s standardi, ki jih podpira združenje FIWARE: <https://smartsociety.gzs.si/standardi/harmonizacija>

Ontologija Smart Appliances REference (SAREF) <https://w3id.org/saref> je osnovni model področja naprav. Popisuje osnovne lastnosti naprav, njihove funkcije, stanja in storitve.

ETSI je v okviru TC SmartM2M standardiziral več razširitev osnovne ontologije SAREF. Omenjamo nekatere, ki so povezane s primerom uporabe prvega uporabnika:

- Pametna mesta: SAREF4CITY (ETSI TS 103 410-4)

¹ https://www.etsi.org/deliver/etsi_gr/CIM/001_099/008/01.01.01_60/gr_CIM008v010101p.pdf

- Energija: SAREF4ENER (TS 103 410-1)
- Zgradbe: SAREF4BLDG (TS 103 410-3)
- Voda: SAREF4WATR (TS 103 410-10)

V okviru delovne skupine Context Information Management (CIM) je tudi podal specifikacijo uporabe teh standardov v okolju NGSI-LD CIM 021².

V okviru skupnosti FIWARE nastaja katalog podatkovnih modelov, skladnih s standardom NGSI-LD: <https://github.com/smart-data-models/data-models>

Projekt SinchroniCity je na podlagi specifikacije OASC Minimal Interoperability Mechanisms (MIMs) razširil podatkovne modele skupnosti FIWARE z nekaterimi posebnostmi zaradi posebnih zahtev projekta: <https://gitlab.com/synchronicity-iot/synchronicity-data-models>

Na področju delovanja prvega uporabnika pilotne platforme se je uveljavila informacijska podpora Building Information Modeling (BIM - podatkovno modeliranje stavb). Uveljavljen je standard ISO 19650. Zajema široko področje življenjskega cikla stavb, za naš namen je zanimivo podpodročje učinkovite rabe stavb (angl. Building Performance).

Nekateri navedeni standardi se tudi prekrivajo ali določajo posamezne specifikacije. V postopku izdelave pilota bo treba podrobno pregledati potrebe in na podlagi tega pregleda izbrati podatkovne modele, ki bodo najbolj ustrezali potrebam. Možno je tudi, da bo treba kakšen model še dopolniti, ker so ti standardi praviloma v stalnem razvoju, skladno s potrebami uporabnikov.

3. Analiza zahtev

Na začetku poglavja se za lažje razumevanje seznanimo s sestavnimi deli in varnostnim vidikom interneta stvari. Nato v skladu s splošnimi zahtevami naročnika začnemo našo analizo s pregledom generičnih omgočevalcev FIWARE, ki se nam kot gradniki zdijo uporabni za razvoj platforme IoT. Za tem navedemo alternativne možnosti za določene gradnike. Na željo naročnika smo dali poudarek na varnosti.

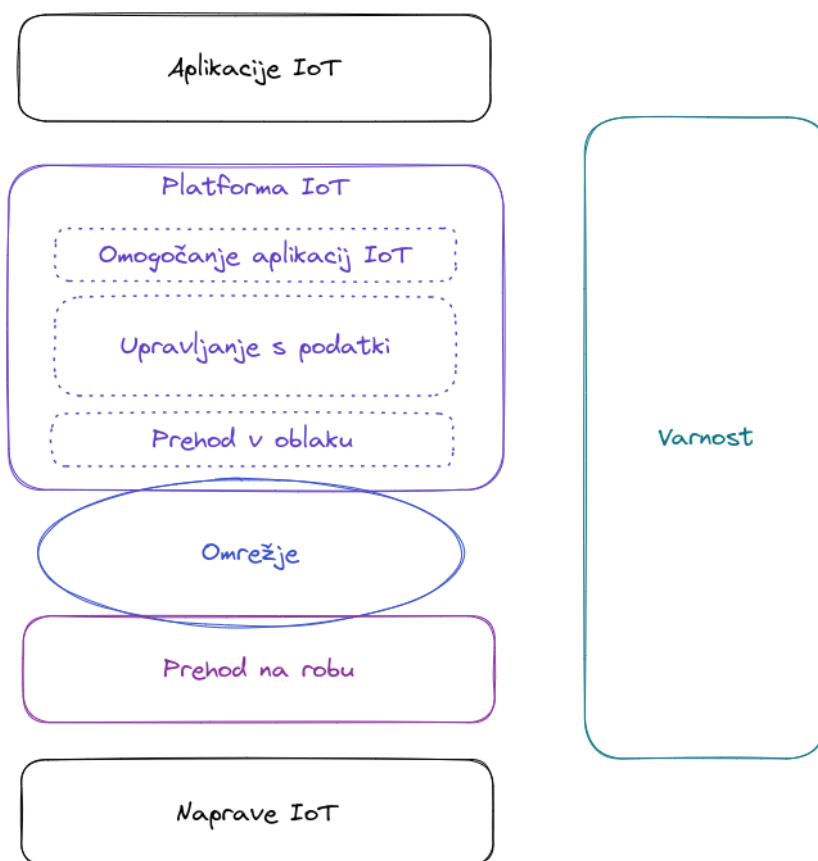
3.1 Sestavni deli interneta stvari

V splošnem so **rešitve interneta stvari** (angl. *internet of things*, krajše *IoT*) zgrajene iz:

- **naprav IoT** (senzorji in aktuatorji),
- **prehoda na robu** (angl. *field / edge gateway*),
- **omrežja**, po katerem se prenašajo podatki,
- **platforme IoT**,
- **aplikacij IoT**.

Pri tem je v celotnem sestavu pomembno prisoten vidik **varnosti**.

² https://www.etsi.org/deliver/etsi_gr/CIM/001_099/021/01.01.01_60/gr_CIM021v010101p.pdf



3.1.1 Prehod na robu

Glavne naloge, ki jih opravlja prehod na robu:

- agregiranje naprav IoT,
- transformacija med različnimi omrežnimi protokoli,
- zagotavljanje povezljivosti.

3.1.2 Omrežje

O **omrežju** (angl. *network*) lahko govorimo, ko si vsaj dve napravi med seboj izmenjujeta podatke. Poznamo več tipov omrežij, ki se razlikujejo po **dosegu** in **pasovni širini** (npr. LAN, WAN). Za prenos podatkov po omrežju se uporabljajo **različni protokoli**, ki definirajo formalen opis pravil za izmenjavo sporočil med različnimi entitetami v omrežju. Lastnosti protokolov so prilagojene glede na domet in pasovno širino, ki sta zahtevana ob njihovi uporabi.

Protokole lahko razporedimo glede na sloje:

sloj	primeri protokolov
fizični sloj (angl. <i>physical layer</i>)	Ethernet, Wi-Fi, Bluetooth, LTE/4G, 5G, NFC, ZigBee, PLC, RFID, SigFox, LoRa, NB-IoT

sloj podatkovne povezave (angl. <i>data link layer</i>)	IEEE 802.15.4, LPWAN
omrežni sloj (angl. <i>network layer</i>)	IP, LoRaWan, 6LoWPAN
prenosni sloj (angl. <i>transport layer</i>)	TCP, UDP
aplikacijski sloj (angl. <i>application layer</i>)	HTTP, WS, MQTT, CoAP, AMQP

V našem primeru se bomo osredotočili predvsem na omrežno komunikacijo med prehodom na robu in platformo IoT.

3.1.3 Platforma IoT

Platformo IoT lahko v grobem razdelimo na:

- **južni del**, t.i. prehod v oblaku (angl. *cloud gateway*),
- **osrednji del**, ki skrbi za upravljanje s podatki,
- **severni del**, ki je zadolžen za izpostavljanje funkcionalnosti platforme IoT aplikacijam IoT.

Glavni nalogi prehoda v oblaku sta (1) sprejemanje in ustrezno posredovanje sporočil in (2) omogočanje upravljanja naprav IoT s pomočjo ukazov. Ponavadi je v vlogi vstopne točke v platformo IoT. Ob tem (podobno kot prehod na robu) nudi transformacijo med različnimi protokoli (predvsem na aplikacijskem nivoju). Poleg tega običajno nudi preoblikovanje oblike sporočil in obogatitve (angl. *enrichment*) sporočil zahtevkov. Lahko rečemo, da je kot neke vrste most (angl. *bridge*) med različnimi protokoli in formati sporočil.

Naprave IoT ustvarijo ogromno količino podatkov. Ravnanje s podatki se lahko v splošnem razdeli na več (zaporednih) faz: (1) generiranje podatkov, (2) zbiranje podatkov, (3) preverjanje veljavnosti podatkov, (4) shranjevanje podatkov, (5) obdelava podatkov, (6) analiza podatkov, (7) izbris podatkov. Pri tem ne smemo pozabiti na varno uporabo in zasebnost podatkov. Prva in druga faza običajno potekata na napravah IoT, medtem ko so naslednje faze v domeni platforme. Ob tem moramo poudariti, da lahko vse zgoraj navedene faze potekajo tudi na dovolj zmogljivih napravah IoT oz. prehodih IoT na robu.

Glavni namen platforme IoT je, da aplikacijam IoT poenostavi zapletenost pridobivanja podatkov z naprav IoT, ki je posledica njihove raznolikosti. Pri tem lahko s pomočjo različnih storitev (analitika, vizualizacija, procesiranje dogodkov (angl. *Complex Event Processing*), odkrivanje znanj iz podatkov) pomaga ustvarjati poslovno vrednost. Storitve platforme so na voljo preko API-jev, običajno so podprte tudi številne integracije z drugimi zunanji sistemi. Poleg tega so na voljo različna orodja za spremljanje stanja platforme in naprav.

3.1.4 Varnost

Preverjanje pristnosti, upravljanje identitet in dostopov

Preverjanje pristnosti ali avtentikacija (angl. *authentication*) je v računalništvu postopek, s katerim se strežnik prepriča, da je uporabnik zares tisti uporabnik, za kogar se predstavlja, da je. Zgled preverjanja pristnosti je vpis uporabniškega imena in gesla pri npr. vpisu v sistem.

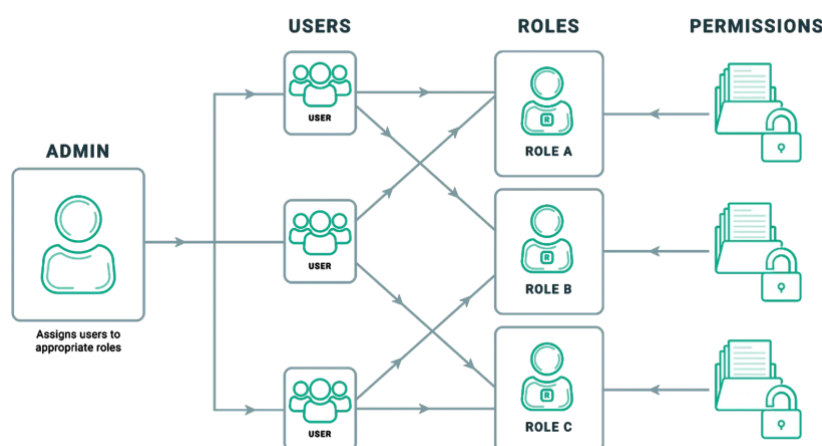
Upravljanje identitet in dostopov (angl. *identity and access management*, krajše **IAM**) je rešitev, ki zajema upravljanje uporabniških poverilnic (angl. *user credentials*), preverjanja pristnosti uporabnikov in nadzora dostopov (angl. *access control*) glede na definirane uporabniške pravice. Njen glavni namen je varno hranjenje podatkov o uporabnikih ter omogočanje "pravih uporabnikom dostop do pravih virov ob pravem času iz pravih razlogov".

Modeli za nadzor dostopa

Poznamo več modelov za nadzor dostopa, spodaj sta na kratko predstavljena dva izmed tradicionalnih modelov.

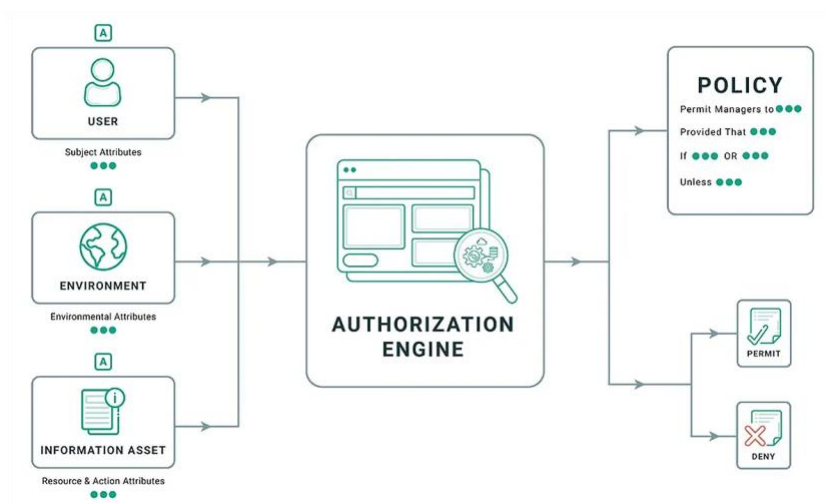
RBAC

Nadzor dostopa na podlagi uporabniške vloge (angl. *Role-Based Access Control*, krajše RBAC) dodeli uporabniku dostop do zaščitenega vira na podlagi njegove uporabniške vloge, na katero je vezan nabor pravic.



