TARTU ÜLIKOOL
Arvutiteaduste instituut
Informaatika õppekava

Märten Kahu

Lahendus avaandmete jagamiseks Cumulocity IoT platvormilt

Bakalaureusetöö (9 EAP)

Juhendaja: Pelle Jakovits, PhD

Tartu 2023

Lahendus avaandmete jagamiseks Cumulocity IoT platvormilt

Lühikokkuvõte:

Cumulocity on IoT seadmete ning nende tekitatud mõõtetulemuste hoidmiseks mõeldud platvorm. TÜ Delta keskus hoiab platvormil oma majaautomaatika andurite poolt tekitatud andmeid. Hetkel ei ole mugav neid andmeid kõigile soovijatele avalikustada, kuna Cumulocity platvormile ligipääsemiseks on vaja eraldi kasutajat ning platvormi haldusrakendus ei ole mõeldud oma kasutajamugavuse poolest tavakasutajatele. Töö eesmärk on luua lahendus, mille abil on võimalik andurite mõõtetulemusi soovitud ajaperioodist alla laadida ning neid avalikustada nii lahenduse siseselt kui ka välisesse andmeportaali. Töö teoreetilises osas kirjeldatakse Cumulocity platvormi, selle rakendusliidest TÜ Delta keskuse Cumulocity andmemudelit. Samuti kirjeldatakse lahenduse arhitektuuri, tehnoloogilisi valikuid ning valmivat lahendust ennast.

Võtmesõnad:

Cumulocity IoT platvorm, värkvõrk, Node.js, TypeScript

CERCS:

P175 Informaatika

Solution for sharing open data from Cumulocity Iot platform

Abstract:

Cumulocity is an IoT data platform. UT Delta centre uses it to store its building automation related sensor data. Currently there isn't an easy and convenient way to share that data to all users as a separate Cumulocity account is needed and Cumulocity admin portal is not really meant to be used by ordinary users just interested in data. The goal of this thesis is to develop a software solution that can be used to download sensor data from chosen time period, publish it locally to all system users and export it to an external data portal. In the theoretical part of the thesis an overview is given of the Cumulocity platform, its API and Cumulocity data model of UT Delta centre. Overviews about the architecture, technological choices and the solution itself are also given.

Keywords:

Cumulocity IoT platform, Internet of Things, IoT, Node.js, TypeScript

CERCS:

P175 Informatics

Sisukord

Sissejuhatus **5**
1.Tausta ülevaade **8**
1.1 Värkvõrk 8
1.2 Cumulocity IoT platvorm 9
1.2.1 Üldine ülevaade 9
1.2.2 Cumulocity API 11
1.2.3 TÜ Delta keskuse andmemudel 11
1.3 Andmeportaalide olulised aspektid 13
1.3.1 Avaandmeportaalide ülevaade 15
1.3.2 CKAN 16
1.3.3 Olemasolev lahendus 17
2. Loodava lahenduse nõuded ja kasutatud tehnoloogiad **18**
2.1 Probleemi püstitus 18
2.2 Lahenduse nõuded 18
2.2.1 Funktsionaalsed nõuded 18
2.2.2 Mittefunktsionaalsed nõuded 19
2.3 Kasutatud tehnoloogiad 20
2.3.1 Arenduskeel 20
2.3.2 Serveripoolne raamistik 20
2.3.3 Andmebaas 21
2.3.4 Suhtlus komponentide vahel 21
3. Arhitektuur ning tööprotsess **24**
3.1 Süsteemi üldkirjeldus 24
3.2 Arhitektuur 24
3.3 Taustatööd 26
3.4 Failide alla ning üleslaadimine 28
4. Valminud lahendus **30**
4.1 Lahenduse ülevaade 30
4.2 Võimekuste kirjeldus 35
4.3 Valideerimine ning järgnevad sammud 36
4.3.1 Valideerimine 36
4.3.2 Järgnevad sammud 38
Kokkuvõte **40**
Viidatud kirjandus **41**
Lisad **43**
Lisa 1. Andmete liikumine TÜ Delta keskuses, seminar “TÜ Targa maja lahendused 43
Lisa 2. Delta keskuse andmepunktide avastamine Cumulocity kokpitis 44
Lisa 3. Haldusrakenduse avaleht 45
Lisa 4. Lähtekood 46