ABAC

Nadzor dostopa na podlagi atributov (angl. *Attribute-Based Access Control*, krajše ABAC), ovrednoti nabor pravili in politik za upravljanje pravic dostopa glede na specifične attribute, kot so informacije o okolju, sistemu, objektu, in/ali uporabniku. Uporablja enostavno logiko za odobritev ali zavrnitev dostopa, ki temelji na vrednotenju atributov in relacij med njimi.



RBAC je enostavnejši za uporabo, ABAC pa ponuja možnost bolj natančni nadzor dostopa.

Nadzor dostopa na podlagi politik

Poznamo več načinov nadzora dostopa, med drugim **nadzor dostopa na podlagi politik** (angl. *Policy-based Access Control*). Rešitve, zgrajene po tem načinu, so običajno sestavljene iz sledečih gradnikov.

Policy Enforcement Point (PEP)

PEP prestreže uporabnikov zahtevke za dostop do določenega vira ter ga odobri ali zavrne. PEP ne sprejema odločitev; jih zgolj uveljavlja. Po potrebi lahko prilagodi vsebino zahtevka ali odgovora na zahtevek.

Policy Decision Point (PDP)

PDP na podlagi definiranih politik in dodatnih podatkov sprejme odločitev, ali se zahtevek odobri ali zavrne.

Policy Administration Point (PAP)

PAP je zadolžen za upravljanje politik, ki jih uporablja PDP.

Policy Information Point (PIP)

PIP je kateri koli vir podatkov (notranji ali zunanji), ki vsebuje atribute, pomembne za sprejemanje odločitev v skladu z definiranimi politikami.

Policy Retrieval Point (PRP)

PRP je glavni vir politik - shranjuje politike, ki jih uporablja PDP, upravlja pa ga PAP.

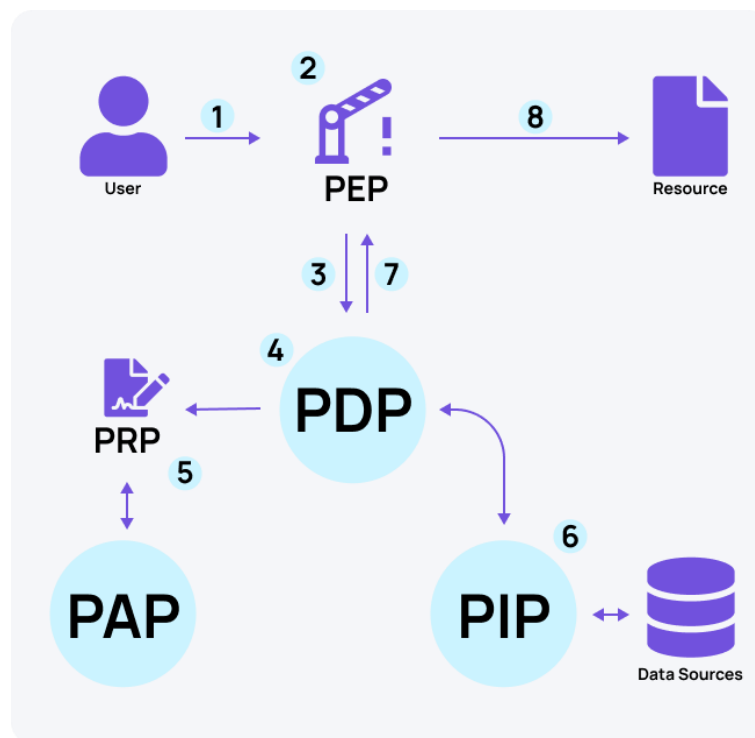
Opisana arhitektura je definirana v sklopu standarda XACML (eXtensible Access Control Markup Language).

Postopek avtorizacije XACML

Spodnja slika prikazuje tipični potek avtorizacije po načinu nadzora dostopa na podlagi politik, implementiranega po standardu XACML:

1. Uporabnik želi izvesti zahtevek do določenega vira.

2. Zahtevek prestreže PEP, ki ga preoblikuje v XACML zahtevek za avtorizacijo.
3. PEP oblikovani zahtevek za avtorizacijo posreduje PDP-ju v odločanje.
4. PDP sprejme odločitev o avtorizaciji na podlagi vnaprej definiranih politik v formatu XACML.
5. Politike so shranjene v PRP, upravlja jih PAP.
6. Po potrebi PDP opravi poizvedbo v relevantnih PIP za morebitne dodatne informacije, ki pomagajo pri odločanju.
7. PDP na podlagi definiranih politik in drugih zbranih podatkov sprejme odločitev (odobri / zavrne zahtevek) ter jo v obliki odgovora na zahtevek za avtorizacijo vrne PEP.
8. PEP uporabniku odobri ali zavrne dostop do vira na podlagi odločitve PDP.



Primer XACML pravil, ki določajo politiko dostopa:

```

<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
    http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd" ReturnPolicyIdList="false">
  <Attributes Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
    <Attribute IncludeInResult="false" AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id">
      <AttributeValue DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name">
        nick.riviera@sfhospital.org </AttributeValue>
      </Attribute>
    <Attribute IncludeInResult="false" AttributeId="net:drozdowicz:sxacml:subject:subject-role-id">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI">
        http://hl7.org/ontology/RoleOntology.owl#PharmacistFunctionalRole </AttributeValue>
      </Attribute>
    </Attributes>
  <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
    <Attribute IncludeInResult="false" AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI">
        file://med/bsimpson/20151015/bloodPressure.json </AttributeValue>
      </Attribute>
    <Attribute IncludeInResult="false" AttributeId="sxacml:resource:resource-class-id">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI">
        http://drozdowicz.net/sxacml/eHealthSample:BloodPressure </AttributeValue>
      </Attribute>
    </Attributes>
  <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
    <Attribute IncludeInResult="false" AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">read</AttributeValue>
      </Attribute>
    </Attributes>
  </Request>

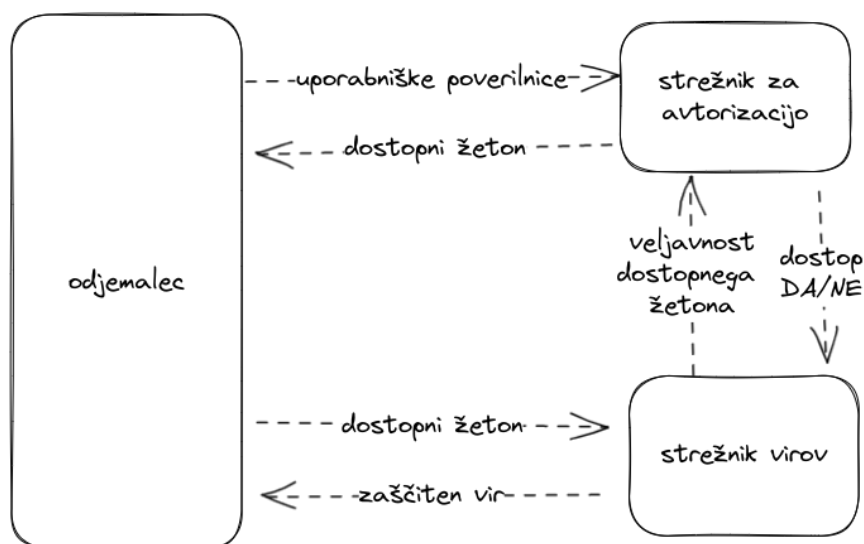
```

Obstaja več implementacij standarda OASIS XACML v3.0; v FIWARE ekosistremu je to rešitev **AuthzForce** (<https://github.com/authzforce/core>).

Za t.i. "cloud-native" okolja se je pojavila podobna rešitev za nadzor dostopa na podlagi politik: **Open Policy Agent** (<https://www.openpolicyagent.org/>).

Nadzor dostopa na podlagi žetona

Za avtentikacijo in avtorizacijo se pogosto uporablja tudi **nadzor dostopa na podlagi žetona** (angl. *token-based authentication*). Odjemalec (npr. uporabnik, naprava IoT,...) v zameno za uporabniške poverilnice od strežnika za avtorizacijo (angl. *authorization server*) prejme dostopni žeton (angl. *access token*), ki mu omogoča dostop do zaščitene vira na strežniku virov (angl. *resource server*).



OAuth 2.0 je odprt standard za avtorizacijo, ki opisuje ogrodje za avtorizacijo (angl. *authorization framework*), znotraj katerega so definirani protokoli in potek avtorizacije.

OpenID Connect (krajše **OIDC**) je odprt standard za avtentikacijo v obliki dodatnega sloja nad OAuth 2.0 protokolom. Namenjen je avtentikaciji uporabnikov, omogoča tudi t.i. *single sign-on*.

Za nadzor dostopa na podlagi dostopnega žetona za naprave IoT se pogosto uporablja t.i. "OAuth 2.0 Device Authorization Grant" (opisan v RFC 8628).

Omrežna varnost (zaščita komunikacije)

Varnost prenosnega sloja

Varnost prenosnega sloja (angl. *Transport Layer Security*, krajše *TLS*) je široko sprejet varnostni kriptografski protokol, zasnovan za zagotavljanje zasebnosti in varnosti podatkov med komunikacijo po medmrežju.

Protokol TLS v glavnem skrbi za:

- **šifriranje** (angl. *Encryption*), ki skriva prenesene podatke pred tretjimi osebami.
- **preverjanje pristnosti** (angl. *Authentication*), ki zagotavlja, da sta stranki, ki si izmenjujeta informacije, tisti, za kateri se izkazuje,
- **integriteto** (angl. *Integrity*), ki preverja, da podatki niso bili ponarejeni ali prirejeni.

Več o delovanju protokola TLS si bralec lahko prebere tukaj: <https://hpbn.co/transport-layer-security-tls/>.

Kriptografski sistemi (angl. *cryptographic systems*) s pomočjo šifirnih algoritmov v kombinaciji s kriptografskimi ključi pretvarjajo golo vsebino sporočila (angl. *plaintext message*) v šifrirano obliko. **Šifrirni algoritem** (angl. *cipher*) je serija dobro definiranih računskih korakov, ki se uporablja za šifriranje sporočila. Onemogočil (ali pa vsaj zelo otežil) naj bi obratno pridobivanje originalne vsebine iz šifrirane vsebine sporočila. Zaradi splošne dostopnosti informacij o delovanju šifirnih algoritmov, današnji kriptografski sistemi poleg njih uporabljajo tudi kriptografske ključe. **Kriptografski ključ** je niz bitov, ki

določa izhod šifrnega algoritma. Za ohranjanje varnosti torej ni več potrebna tajnost šifrnega algoritma, saj se lahko šifrirano sporočilo dešifrira le, če dešifrirni kriptografski ključ ustreza šifrnemu kriptografskemu ključu.

Kriptografija javnega ključa (angl. *public-key cryptography*), poznana tudi kot asimetrična kriptografija (angl. *asymmetric cryptography*), je kriptografski sistem, ki za enkripcijo uporablja par kriptografskih ključev: javni ključ, ki je lahko javno znan, in zasebni ključ, ki je znan samo lastniku. Pošiljatelj s pomočjo javnega ključa, ki služi kot parameter šifrnemu algoritmu, zašifrira vsebino sporočila. Javni ključ se običajno uporablja kot parameter v šifrnem algoritmu, medtem ko se zasebni ključ uporablja za dešifriranje.

Digitalno potrdilo javnega ključa (angl. *public key certificate*) je digitalni dokument, ki potrjuje povezavo med javnim ključem in neko osebo, institucijo ali strežnikom.

HTTPS (kratica za *Hypertext Transfer Protocol Secure*) je razširitev protokola HTTP. V protokolu HTTPS je komunikacijski protokol šifriran s protokolom TLS, zato je njegovo ime tudi "HTTP over TLS". Varnost protokola HTTPS torej zagotavlja protokol TLS, ki običajno uporablja dolgoročne javne in zasebne ključe za ustvarjanje kratkoročnega ključa seje, ki se nato uporablja za šifriranje podatkovnega toka med odjemalcem in strežnikom. Za preverjanje pristnosti strežnika (in včasih tudi odjemalca) se uporabljajo digitalna potrdila. Posledično so organi za izdajo digitalnih potrdil (angl. *certificate authorities*) in digitalnega potrdila javnega ključa potrebni za preverjanje razmerja med digitalnim potrdilom in njegovim lastnikom ter za generiranje, podpisovanje in upravljanje veljavnosti digitalnih potrdil.

Podobno velja tudi za protokol **MQTT**. Manjša slabost uporabe MQTT je v tem, da varnost s seboj prinese dodatne obremenitve v smislu porabe CPE in dodatno potrebne komunikacije. Čeprav je dodatna poraba CPE zanemarljiva na infrastrukturi, kjer so običajno nameščeni posredniki sporočil (angl. *message brokers*), lahko postane problematična za naprave z omejenimi viri (angl. *constrained devices*), ki niso zasnovana za računsko intenzivna opravila. V primeru kratkoživih povezav, je vpliv TLS Handshake-a relativno velik, zato se priporoča uporaba dolgoživih povezav, v kolikor je to možno.

Varnost v omrežjih Modbus

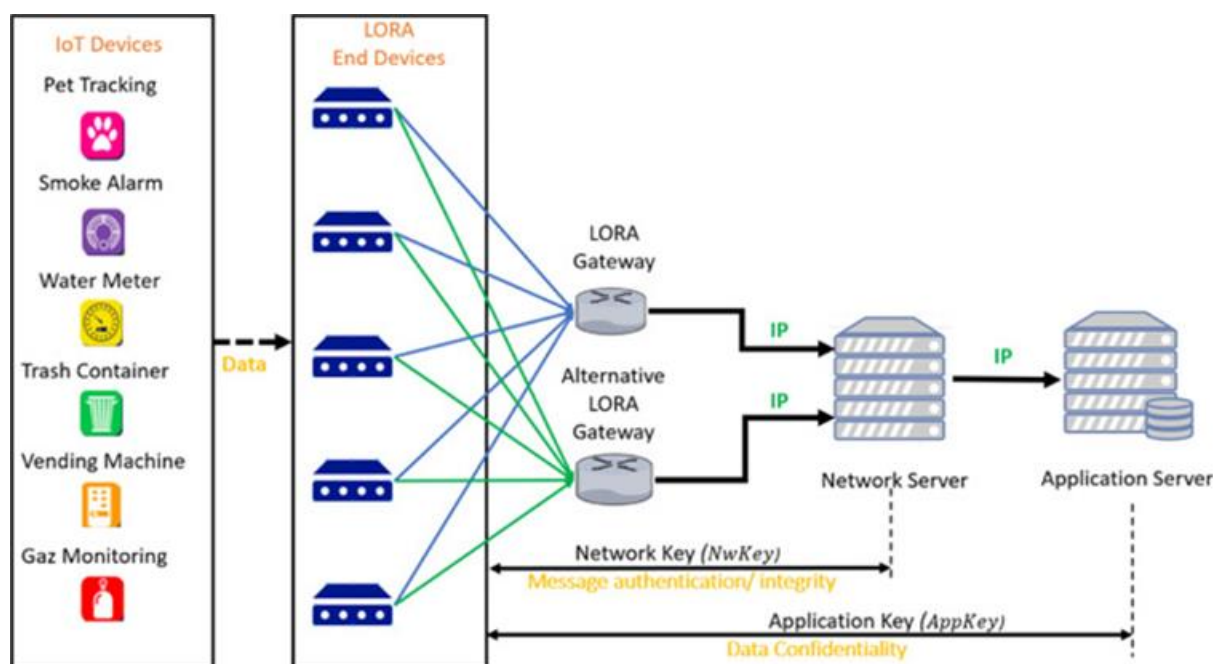
MODBUS, ki je bil prvotno zasnovan za komunikacijo med različnimi napravami v industrijskih sistemih, ne vključuje nobenih varnostnih protokolov. V primeru uporabe MODBUS prek TCP/IP omrežij (MODBUS/TCP) pa je treba uporabiti standardne varnostne protokole in mehanizme za zagotavljanje varnosti v komunikaciji na protokolu IP (TLS, DTLS, SSH, IPsec, idr.).

Varnost v LoraWan omrežjih

Omrežja LoRaWAN podpirajo sodobne tehnologije zaščite podatkov:

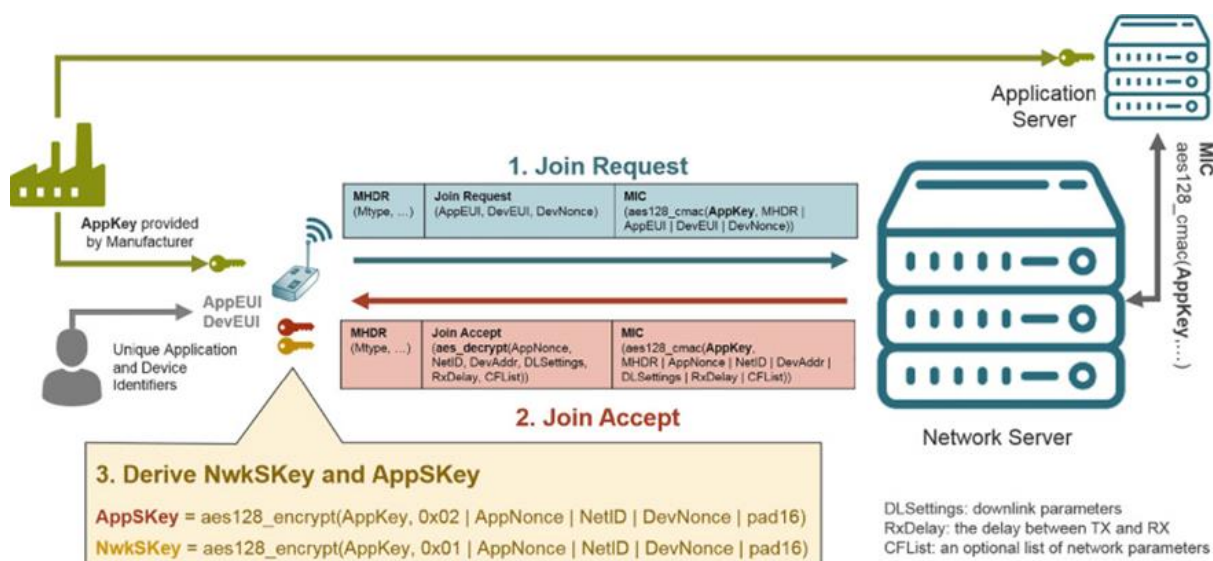
- AES-128
- HMAC
- EUI-64 za identifikacijo (naprav ipd.)
- Join Request/Join Accept protokol za zavarovanje pridruževanja naprav
- Frame Counter protokol za preprečevanje podvojitve prenosa podatkov

Za zaščito prenosa podatkov je v omrežjih LoRaWAN možnih več nivojev.



Avtentikacija in zaščita pred ponarejanjem je izvedena z uporabo omrežnega ključa (Network Key), ki ščiti podatke med prenosom od LoRa končnih naprav (npr. senzorjev) do omrežnega strežnika, in to po brezžičnem prenosu LoRa in kasneje po internetnem protokolu (IP). Zaupnost vsebine pa omogoča uporaba aplikacijskega ključa (Application Key), ki ščiti podatke še naprej, do aplikacijskega strežnika, in s tem onemogoča omrežnemu strežniku vpogled v podatke. Oba ključa sta po standardu AES-128.³ Varnost prenosa po omrežjih IP pa je lahko zagotovljena s standardnimi rešitvami za IP, npr. TLS, IPsec, požarne pregrade.

Protokol Join Request/Join Accept:

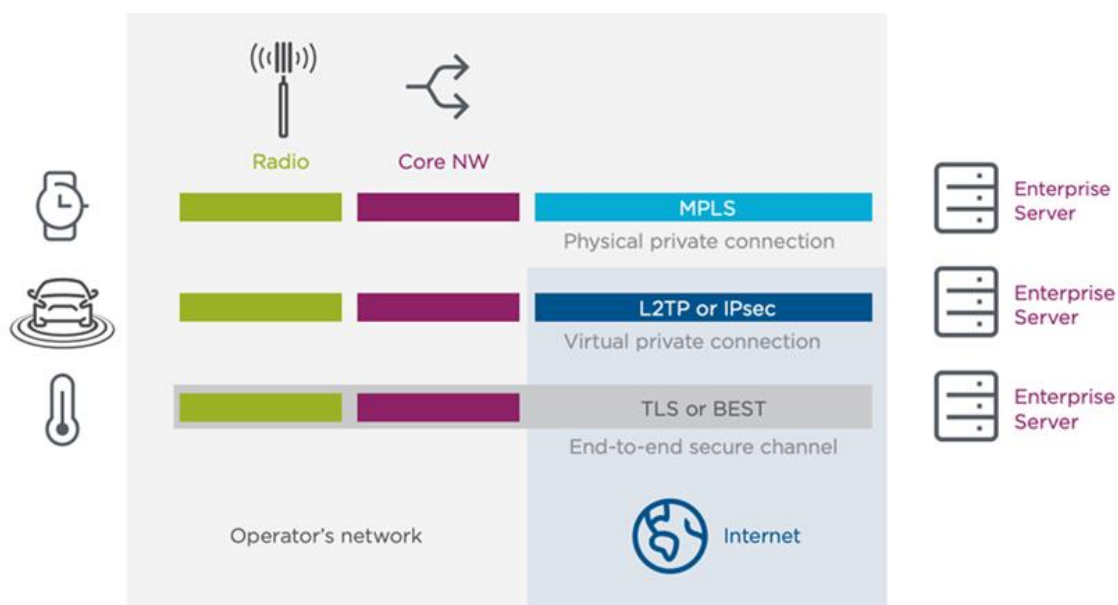
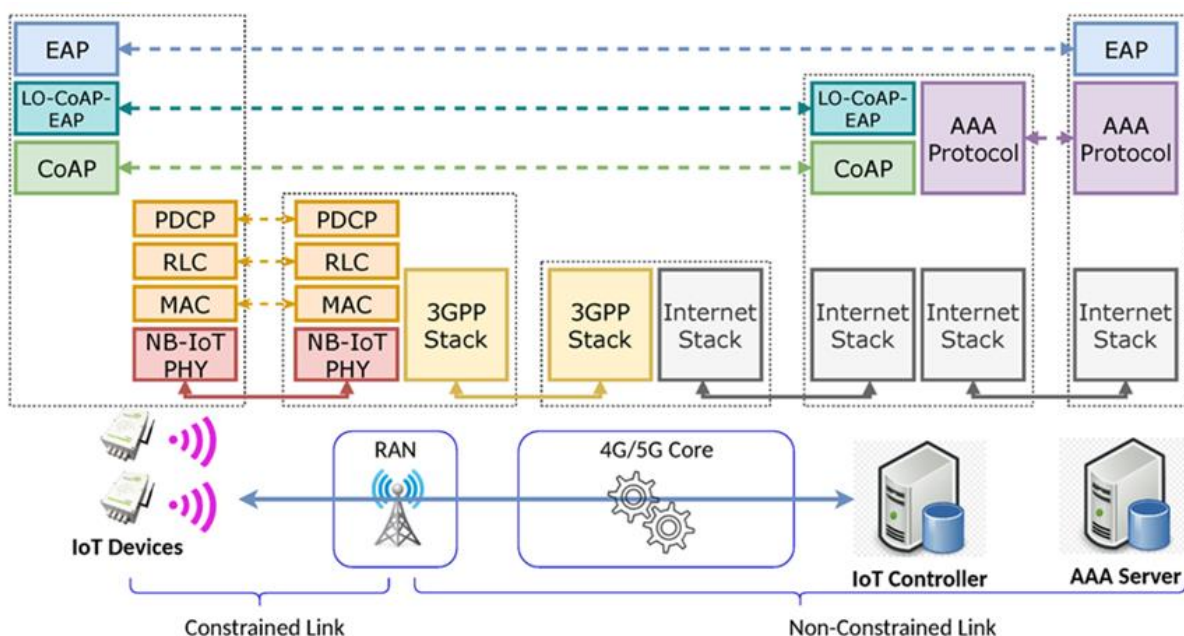


³ <https://lora-alliance.org/about-lorawan/>, zavihek Security; <https://lora-alliance.org/security/>

Varnost v mobilnih omrežjih

Mobilna omrežja 4. in 5. generacije s protokoli, kot so NB-IoT in Cat-M, zaradi uporabe tehnologije LTE omogočajo varno vključitev naprav v zaščiteno omrežje telekomunikacijskega operaterja. Ker je tehnologija LTE osnovana na protokolu IP, podpirajo vse standardne tehnologije zaščite podatkov v protokolu IP:

- IPsec navidezna zasebna omrežja
- TLS (Transport Layer Security) in DTLS (Datagram Transport Layer Security)
- CoAP (Constrained Application Protocol) omogoča avtentikacijo in šifriranje (CoAPs)
- MQTT (Message Queuing Telemetry Transport) omogoča avtentikacijo in šifriranje (MQTTs)



Arhitektura OPC UA omogoča varen in zanesljiv prenos podatkov z uporabo naslednjih pristopov:

- avtentikacija in avtorizacija na strežniku in klientu,
- varnost, ki temelji na veljavnih standardih (SSL/TLS),
- uporaba standardiziranih digitalnih potrdil X.509,
- uporaba standardizirane infrastrukture CA (angl. *Certificate Authority*),
- šifriranje podatkov z 128 in 256 bitnimi ključi,
- preprosta konfiguracija požarnega zidu.

3.2 FIWARE generični omogočevalci kot gradniki platforme IoT

FIWARE obsega množico gradnikov oz. t.i. generičnih omogočevalcev (angl. *Generic Enablers*, krajše *GE*), ki se štejejo za splošno namenske in neodvisne od področja uporabe, in skupaj z gradniki tretjih oseb omogoča hitrejši, lažji in cenejši razvoj pametnih rešitev, vključno s platformami IoT.