Litsents **47**

Sissejuhatus

Tänapäeval kogutakse andmeid kõige ja kõigi kohta. Andmed tekivad igapäevase digivahendite kasutamise käigus, neid kogutakse sihipäraselt avalik-õiguslikes ja teadusasutustes. 2000ndate alguses lisandus täiesti uus viis andmete kogumiseks. Sel ajaperioodil hakati arendama asjade internetti, mis tähendas, et objektid (nt andurid) hakkasid koguma infot ümbritseva keskkonna kohta ning saatma seda üle interneti andmebaasidesse. Tänu uudsete seadmete kasutuselevõtule tekkis võimalus koguda täpseid andmeid erinevate protsesside, näiteks ilma ning liikluse kohta [1]. Seega lisandusid inimeste loodud ja kogutud andmetele tarkade seadmetega kogutud andmed [1].

Nii inimeste loodud kui ka tarkade seadmetega kogutud andmete hulk suureneb pidevalt, kuid samas paraneb ka oskus suuri andmehulki töödelda, et neis olevat infot üles leida. Seepärast nähakse andmetes suurt väärtust. 2003. aastal kehtestati Euroopa Liidus esimest korda kohustus [2] teha avaliku sektori valduses olev teave kõigile kättesaadavaks, et igaüks saaks kasutada andmeid talle sobival moel ja luua saadud teabe põhjal lisaväärtust, mis majanduse ja ühiskonna arengut soodustaks.

Eelmises lõigus nimetatud ja aja jooksul täiendatud eeskirjast lähtudes avalikustatakse ka Eestis avalikus sektoris kogutud andmeid ehk avaandmeid. Üldjoontes saab avaandmed jagada kaheks: inimeste kogutud ja suure intervalliga uuenevad andmed ning tarkade seadmete vahendusel kogunevad andmed, mille puhul lisandub uus mõõtetulemus näiteks iga poole minuti tagant. Esimest tüüpi andmetele on loodud ligipääs Eesti avaandmete teabevärava kaudu. Asjade interneti andmeid hoitakse spetsiaalsetel IoT (värkvõrgu) platvormidel, näiteks Cumulocityl. Seda platvormi kasutavad näiteks Tartu Tark linn ja TÜ Delta keskus oma tarkade andurite andmete hoidmiseks, kuid need andmed ei ole praegu veel kõigile vabalt kättesaadavad.

Cumulocity IoT platvormil saab hallata värkvõrgu seadmeid, luua kasutaja defineeritud reegleid ning visualiseerida andurite genereeritud andmeid Cumulocity veebipõhise haldusrakenduse ehk kokpiti kaudu. Nendest võimalustest hoolimata on Cumulocity IoT platvormi kasutamisega seotud mõned probleemid. Esiteks on kogu platvormile piiratud ligipääs, kuna vajalik on Cumulocity konto olemasolu. Samuti ei ole andmete otse allalaadimine kasutaja jaoks mugav, sest kokpiti rakendus on mõeldud rohkem IT-teadlikule

kasutajale. Cumulocity veebipõhine haldusrakenduse ülesehitus on keeruline ning see ei ole mõeldud tavakasutajatega andmete jagamiseks. Seal ei ole lihtne aru saada, millised andurid ning andmetüübid on platvormil üldse olemas. Näiteks on andmemudel jagatud erinevate objektide alla, kusjuures ühel objektil võib olla alamobjekte ning samuti võivad andurite andmepunktid konkreetsel objektil puududa [3]. TÜ Delta keskus kasutab Cumulocity platvormi vaid värkvõrgu andmete integratsiooniks. Seadmete haldamist platvormi kaudu hetkel ei toimu. Samuti on Cumulocity platvormil oleva Delta keskuse andmekogu andmepunktide sirvimine keeruline. Seetõttu ei ole kerge näha, milliseid andmeid on platvormilt võimalik alla laadida.