3.2.1 Pregled generičnih omogočevalcev FIWARE

Podajamo seznam nekaj izmed FIWARE generičnih omogočevalcev, ki se lahko uporabijo pri izgradnji platforme IoT:

sloj platforme IoT	FIWARE generični omogočevalec	vloga
južni	IoT agents	most med različnimi omrežnimi protokoli, pretvarjanje v/i z NGSI specifikacij
osrednji	Context Broker	izvajanje posodobitev, poizvedb ali naročanja na spremembe kontekstualnih informacij preko NGSI API-ja
	STH Comet	shranjevanje kratkoročne zgodovine kontekstualnih podatkov v MongoDB podatkovno bazo
	Cygnus	upravljanje zgodovine konteksta, ustvarjenega iz toka podatkov, ki ga je mogoče vnesti v več podatkovnih ponorov (npr. različne podatkovne baze in platforme za masovne podatke (angl. <i>Big Data</i>))
	QuantumLeap	shranjevanje kontekstualnih podatkov v podatkovne baze za časovne vrste kot sta CrateDB in Timescale

severni	WireCloud	vizualizacija podatkov s pomočjo prilagodljivih nadzornih plošč
varnostni	Keyrock	OAuth 2.0 IAM
	Wilma	PEP
	AuthZForce	PDP/PAP na podlagi standarda XAMCL

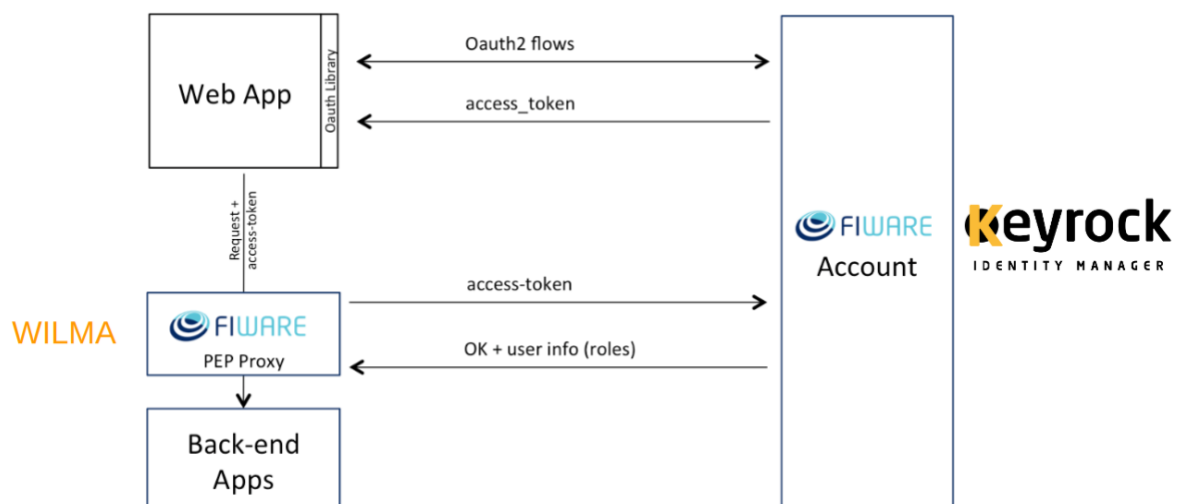
3.2.2 Ravni avtorizacije FIWARE

Spodaj so na kratko predstavljeni trije načini preverjanja pristnosti in nadzora dostopa, ki temeljijo na uporabi FIWARE generičnih omogočevalcev.

Raven 1: avtentikacija

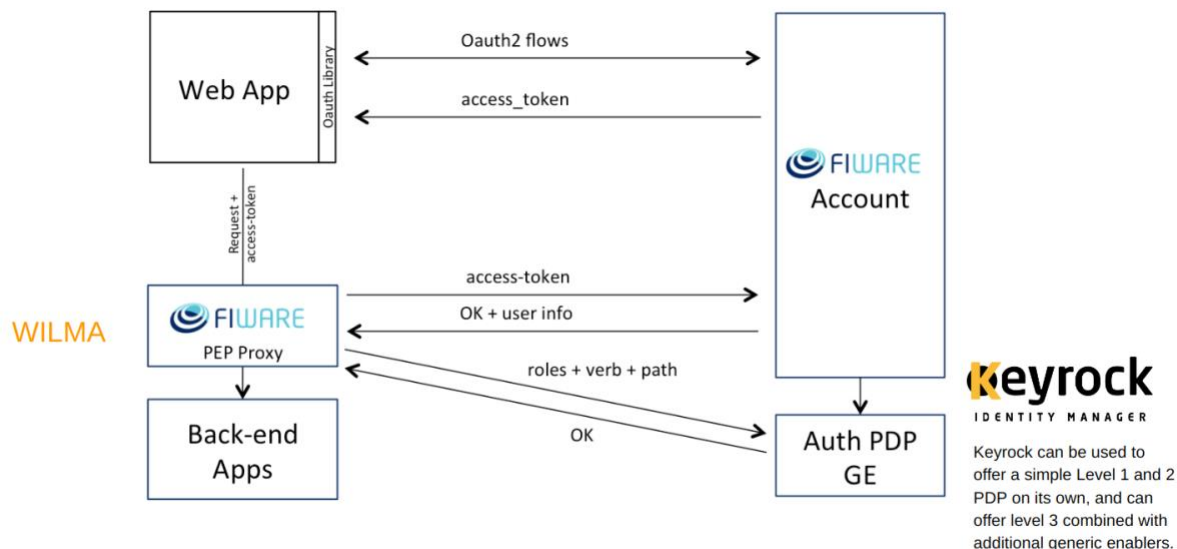
Na tej ravni se zgolj preverja pristnost odjemalca; tj. dovoli vsa dejanja vsakemu prijavljenemu odjemalcu in nobenih dejanj anonimnemu odjemalcu.

Odjemalec z IAM rešitvijo (Keyrock) izmenja uporabniške poverilnice za dostopni žeton, ki ga nato vključi v zahtevek. Zahtevek prestreže PEP (Wilma), ki odločitev o dostopu prepusti IAM rešitvi, ki igra vlogo PDP.



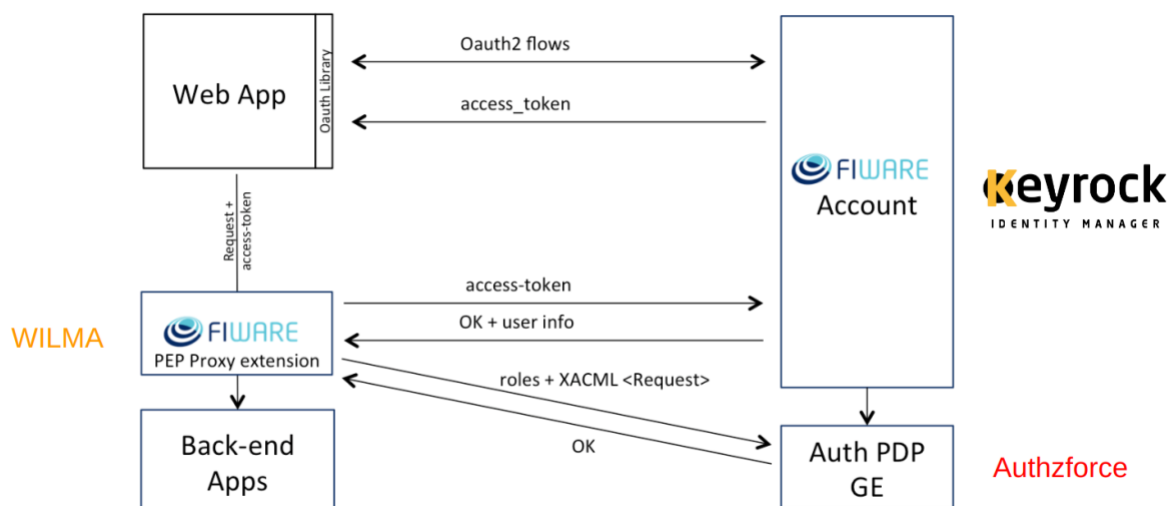
Raven 2: osnovna avtorizacija

IAM rešitev na tej ravni poleg izdajanja dostopnih žetonov in preverjanja pristnosti odjemalcev v okviru odločanja o dostopu kot PDP poskrbi tudi za nadzor dostopa (RBAC model).



Raven 3: napredna avtorizacija

V primeru uporabe ABAC modela za nadzor dostopa, IAM rešitev (Keyrock) delegira nadzor dostopa na Authzforce, ki podpira ABAC z uporabo XACML 3.0 standarda.



3.3 Alternativne možnosti gradnikov platforme IoT

Ob raziskovanju smo prišli do zaključka, da nekateri generični omogočevalci FIWARE morda premalo ponujajo - bodisi iz stališča samih funkcionalnosti, bodisi iz stališča njihovega razvoja in vzdrževanja. Posledično smo za poiskali alternativne možnosti, ki jih skupaj s primerjavo navajamo v tem podpoglavju.

K temu nas spodbuja tudi citat iz FIWARE dokumentacije:

“FIWARE is not about take it all or nothing. You are not forced to use these complementary FIWARE Generic Enablers but are free to combine them with other third-platform components to design the hybrid platform of your choice. FIWARE is also not limited to using a single security stack. Since the authorization token is held in a header, security is

effectively orthogonal to processing of a context broker, and the developer is free to choose their preferred Identity Manager, PDP and PEP. As an example, we can swap out the Wilma PEP Proxy and use the Kong API Management tool in coordination with a Kong Plugin for Keyrock.”

3.3.1 Prehod API namesto povratnega posredniškega strežnika PEP

Prehod API (angl. *API gateway*) je orodje za upravljanje API-ja, ki se nahaja med odjemalcem in zbirko zalednih storitev. Prehod API deluje kot povratni posredniški strežnik (angl. *reverse proxy*), ki sprejema zahteve programskega vmesnika (angl. *application programming interface*, krajše *API*). V ozadju združuje različne zaledne storitve, potrebnih za izpolnjevanje zahtevkov.

Naloga PEP proxy-a je zgolj ta, da prestreže zahtevek ter na podlagi odločitve PDP odjemalcu odobri ali zavrne dostop do zahtevanega vira.

Prehod API poleg avtentikacije in avtorizacije zahtevkov običajno ponuja še druge funkcionalnosti, med drugim:

- napredno usmerjanje (angl. *routing*) zahtevkov do ustreznih zalednih storitev,
- omejevanje dostopa na podlagi uporabe (angl. *rate limiting*),
- preoblikovanje vsebine zahtevka in odgovora na zahtevek,
- podpora za L4/L7 promet,
- avtomatsko posodabljanje certifikatov TLS in odstranjevanje šifriranja povezave TLS ob posredovanju zahtevka do zalednih storitev v varnem ozadju,
- beleženje zahtevkov,
- monitoring.

V zadnjem času se v FIWARE postavitvah namesto PEP proxy-a pojavlja prehod API, poimenovan **Kong** (več v poglavju 4.7.1).

Primerjava prehoda API **Kong** v primerjavi s povratnim posredniškim strežnikom PEP **FIWARE Wilma**:

	Kong	FIWARE Wilma
Github stars	36,4 tisoč	26
prednosti	<ul style="list-style-type: none"> • več funkcionalnosti (API gateway) • “native” podpora za K8s (ingress controller) • robustnost, hitrost • večja skupnost => boljša podpora => lažje vzdrževanje • GUI (Kong Manager) • vtičniki • se že uporablja v produkciji 	<ul style="list-style-type: none"> • enostavna namestitev in konfiguracija • brezplačna uporaba • “native” Fiware komponenta

slabosti	<ul style="list-style-type: none"> • kompleksnejša nastavitve • napredne funkcije so lahko plačljive 	<ul style="list-style-type: none"> • omejena funkcionalnost (zgolj PEP) • brez GUI • vprašljiva obstojnost primerov uporabe v produkciji
-----------------	--	---

Navajamo še dve odprtokodni rešitvi prehoda API:

- Tyk (<https://tyk.io/>),
- GlooEdge (<https://github.com/solo-io/gloo>).

3.3.2 Alternativne rešitve za upravljanje identitet in dostopov

Keycloak (več v poglavju 4.7.2) je popularna odprtokodna rešitev za upravljanje identitet in dostopov.

Na FIWARE predstavitvi “*Kong-Keycloak extension; Keycloak as PDP in a FIWARE platform*” (dosegljiva tukaj: <https://github.com/wistefan/presentations/blob/main/summit-gran-canaria/keycloak/FGS22-Keycloak.pdf>) so FIWARE razvijalci navedli nekaj razlogov, zakaj priporočajo uporabo Keycloak-a:

- aktivno vzdrževanje => pogoste varnostne posodobitve,
- podpora za OAuth 2.0 in OIDC,
- se že uporablja v obstoječih FIWARE sistemih, npr.: <https://github.com/FIWARE-Ops/marinera>, demo: <https://keycloak.firmware.dev/>.

Primerjava **Keycloak-a** in **FIWARE Keyrock-a**:

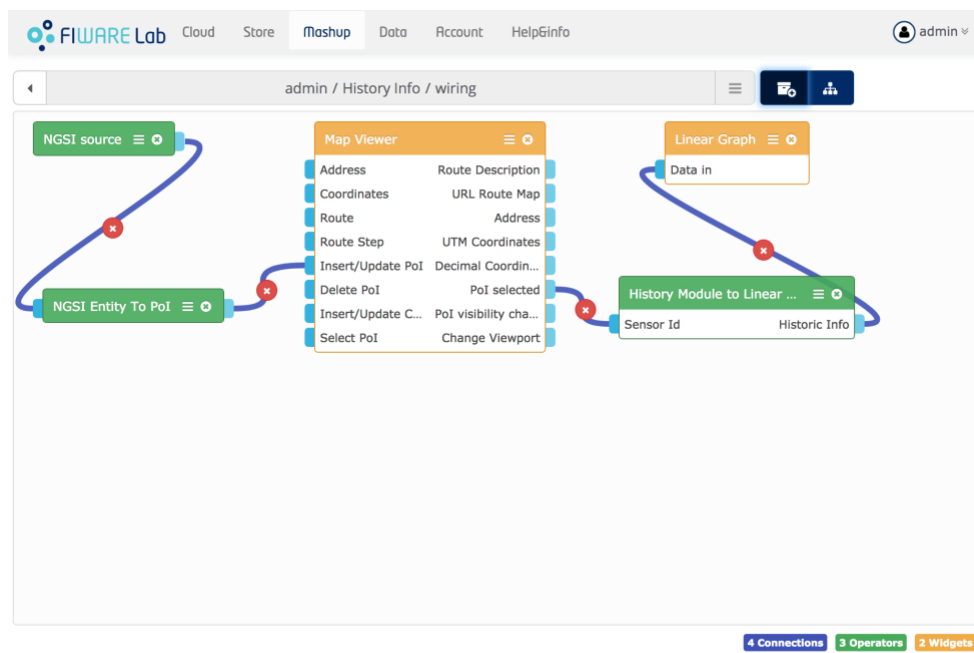
	Keycloak	FIWARE Keyrock
Github stars	15,7 tisoč	31
prednosti	<ul style="list-style-type: none"> • več funkcionalnosti • se že uporablja v produkciji, tudi v FIWARE sistemih 	<ul style="list-style-type: none"> • “native” Fiware komponenta • bolj prilagojen za uporabo v IoT domenah
slabosti	<ul style="list-style-type: none"> • bolj splošen 	<ul style="list-style-type: none"> • manjša skupnost • slabša podpora • vprašljivo vzdrževanje • vprašljiva obstojnost primerov uporabe v produkciji

3.3.3 Vizualizacija podatkov

FIWARE za vizualizacijo podatkov ponuja generični omogočevalec **WireCloud**.



2017 © FIWARE. The use of FIWARE Lab services is subject to the acceptance of the [Terms and Conditions](#), [Privacy Policy](#) and [Cookies Policy](#).



2017 © FIWARE. The use of FIWARE Lab services is subject to the acceptance of the [Terms and Conditions](#), [Privacy Policy](#) and [Cookies Policy](#).

Po pregledu smo ugotovili, da sta se razvoj in vzdrževanje ustavila v letu 2021, prav tako so primeri uporabe v konkretnih primerih pokazali določene slabosti. Iz tega vidika odsvetujemo njegovo resno uporabo.

Spodaj na kratko predstavljamo dve alternativni možnosti.

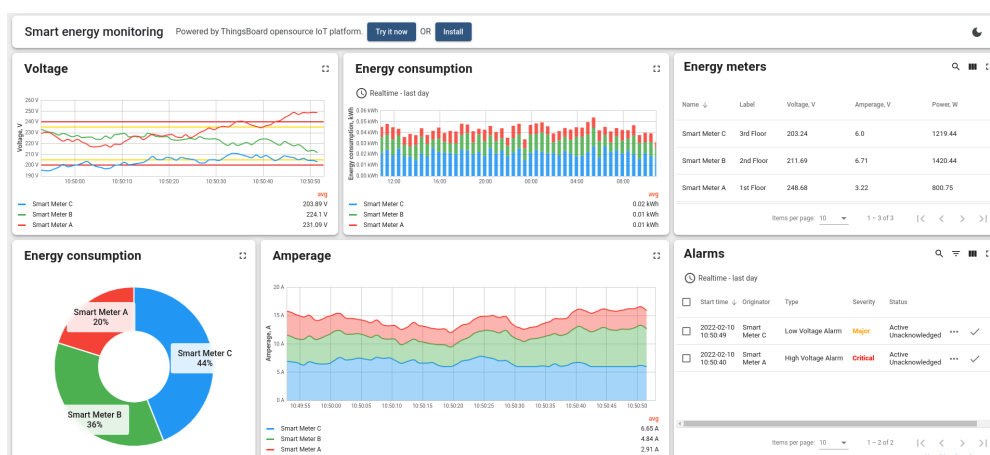
Grafana

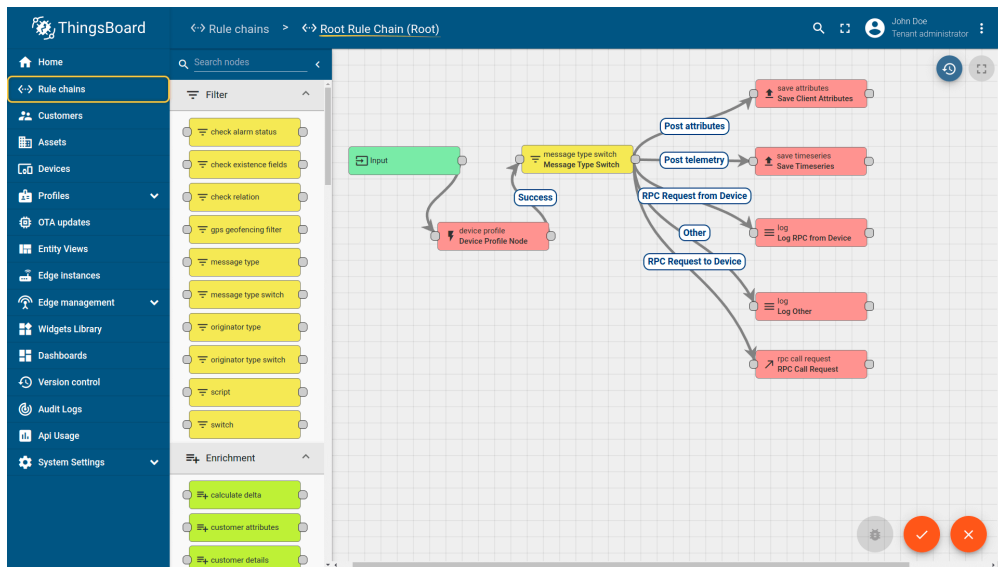
Grafana je odprtokodna rešitev, ki omogoča poizvedovanje, vizualizacijo, opozarjanje in razumevanje podatkov (metrik). Omogoča gradnjo dinamičnih nadzornih plošč ter podpira različne vire podatkov, med drugim podatkovne baze, ki jih je s pomočjo FIWARE GE mogoče integrirati s Context Broker-jem.



ThingsBoard

ThingsBoard je odprtokodna platforma IoT za zbiranje, obdelavo in vizualizacijo podatkov ter upravljanje naprav interneta stvari. Več kot 30 prilagodljivih gradnikov omogoča izdelavo bogatih nadzornih plošč po meri končnega uporabnika za večino primerov uporabe interneta stvari. Z uporabo t.i. "Rule Engine" je možno tudi ustvarjanje zapletenih verig pravil (angl. *rule chain*) za obdelavo podatkov ter prilagajanje specifičnim primerom njihove uporabe.





4 Predlog načina izvedbe pilotne platforme IoT

V tem poglavju je predstavljena predlagana arhitektura platforme IoT, sestavljena na podlagi dosedanjih ugotovitev. V prvem podpoglavju se nahaja visokonivojski pogled arhitekture. V nadaljevanju so po slojih podrobneje predstavljeni gradniki platforme IoT.

4.1 Visokonivojski pogled arhitekture

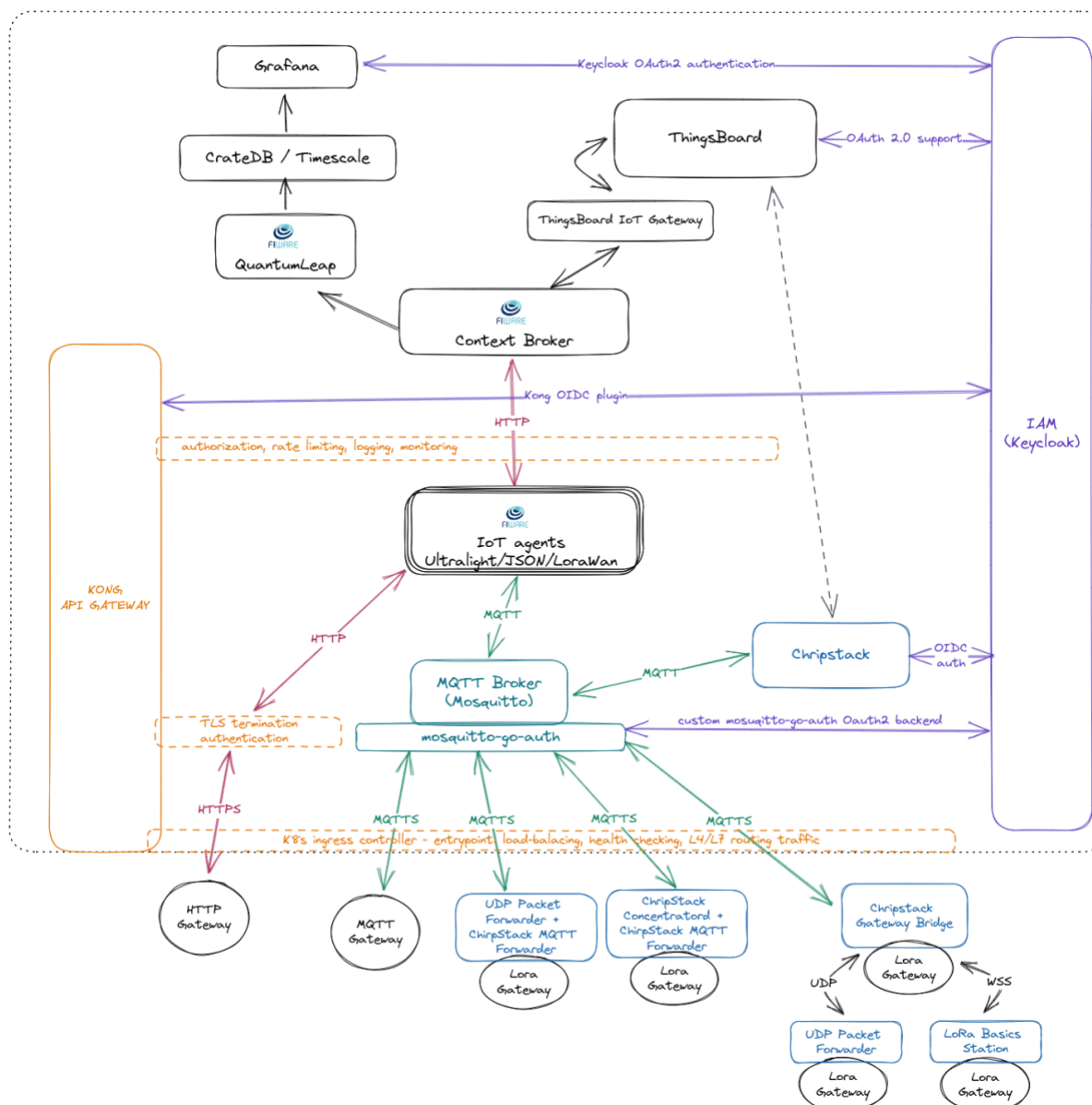
- Za opazovanje stvari iz realnega sveta se uporabljajo različni senzorji.
- Naloga prehodov na robu je, da iz množice senzorjev in z njimi povezanih protokolov pridemo do obvladljivega načina prenosa podatkov v platformo. Predlagamo uporabo TCP/IP prehodov in Lora prehodov z nameščeno ustrezno programsko opremo.
- Za vstopni točki v platformo IoT sta zaenkrat predvidena dva aplikacijska omrežna protokola, ki temeljita na TCP/IP:
 - HTTPS,
 - MQTT.

V primeru Lora prehodov se priporoča uporaba namestitev programske opreme *Chirpstack MQTT Forwarder*, ki poskrbi za ustrezno translacijo protokolov na MQTT.

- Prenos podatkov med prehodom na robu in vstopno točko platforme IoT poteka po zaščiteni povezavi; za TCP/IP povezave se uporablja protokol TLS => HTTPS, MQTTS.

V slučaju direktne uporabe protokola UDP, se varno povezavo lahko vzpostavi s pomočjo protokola DTLS.

- Ob vstopu v platformo IoT zahtevke (sporočila) najprej prestreže prehod API: Kong, ki poskrbi za njihovo ustrezno preusmerjanje (angl. *routing*) in omejevanje (angl. *rate limiting*), po potrebi tudi za beleženje dostopov (angl. *access logging*). V primeru komunikacije preko protokola HTTPS izvede tudi odstranitev zaščitene povezave ("TLS termination") in preverjanje pristnosti.
- Kong sporočila glede na aplikacijski omrežni protokol posreduje v zaledni del:
 - HTTPS sporočila posreduje direktno agentom IoT,
 - MQTT sporočila posreduje posredniku sporočil *Mosquitto*.
- Posrednik sporočil MQTT poskrbi za preverjanje pristnosti. Nanj so preko nezaščitene MQTT povezave povezani tudi *Chirpstack* in agenti IoT.
- Agenti IoT so zadolženi za (1) pretvorbo iz različnih formatov sporočil (npr. Ultralight 2.0) v standardni format NGSI in (2) pretvorbo iz različnih aplikacijskih omrežnih protokolov (npr. MQTT) v protokol HTTPS.
- Context Broker izpostavlja NGSI API, preko katerega se nanj povežejo agenti IoT. Kong na tej točki med drugim poskrbi za nadzor dostopa.
- Za vizualizacijo podatkov sta trenutno predvideni dve možnosti:
 - QuantumLeap se naroči na podatke Context Broker-ja in jih shranjuje v podatkovno bazo CrateDB, ki je vir podatkov za vizualizacije v Grafani
 - S pomočjo programske opreme ThingsBoard IoT Gateway se vzpostavi povezava med Context Broker-jem in ThingsBoard-om, ki omogoča gradnjo bogatih nadzornih plošč.
- Za upravljanje identitet in dostopov je v celotnem skladu predviden Keycloak, na katerega se s pomočjo integracij povežejo ostali gradniki.



4.2 Senzorika

Podatki senzorjev se prek prehodov (angl. gateway), ki omogočijo tudi oddaljen dostop do teh podatkov po internetnem omrežju (protokol IP), pošiljajo IoT agentom. Predlagamo, da naročnik vse senzorje, ki bodo predmet povezovanja v ta sestav, opremi z ustreznimi pretvorniki, da jih bo moč povezati na prehode. Priporočljivo je, da tako pretvorbo izvede izvajalec, ki je vzpostavil senzorje, ali vzdrževalec, ker pozna vse tehnične parametre te postavitve.