Töö eesmärk on luua Cumulocity IoT andmeportaalil põhinev prototüüplahendus, mille abil oleks võimalik eksportida ning avalikustada platvormil olevate andurite andmeid kõigile kasutamiseks. Selleks luuakse süsteem, mis võimaldab näha, millised andurid on süsteemis olemas, defineerida vajadusel nende kohta metaandmed ning laadida alla valitud anduri andmed soovitud ajaperioodi kohta. Samuti luuakse võimalus vaadata juba avalikustatud andmeid eesmärgiga vähendada üleliigset andmete allalaadimist Cumulocity platvormilt. Süsteem tehakse vabalt kättesaadavaks Github'i koodirepositooriumi vahendusel. Prototüüplahenduse loomisel kasutatakse TÜ Delta keskuse andmeid, kuid tulevikus on soov kasutada seda ka Tartu Linnavalitsuse andmetel.

IoT (*Internet of Things*) eestikeelseteks vasteteks on AKIT-is (Andmekaitse ja infoturbe leksikonis) *esemevõrk* ja *värkvõrk*, Sõnaveebis *värkvõrk*, *asjade internet* ja *nutistu*. Selles töös kasutatakse IoT kohta vaheldumisi sõnu *asjade internet* ja *värkvõrk*.

Bakalaureusetöö koosneb neljast peatükist. Esimeses peatükis antakse ülevaade värkvõrgust, Cumulocity IoT platvormist, TÜ Delta Keskuse andmemudelist, andmeportaali tähtsamatest komponentidest, avaandmete andmeportaalidest ning olemasolevast Pythoni põhjal loodud lahendusest. Teises peatükis analüüsitakse loodava lahenduse funktsionaalseid ja mittefunktsionaalseid nõudeid, kirjeldatakse lahenduses kasutatud tehnoloogiaid ning põhjendatakse nende valikut. Kolmandas peatükis räägitakse arendusprotsessist ning lahenduse arhitektuurist. Neljandas peatükis antakse ülevaade valminud lahenduse kolmest mikroteenusest. Seejärel kirjeldatakse, mida on võimalik praeguseks hetkeks valminud lahendusega teha. Käsitletakse ka lahenduse valideerimist ja võimalusi lahenduse edasi arendamiseks.

Bakalaureustöö juures on neli lisa. Lisa 1 näitlikustab TÜ Delta keskuse andmete liikumist. Lisa 2 on kuvatõmmis TÜ Delta keskuse andmepunktide avastamisest Cumulocity IoT kokpitis. Lisa 3 näitab, milline näeb välja loodava haldusrakenduse avaleht. Lisa 4 kirjeldab loodava lahenduse lähtekoodi asukohta.

1.Tausta ülevaade

Peatüki esimeses osas käsitletakse kõigepealt värkvõrku. Seejärel antakse ülevaade Cumulocity IoT platvormist, mis on üks värkvõrgu platvormidest, kus saab hoida tarkadest seadmetest kogunenud andmeid. Samuti selgitatakse lühidalt andmetega suhtlemiseks kasutatava kasutajaliidese ehk Cumulocity API olemust. Peatüki esimese osa lõpus kirjeldatakse TÜ Delta keskuse andmemudelit.

Peatüki teises osas antakse ülevaade avaandmete andmeportaali olulisematest aspektidest. Samuti vaadeldakse olemasolevaid andmeportaale eesmärgiga, kas vastavad portaalid sobiksid Delta keskuse värkvõrgu andmete avalikustamise jaoks. Seejärel tutvustatakse Pythoni programmeerimiskeeles varem loodud rakendusteeki andmete pärimiseks Cumulocity platvormilt.