Primer: vzorčni primer prvega uporabnika uporablja senzorje za temperaturo zraka v notranjih prostorih in števec porabe električne energije, ki so povezani po protokolu MODBUS na osrednji prehod Endress+Hauser Memograph M RSG 45. Ta prehod omogoča oddaljeni dostop po protokolu HTTP za interaktivni dostop in po protokolu OPC za strojni dostop.

4.3 Podprti prehodi IoT na robu

Poleg TCP/IP in LoRaWAN prehodov je smiselno preučiti arhitekturo OPC UA, ki se kaže uporabna za realizacijo prvega uporabniškega scenarija.

4.3.1 Prehod TCP/IP

TCP/IP komunikacijski prehod je naprava ali programska oprema, ki omogoča povezavo med dvema različnima omrežji, ki uporabljata različne protokole za prenos podatkov. Glavna naloga TCP/IP prehoda je zagotoviti prevajanje med različnimi protokoli, tako da lahko naprave v enem omrežju komunicirajo z napravami v drugem omrežju.

TCP/IP prehod se uporablja za povezovanje naprav, ki uporabljajo različne protokole, kot so Ethernet, Wi-Fi, Bluetooth, RS-232, RS-485, CAN, itd. Prehod sprejme podatke, ki jih prejme od naprave v enem omrežju, pretvori v standardne podatke IP protokola in jih pošlje preko drugega omrežja. To omogoča medsebojno povezovanje naprav in sistemov, ki uporabljajo različne protokole, brez potrebe po spreminjanju same naprave ali sistema.

TCP/IP prehod je lahko samostojna fizična naprava, kot je na primer usmerjevalnik ali stikalo, ali pa programska oprema, ki se izvaja na strežniku ali vgrajenem sistemu.

4.3.2 Prehod LoRaWAN

LoRaWAN komunikacijski prehod je naprava, ki služi kot most med LoRaWAN IoT napravami in Internetom. Prehod je namenjen za sprejemanje podatkov iz LoRaWAN naprav v bližini, jih pretvarjanje v standardne podatke IP protokola in pošiljanje preko interneta na strežnik, ki omogoča obdelavo in uporabo podatkov.

LoRaWAN komunikacijski prehodi so običajno povezani preko javne dostopne spletne omrežne povezave, kot so Ethernet, Wi-Fi ali mobilnega omrežja 3G/4G na LoRaWAN omrežni strežnik LNS.

Vloga LoRaWAN komunikacijskega prehoda je nujna, saj zagotavlja konvergenco med LoRaWAN IoT omrežjem in protokolom TCP/IP ter omogoča učinkovito upravljanje in spremljanje naprav, ki so povezane v IoT omrežje.

4.3.3 Arhitektura OPC UA

OPC UA je odprtokodni standard za komunikacijo v industrijski avtomatizaciji. Namenjen je varnemu in zanesljivemu povezovanju višje nivojskih programskih sistemov s strojno opremo na nižjih nivojih. Temelji na konceptu odjemalec – strežnik (angl. client - server), kjer je strežnik aplikacija, ki svoje podatke nudi drugim aplikacijam, odjemalec pa od nje zahteva

oz. pridobiva podatke. Razvijalci v praksi stremijo k temu, da so aplikacije odjemalci in strežniki hkrati.

UPC UA temelji na storitveno orientirani arhitekturi, ki ima tri osnovne entitete: strežnik (angl. server), odjemalec (angl. client) in prehod (angl. gateway).

Strežnik OPC UA predstavlja točko arhitekture OPC UA s standardiziranim komunikacijskim vmesnikom, prek katere se izbrani podatki nudijo drugim aplikacijam oz. odjemalcem. Z implementacijo strežnika v strojno opremo se zagotovi standardiziran in varen dostop do podatkov iz različnih naprav. Strežnik je lahko realiziran kot samostojna programska oprema ali pa kot del vgrajene strojne opreme, npr. na krmilniku PLC. Strežnik predstavlja izvor podatkov za odjemalce. Pogosto je zato že vgrajen v senzorskih sistemih.

Odjemalec OPC UA bere podatke iz katerega koli strežnika OPC UA. Tipična uporaba odjemalca so aplikacije, ki so odvisne od izmenjave podatkov.

Prehod OPC UA pretvori določen komunikacijski protokol na vmesnik OPC UA. Tipično se prevaja protokole IEC 103, IEC 104, Modbus, RS-485, M-bus, MQTT itd. na poenoteno platformo, kar omogoča tudi priklop starejših naprav na skupno komunikacijsko arhitekturo.

Lastnosti arhitekture OPC UA

V primerjavi z obstoječimi heterogenimi rešitvami različnih izvajalcev ponuja OPC UA številne prednosti:

- varen in zanesljiv prenos podatkov
- širok nabor vgrajenih podatkovnih modelov, npr. za dostop do podatkov DA (angl. Data Access), dostop do arhiva podatkov HDA (angl. Historical Data Access) in alarmiranje AE (angl. Alarm and Events),
- neodvisnost od platform (operacijski sistem, programski jezik, strojna oprema),
- uporaba standardnih protokolov, ki temeljijo na TCP/IP,
- skalabilnost, kar omogoča delovanje na vgrajenih napravah (mikrokontrolerji), krmilnikih PLC, strežnikih in aplikacijah v oblaku,
- vključevanje starejših sistemov z ne-OPC UA protokoli v arhitekturo s pomočjo prehodov OPC UA,
- podpira ga veliko število proizvajalcev, npr. Siemens, ControlLogix, Mitsubishi Melsec-Q, Rockwell, Microsoft itd.,
- visoka razpoložljivost komunikacije
 - redundantna konfiguracija za OPC-UA odjemalec in strežnik,
 - OPC-UA Heartbeat za spremljanje aktivnosti povezave v obe smeri,
- visoka zmogljivost
 - zmožnost obdelave velike količine podatkovnih točk (naprav),
 - uporaba učinkovitega binarnega protokola »OPC-UA Binary« in »UA TCP« z majhnimi podatki režije,
 - majhna poraba sistemskih sredstev (CPU, delovni spomin),
- zelo dobra združljivost različnih proizvajalcev naprav s podporo OPC UA.

4.4 Južni sestav platforme IoT

4.4.1 Vstopna točka (prehod API)

Kong (prehod API) na varen način izpostavlja API, s tem navzven ponuja funkcionalnosti platforme IoT in služi kot vstopna točka do zalednih storitev. Več o Kong-u v poglavju 4.7.1.

Sporočila, ki se prenašajo po protokolu HTTP, se po uspešnem preverjanju pristnosti direktno posredujejo naprej do agentov IoT, pri ostalih pa je potrebna uporaba posrednika sporočil MQTT in/ali omrežnega strežnika LNS (LoRaWAN).

4.4.2 Posrednik sporočil MQTT

MQTT je standardni protokol sporočanja za internet stvari (IoT). Zasnovan je kot izredno lahek prenos sporočil po načinu publish/subscribe, ki je idealen za povezovanje oddaljenih naprav z majhnim odtisom kode in minimalno porabo pasovne širine omrežja.

Za komunikacijo preko protokola MQTT je potreben posrednik sporočil (angl. *message broker*), ki skrbi za prenos sporočil med odjemalci. **Eclipse Mosquitto** je odprtokodni posrednik sporočil, ki implementira protokol MQTT verzij 5.0, 3.1.1 in 3.1.

Na posrednik sporočil so povezani naslednji odjemalci:

- prehodi IoT na robu,
- agenti IoT,
- omrežni strežnik LoRaWAN.

V spodnji preglednici so navedene vloge odjemalcev:

odjemalec / vloga	izdajatelj sporočil (angl. publisher)	naročnik na sporočila (angl. subscriber)
prehodi IoT na robu	telemetrija senzorjev	ukazi za aktuatorje
agenti IoT	ukazi za aktuatorje	telemetrija senzorjev
omrežni strežnik LoraWan	ukazi za aktuatorje (naprave LoRa razreda C) za prehode LoRa telemetrija senzorjev LoRa za agente IoT	telemetrija senzorjev s prehodov LoRa ukazi za aktuatorje LoRa iz agentov IoT

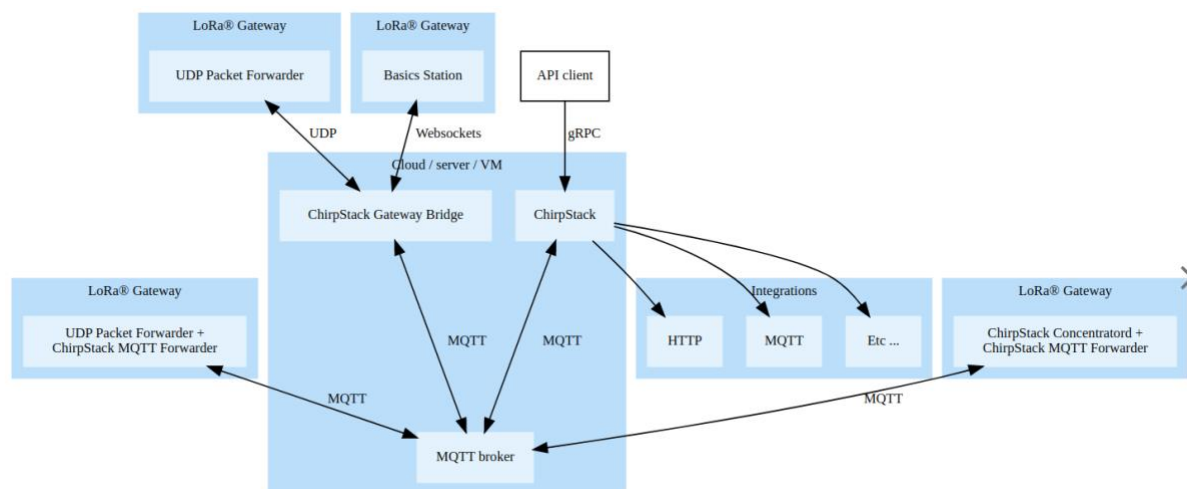
Za komunikacijo med prehodi IoT na robu in posrednikom sporočil se na podlagi digitalnih potrdil uporablja varna povezave MQTTS.

4.4.3 Omrežni strežnik LoRaWAN

Odprtokodna rešitev **ChirpStack** ponuja naslednje gradnike:

- **ChirpStack Concentrator** je odprtokodni demon, ki deluje kot koncentrador LoRa sporočil. Izpostavlja API na osnovi ZeroMQ, ki ga lahko uporablja ena ali več posredniških aplikacij za interakcijo s strojno opremo prehoda LoRaWAN.
- **ChirpStack MQTT Forwarder** je odprtokodni posrednik JSON / Protobuf sporočil preko protokola MQTT, ki lahko uporablja Semtech UDP Packet Forwarder ali ChirpStack Concentrator.
- **ChirpStack Gateway Bridge** je odprtokodni most, ki sporočila, prejeta od Semtech UDP Packet Forwarder ali Semtech Basics Station, pretvori v MQTT. Nameščen je lahko na prehodu na robu ali znotraj platforme IoT.
- **ChirpStack** je odprtokodni omrežni strežnik LoRaWAN, ki se lahko uporablja za vzpostavitev omrežij LoRaWAN. ChirpStack zagotavlja spletni vmesnik za upravljanje prehodov, naprav in najemnikov (angl. *tenants*) ter nastavitve podatkovnih povezav z glavnimi ponudniki storitev v oblaku, podatkovnimi zbirkami in storitvami, ki se običajno uporabljajo za obdelavo podatkov z naprav LoRa.

Poleg tega je na voljo tudi **ChirpStack Gateway OS**, odprtokodni vgrajeni operacijski sistem, ki temelji na Linuxu in lahko deluje na različnih modelih prehodov LoRa. Cilj je olajšati začetek uporabe omrežja LoRaWAN in odprtokodnega omrežnega strežnika LoRaWAN ChirpStack z minimalnimi koraki, potrebnimi za konfiguracijo prehodov na robu.



4.4.4 Agenti IoT

IoT agenti služijo pretvorbi sporočil po drugih standardih iz nabora IoT v standard NGSI v2 ali NGSI-LD. Omogočajo pretvorbo v obe smeri, torej od senzorjev proti osrednjemu sestavu in obratno, od osrednjega sestava proti aktuatorjem.

Trenutno so v katalogu FIWARE podprti naslednji protokoli:

- IoT Agent for JSON - pretvornik med HTTP/MQTT sporočili (vsebina v obliki JSON) in NGSI
- IoT Agent for LWM2M - pretvornik med protokolom Lightweight M2M in NGSI
- IoT Agent for Ultralight - pretvornik med HTTP/MQTT sporočili (vsebina v obliki UltraLight 2.0) in NGSI
- IoT Agent for LoRaWAN - pretvornik med protokolom LoRaWAN in NGSI
- IoT Agent for OPC-UA - pretvornik med protokolom OPC Unified Architecture in NGSI

- IoT Agent for Sigfox - pretvornik med protokolom Sigfox in NGSI
- IoT Agent for ISOXML - pretvornik med protokolom ISOXML/ADAPT za kmetijsko mehanizacijo in NGSI

Poleg teh je na voljo tudi knjižnica IoT Agent library za razvoj IoT Agentov po meri.

IoT agenti naj bi že podpirali standard NGSI-LD, vendar bo treba izvesti podrobno testiranje. Če ta podpora ne bo zadostna, bo treba uporabiti standard NGSI v2, kar potegne za sabo izbiro ustreznega Context Brokerja.

Repozitorij: <https://github.com/FIWARE/catalogue#interface-with-iot-robots-and-third-party-systems>

Za poenostavitev konfiguracij, kjer je povezanih večje število IoT agentov, priporočajo uporabo IoT Agent Managerja, ki na severni strani izpostavlja eno povezavo proti Context Brokerju, na južni strani pa preusmerja promet proti tistemu IoT agentu, ki skrbi za dotični protokol. Uporabo IoT Agent Managerja bo treba še preučiti.

Repozitorij: <https://github.com/telefonicaid/iotagent-manager>

4.5 Osrednji sestav platforme IoT

4.5.1 Fiware Context Broker

Združenje FIWARE spremlja razvoj treh implementacij Context Brokerja: Orion, Scorpio in Stellio. Za uporabo v sestavu naročnika sta primerna prva dva, tretji je bolj poenostavljen. Izmed prvih dveh je bolj primeren Orion, ker pokriva največ znanih primerov uporabe, ki jih v javnosti predstavlja združenje FIWARE. Razvija ga podjetje Telefonica iz Španije. Združenje FIWARE potem razvite funkcionalnosti prenaša v svojo različico Orion-LD, ki je usklajena s specifikacijami NGSI-LD. Standard NGSI v2 so nadgradili z dodano funkcionalnostjo Linked Data v nov standard NGSI-LD, ki pa ni združljiv s starim. Razvoj implementacij Context Brokerja sledi temu tako, da v ločeno verzijo Orion-LD vgrajujejo večinoma zmožnosti po novem standardu, podpora staremu standardu pa odraža stanje ob odcepitvi repozitorija Orion-LD od repozitorija Orion in je ne dopolnjujejo. Med načrtovanjem implementacije sestava zato predvidevamo končno odločitev, ali uporabiti stari standard NGSI v2 ali novega NGSI-LD. To je odvisno tudi od podpore ostalih gradnikov enemu ali drugemu standardu.

Naslov repozitorija Telefonice za Orion (standard NGSI v2):

<https://github.com/telefonicaid/fiware-orion/>

Naslov repozitorija FIWARE za Orion-LD (standard NGSI-LD):

<https://github.com/FIWARE/context.Orion-LD>

4.6 Severni sestav platforme IoT

4.6.1 QuantumLeap, CrateDB, Grafana

QuantumLeap je storitev REST za shranjevanje, poizvedovanje in iskanje prostorsko-časovnih podatkov NGSI v2 in NGSI-LD (eksperimentalna podpora). QuantumLeap pretvori

polstrukturirane podatke NGSI v tabelarno obliko in jih shrani v podatkovno zbirko časovnih vrst, pri čemer vsak zapis podatkovne zbirke poveže s časovnim indeksom in, če je podatek prisoten, z lokacijo na Zemlji. Odjemalci lahko nato pridobijo entitete NGSI s filtriranjem nizov entitet prek časovnih razponov in prostorskih operatorjev. Funkcionalnost poizvedb, ki je na voljo prek vmesnika REST, je precej osnovna, za bolj zapletene poizvedbe morajo odjemalci običajno neposredno dostopati do podatkovne zbirke. Za shranjevanje podpira podatkovne zbirke CrateDB in Timescale.

CrateDB je porazdeljena zbirka podatkov. Združuje domačnost SQL z razširljivostjo in prilagodljivostjo NoSQL, kar razvijalcem omogoča:

- uporabo SQL za obdelavo vseh vrst podatkov, tako strukturiranih, kot nestrukturiranih,
- izvajanje poizvedb SQL z veliko hitrostjo,
- preprosto skaliranje.

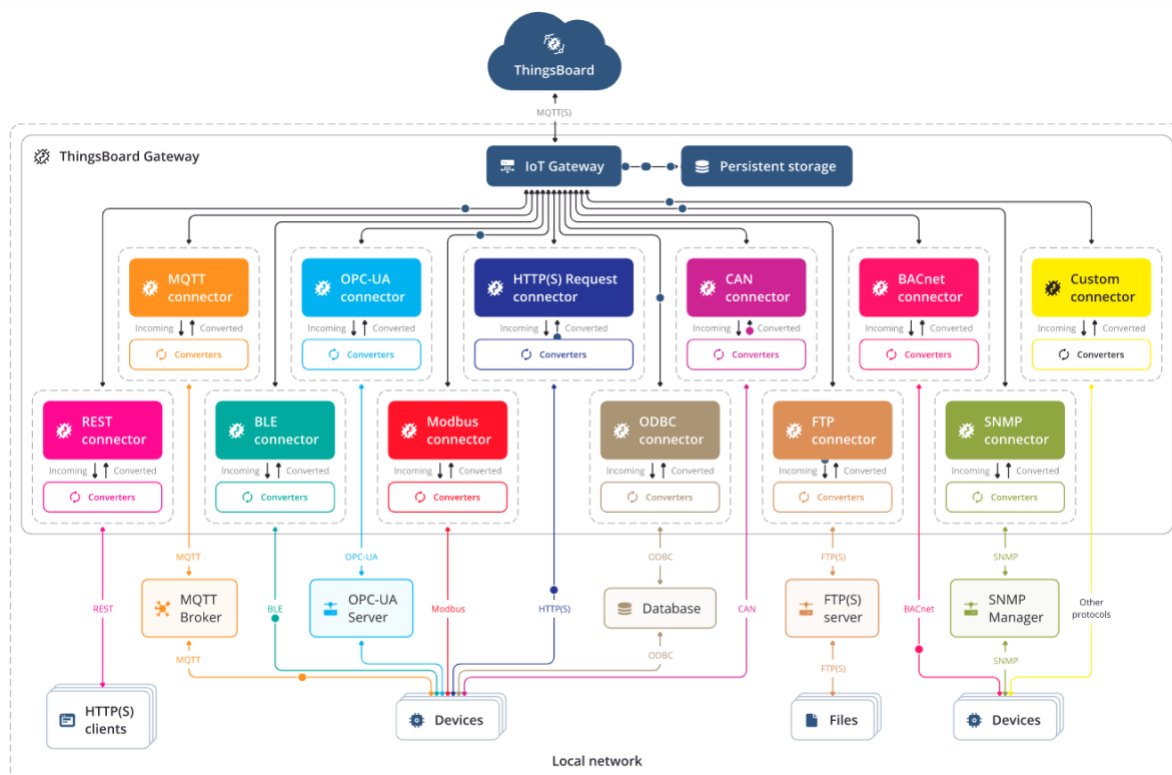
QuantumLeap se naroči na podatke iz Context Broker-ja in jih zapisuje v podatkovno bazo CrateDB.

Kot zapisano zgoraj, CrateDB uporablja SQL za poizvedbe. Posledično se za vizualizacijo podatkov v Grafani kot podatkovni vir (angl. *data source*) uporabi PostgreSQL. Primer: <https://crate.io/blog/visualizing-time-series-data-with-grafana-and-cratedb> .

4.6.2 Thingsboard

Thingsboard IoT gateway (<https://github.com/thingsboard/thingsboard-gateway>) je odprtokodna rešitev, primarno namenjena integraciji naprav IoT, ki so povezane s starimi sistemi in sistemi tretjih oseb, v ThingsBoard.

Med drugim omogoča shranjevanje podatkov (npr. senzorjev), pošiljanje ukazov iz Thingsboard-a (namenjenih aktuatorjem) in avtomatsko vključitev naprav IoT v ThingsBoard.



Thingboard IoT Gateway ponuja različne priključke (angl. *connectors*), ki s pomočjo pretvornikov (angl. *converters*) podatke pretvorijo v format, ki ustreza ThingsBoard-u oz. na drugi strani sistemu, iz/v katerega želimo brati/pošiljati podatke. Za integracijo s Context Broker-jem bi morda bilo potrebno razviti priključek po meri ("Custom connector").

Teoretično bi torej lahko ThingsBoard s pomočjo ThingsBoard IoT Gateway-a uporabili za:

- shranjevanje in vizualizacijo podatkov iz senzorjev,
- upravljanje aktuatorjev,
- register naprav IoT (angl. *IoT device registry*).

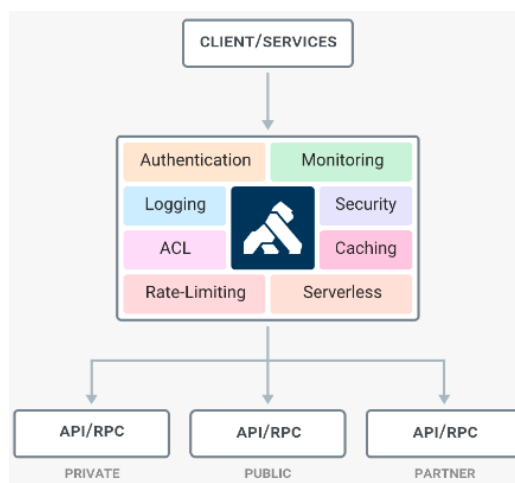
ThingsBoard IoT Gateway se naroči na podatke Context Broker-ja. Ko v Context Broker prispe nova vrednost z naprave IoT (npr. senzorja), se posreduje ThingsBoard IoT Gateway-u. Ta podatke pretvori v ustrezno obliko in jih preko povezave MQTT pošlje v ThingsBoard. Če naprava še ne obstaja, se avtomatsko doda v register naprav ThingsBoard-a. Na drugi strani lahko uporabnik iz ThingsBoard-a pošlje ukaz aktuatorju. ThingsBoard v tem primeru sporočilo z ukazom posreduje na ThingsBoard IoT Gateway, ki poskrbi, da se pretvori v format NGSI in pošlje zahtevo za posodobitev podatkov v Context Broker. Le-ta preko agenta IoT sporoči določenemu prehodu na robu, ki zahtevo ustrezno posreduje do dotičnega aktuatorja.

4.7 Varnost platforme IoT

4.7.1 Kong

Funkcionalnosti

Kong ali **Kong API Gateway** je “cloud-native”, platformno agnostični, skalabilni prehod API, ki ga odlikujeta visoka zmogljivost in razširljivost s pomočjo vtičnikov.



Med drugim podpira sledeče funkcionalnosti:

- preverjanje pristnosti in nadzor dostopa,
- napredne možnosti usmerjanja (angl. *routing*), uravnoveženja obremenitve (angl. *load balancing*),
- preoblikovanje oblike vsebine zahtevkov in odgovorov na zahteve (angl. *request/response transformations*),
- TLS terminacija,
- podpora za različne protokole na prenosnem (L4) in aplikacijskem (L7) omrežnem sloju.

Kong deluje v sistemu Kubernetes zahvaljujoč uradnemu kontrolniku ***Kubernetes Ingress Controller***.

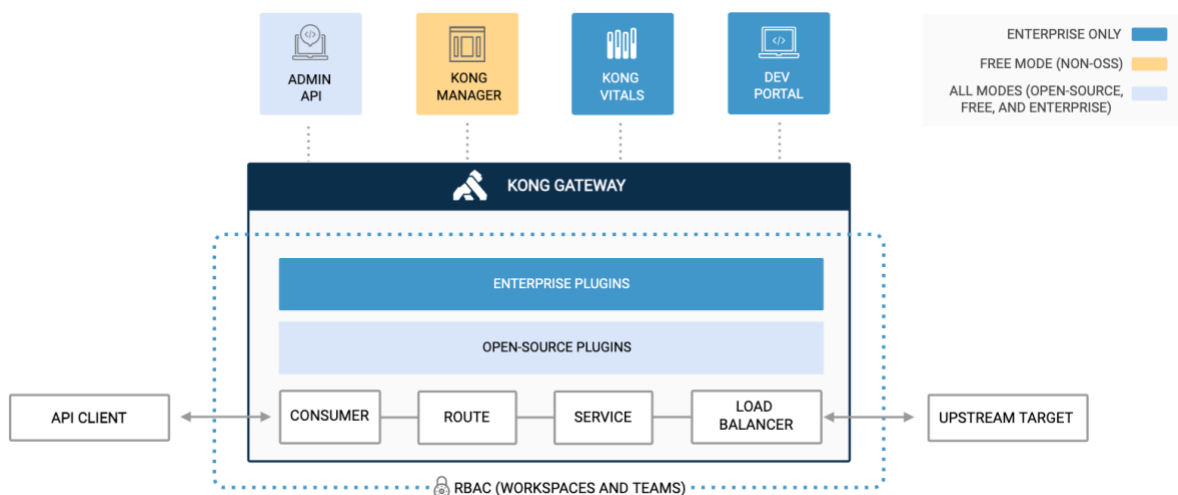
Različice Kong-a

Kong se lahko namesti na dva načina: (1) v oblaku s storitvijo Kong Konnect in (2) lokalno (angl. *on-premises*).

Na voljo so naslednje različice Kong-a:

- **KONG OSS (open source)**
 - Na voljo so zgolj nekatere funkcionalnosti in odprtokodni vtičniki.
 - Upravljanje je možno zgolj preko API-ja.
- **Kong FREE**
 - Vse iz Kong OSS.

- Dodatno: upravljanje s pomočjo uporabniškega vmesnika upravljalne konzole Kong Manager
- **Kong Premium** (\$250 / storitev, na voljo zgolj za postavitev v oblaku (Kong Konnect))
 - Dodatne funkcionalnosti in plačljivi vtičniki.
- **Kong Enterprise**
 - Še več dodatnih funkcionalnosti:
 - **Kong Dev Portal** se uporablja za uvajanje novih razvijalcev in ustvarjanje dokumentacije API, ustvarjanje strani po meri, upravljanje različic API-ja in varen dostop razvijalcev.
 - **Kong Vitals** zagotavlja uporabne metrike o stanju in zmogljivosti vozlišč prehoda Kong ter metrike o uporabi API-jev.
 - **RBAC model nadzora dostopa** za uporabnike, ki upravljajo Kong,
 - Še več plačljivih vtičnikov.



4.7.2 Keycloak

Funkcionalnosti in arhitektura

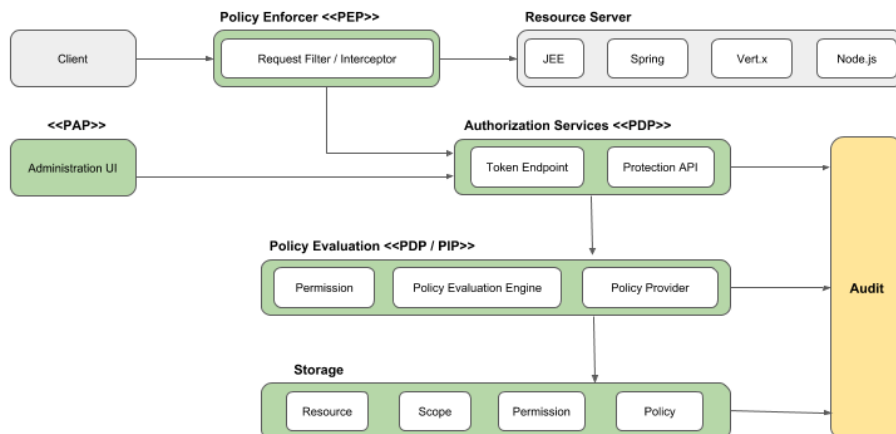
Keycloak je odprtokodna rešitev za upravljanje identitet in dostopov (IAM), ki temelji na standardnih protokolih in zagotavlja podporo za OpenID Connect, OAuth 2.0 in SAML.

Izpostavljene funkcionalnosti:

- **Upraviteljska konzola:** Prek upraviteljske konzole lahko skrbniki centralno upravljajo vse vidike strežnika Keycloak.
- **Konzola za upravljanje računov:** V konzoli za upravljanje računov lahko uporabniki upravljajo svoje račune. Posodobijo lahko profil, spremenijo gesla in nastavijo dvofaktorsko preverjanje pristnosti.
- **Storitve avtorizacije:** Če avtorizacija na podlagi vlog (RBAC) ne pokriva potreb, ponuja Keycloak tudi storitve avtorizacije s fino znatostjo (npr. ABAC).
- **Single-Sign On (SSO)**
- **Posredovanje identitete in prijave z družabnimi omrežji:** Omogočanje prijave z družabnimi omrežji je enostavno dodati prek upraviteljske konzole.

- **Federacija uporabnikov:** Keycloak ima vgrajeno podporo za povezavo z obstoječimi strežniki LDAP ali Active Directory. Če se uporabniki shranjujejo v drugih shrambah, npr. v relacijski podatkovni zbirki, je možna tudi implementacija lastnega ponudnika.

Na spodnji sliki je predstavljen visokonivojski pogled na arhitekturo Keycloak-a:



Integracije Keycloak-a

Kong

Na podlagi pregleda seznama Kong vtičnikov (<https://docs.konghq.com/hub/>) sta za integracijo s Keycloakom najprimernejša vtičnika **OpenID Connect** (<https://docs.konghq.com/hub/kong-inc/openid-connect/>) in **OAuth 2.0 Introspection** (<https://docs.konghq.com/hub/kong-inc/oauth2-introspection/>), ki pa sta oba na voljo le v plačljivi različici Kong-a.

Mosquitto

Za preverjanje pristnosti je na voljo vtičnik **Mosquitto Go Auth** (<https://github.com/iegomez/mosquitto-go-auth>), ki ponuja več različnih možnosti avtentikacije. Za OAuth 2.0 sicer ni na voljo direktne integracije z rešitvami IAM, se pa lahko implementira vtičnik po meri. Za primer se lahko vzame <https://github.com/gewv-tu-dresden/mosquitto-go-auth-oauth2> .

Chirpstack

Chirpstack ponuja integracijo z OpenID ponudniki, med katere se šteje tudi Keycloak.

Na spodnji sliki je prikazan izsek iz konfiguracije Chirpstack-a, na podlagi katere je omogočena integracija s ponudniki OIDC.

```
# User authentication configuration.
[user_authentication]

# OpenID Connect.
[user_authentication.openid_connect]

# Enable OpenID Connect authentication.
#
# Enabling this option replaces password authentication.
enabled=false

# Registration enabled.
#
# Enabling this will automatically register the user when it is not yet present
# in the ChirpStack database. There is no registration form as the user information
# is automatically received using the OpenID Connect provided information.
# The user will not be associated with any organization, but in order to
# facilitate the automatic onboarding of users, it is possible to configure a
# registration callback URL (next config option).
registration_enabled=false

# Registration callback URL.
#
# This (optional) endpoint will be called on the registration of the user and
# can implement the association of the user with an organization, create a new
# organization, ...
# ChirpStack will make a HTTP POST call to this endpoint,
# with the following URL parameters:
# - user_id, of the newly created user in ChirpStack.
#
# The POST body contains a JSON payload with the OpenID Connect UserInfo payload.
registration_callback_url=""

# Provider URL.
# This is the URL of the OpenID Connect provider.
# Example: https://auth.example.org
provider_url=""

# Client ID.
client_id=""

# Client secret.
client_secret=""

# Redirect URL.
#
# This must contain the ChirpStack Application Server web-interface hostname
# with '/auth/oidc/callback' path, e.g. https://example.com/auth/oidc/callback.
redirect_url=""

# Logout URL.
#
# When set, ChirpStack Application Server will redirect to this URL instead
# of redirecting to the login page.
logout_url=""

# Login label.
#
# The login label is used in the web-interface login form.
login_label=""
```

Grafana

Grafano je moč integrirati s Keycloak-om s pomočjo t.i. **Keycloak OAuth2** avtentikacije. Primer je opisan tukaj: <https://grafana.com/docs/grafana/latest/setup-grafana/configure-security/configure-authentication/keycloak/> .

ThingsBoard

Thingsboard ponuja podporo za integracijo na podlagi protokola OAuth 2.0: <https://thingsboard.io/docs/user-guide/oauth-2-support/> .

5. Namestitev v okolju Kubernetes (K8s)

5.1 Uvod v Kubernetes

Kubernetes je postala de facto standardna platforma za avtomatizirano nameščanje, skaliranje in upravljanje kontejneriziranih aplikacij.

5.2 Možnosti namestitev s pomočjo K8s

Kubernetes se lahko postavi oz. uporablja kot:

- upravljana postavitve ("managed") ali lastna postavitve ("self-managed");
- nameščena v oblaku (privatni, javni, hibridni), ali na lastni infrastrukturi ("on-premise")

Najlažje je samo uporabiti upravljano postavitev v oblaku, ker se tu celotno breme upravljanja in nadgrajevanja prenese na ponudnika. Najbolj fleksibilna pa je lastna postavitev na svoji infrastrukturi, saj se lahko popolnoma prilagodi in optimizira za točno določeno aplikacijo+uporabo.

- Najbolj pogoste distribucije so RH/IBM OpenShift, AWS EKS, Azure AKS, Google GKE, VMware Tanzu, Rancher (Suse) RKE2.
- Certified K8s: v poštveh naj pridejo izključno certificirane K8s distribucije
- Kubernetes in veliko najpomembnejših "cloud-native" projektov upravlja CNCF (Cloud-Native Computing Foundation, hčerinska fundacija Linux Foundation). CNCF definira več tipov projektov, glede na njihovo zrelost in celovitost/velikost podpore (Graduated, Incubating, Sandbox). Aplikacije nameščamo na več načinov, največkrat je to Helm (K8s package manager). Za bolj zapletene in kompleksne postopke nameščanja in upravljanja posameznih aplikacij ali podpornih storitev se uporabljajo t.i. operatorji (Operators), ki avtomatizirajo potrebno domensko znanje (recimo operator za namestitve in upravljanje z visoko razpoložljivo relacijsko bazo Postgres, MongoDB, itd.)
- Kubernetes ne definira standardnega omrežnega vtičnika (networking plugin). Najpogosteje uporabljeni odprtokodni projekti so npr. Flannel, Calico, Cilium. Izbor je odvisen tudi od značilnosti in potreb omrežja, v katerem bo nameščena sama K8s gruča.

5.3 Izzivi pri uporabi Kubernetesa

Kubernetes je primarno načrtovan za podporo storitvam brez ohranjanja stanja (stateless services). Za podporo vsem tipom storitev je potrebno rešiti v večini primerov naslednje potencialne izzive:

- shranjevanje podatkov za same storitve (stateful services)
 - SQL, noSQL
Fiware uporablja tako klasične relacijske baze (recimo Postgres), kot noSQL baze, kot je npr MongoDB. Oba tipa baz je potrebno namestiti visoko razpoložljivo in skalabilno, glede na predvidene potrebe aplikacij.
 - tipi shranjevanja - block, file, object storage (container native vs legacy; distributed, centralised): vse tri običajne tipe shranjevanja podatkov je možno implementirati tako v novejših, "container-native" rešitvah, kot tudi uporabiti že obstoječe naročnikove instance baz (priporočljiva opcija). Če je le možno, je zaradi že vpeljanih postopkov upravljanja, nadgradenj in varnostnega kopiranja vsaj v prvi fazi projekta najbolj optimalno uporabiti že obstoječe, tipično centralizirane rešitve. Baze, ki so nameščene v samem Kubernetes clustru, je namreč zaradi obilice potrebnega novega znanja, ravno v tradicionalnih okoljih težko upravljati.
- posodabljanje/upravljanje platforme; "Day 2 ops": sama namestitve in vzpostavitev Kubernetes platforme je samo prvi korak ("Day 1"), precej večji izziv je ravno posodabljanje in učinkovito upravljanje platforme na daljši rok. Med pilotom je

potrebno pridobiti izkušnje in vzpostaviti lastno ekipo (ali pridobiti ustrezne zunanje izvajalce), ki bodo po pilotu vzdrževali, nadgrajevali in upravljali s platformo. Kubernetes je še vedno platforma, ki se hitro spreminja (3 nove verzije na leto, Kubernetes skupnost razvijalcev zagotavlja združljivost za nazaj za (vsaj) 3 verzije) in je zato nujno, da se redno nadgrajuje.

- HA in skalabilnost; smiselna definicija za SLO in SLI (Service Level Objective/Indicator): pri definiranju sprejemljivih SLO in SLI metrik je nujno potrebno dobro premisliti in jih definirati glede na realne potrebe, saj v nasprotnem primeru produkcijskem sistemu, ki se bo vzpostavil po uspešnem pilotu, pretirano zaostreni SLO/SLI in HA prej škodijo kot koristijo. Pretirane zahteve (oz. nerealne glede na resnične potrebe) lahko ali močno podražijo projekt in/ali dodatno zakomplicirajo arhitekturo.

5.4 Kubernetes Fiware Foundation referenčna postavitve

Fiware foundation ima objavljeno referenčno postavitve nekaterih Fiware komponent (<https://github.com/FIWARE-Ops/fiware-gitops>). Za vzpostavitev in namestitve uporabljajo naslednje koncepte in produkte:

- GitOps: namestitvene deklaracije komponent se hranijo v Git. Tu se uporabljajo enaki koncepti, kot se za spreminjanje in upravljanje z verzijami kode pri pisanju aplikacij (git kot repozitorij, veje, Pull Requests, itd.). Sam koncept ni vezan na Kubernetes, v praksi pa se uporablja večinoma ravno v tem okolju. V splošnem se pri GitOps uporabljata dva koncepta: "Pull-based" in "Push-based".
 - ArgoCD (<https://github.com/argoproj/argo-cd/>) je poleg FluxCD najbolj razširjena uporabljana odprtokodna platforma za GitOps. ArgoCD ima tudi grafični vmesnik, ki omogoča pregled namestitve aplikacij v različnih Kubernetes gručah.
- Kubernetes distribucija: uporabljajo RedHat (IBM) OpenShift, ki je certificirana Kubernetes distribucija, v katero je proizvajalec dodal svoje nestandardne razširitve.
- Uporabljajo veliko dobrih praks in integracij; ampak ker so narejene z OpenShift razširitvami, jih bo potrebno adaptirati za standardno K8s distribucijo.

5.5 Načrt namestitve pilotne platforme

- V prvi fazi postavitve v Docker-ju brez K8s, z uporabo Docker Compose.
- Na podlagi funkcionalnega testiranja prve faze se bodo pripravili recepti, postopki in optimalne privzete vrednosti parametrov za namestitve izbranih komponent v okolju Kubernetes. Namestitve bomo ciljno poskušali narediti za standardni certificirani Kubernetes, saj OpenShift podpira tudi takšne namestitvene definicije, obratno pa ne velja (torej OpenShift specifičnih deklaracij namestitve same IoT platforme ni možno namestiti brez sprememb na standarden Kubernetes). Tako pripravljene standardne namestitve so zato bolj fleksibilne kot tiste, ki uporabljajo razširitve OpenShift. Ker OpenShift podpira dodatne funkcionalnosti, ki jih standardni Kubernetes nima, včasih ni možno vsega lepo prenesti, oz. zahteva takšen prenos specifične komponente. Vsa takšna morebitna odstopanja bomo dokumentirali in eksplicitno našli, skupaj z možnimi rešitvami.