1.1 Värkvõrk

Hanes jt [1] osutab, et värkvõrgu ehk asjade interneti mõistele pani 1990ndate lõpus aluse Kevin Ashton (*Auto-ID Center at MIT* kaasasutaja), kui ta kirjeldas selle terminiga tarneahelate ühendamist internetiga. Hiljem on Ashton veel öelnud, et värkvõrgu seadmed on kui tajudega arvutid, mis suudavad iseseisvalt vastu võtta välismaailmast tulevaid signaale. Veel 20. sajandil arvutid seda ei osanud ning teadsid vaid seda, mida inimesed olid neile öelnud. Juba 2000ndate lõpuks oli maailmas internetti ühendatud rohkem värkvõrgu seadmeid, kui oli maailmas inimesi [1].

Värkvõrgu põhiline eesmärk on internetti ühendada sinna veel mitte ühendatud füüsilised seadmed, et teha nad “targaks” ehk võimeliseks interakteeruma inimestega ning teiste võrku ühendatud seadmetega. Üks värkvõrgu oluline mõiste on andur. David Hanes jt [1] kirjutab, et andur mõõdab füüsikalist suurust (temperatuur, rõhk, niiskus, deformatsioon jne) või keemilist suurust (süsihappegaas jne) ja teisendab mõõtetulemuse elektrisignaalidena digitaalkujule, mida saab talletada andmehulgana ja töödelda arvutites. Anduritega saab koguda andmeid nii objektide kui keskkonna kohta, et teada, milline on mõõdetava suuruse hetkeolukord ning jälgida, kuidas füüsikaline või keemiline suurus ajaperioodis muutub [1].

Samuti aitab andurite ja teiste tarkade seadmete internetti ühendamine luua tihedama interaktsiooni füüsilise ning digitaalse maailma vahel ning loob eeldused efektiivsuse kasvuks ning uute võimekuste loomiseks väga paljudel erinevatel aladel alates targa maja tehnoloogiatest ja tööstusautomaatikast kuni inimese tervise juhtimiseni. Ühe näitena saab välja tuua transpordivaldkonna - tänapäeva autodega on ühendatud palju erinevaid värkvõrgu seadmeid, mis mõõdavad kütuse kasutust, mootori pöördeid, rataste asendit jm energiakasutuse optimeerimiseks. Logistikafirmad kasutavad GPS-seadmeid oma veokite asukoha jälgimiseks, selle abil saab ennustada reaalajas veoki jõudmist sihtpunkti või optimeerida marsruuti vastavalt kohalikele teeoludele [1]. Värkvõrgu ehk IoT võimalusi kasutatakse ka hoonete haldamisel näiteks kliimaseadmete, valgustuse ning kütte kasutuse optimeerimiseks.

Värkvõrkude tarkadest seadmetest koguneb pidevalt uusi andmeid. Sellist tüüpi andmete hoidmiseks ja töötlemiseks on loodud palju platvorme, Cumulocity IoT on neist üks. Järgnevalt räägitakse lähemalt selle platvormi omadustest.

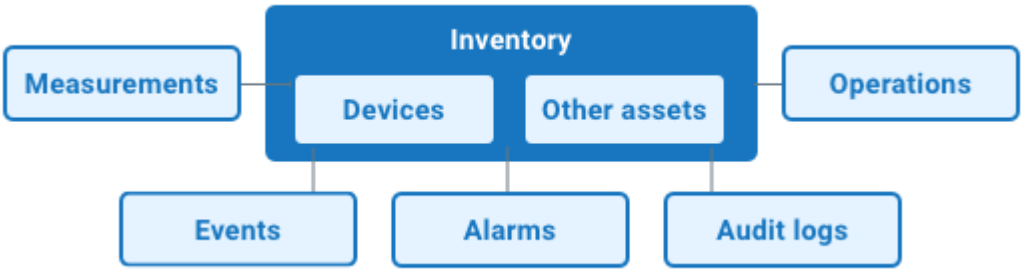
1.2 Cumulocity IoT platvorm

1.2.1 Üldine ülevaade

Cumulocity IoT on peamiselt värkvõrgu seadmete haldamiseks mõeldud platvorm, mida pakutakse platvorm teenusena (*PaaS*). Pärast värkvõrgu seadmete ühendamist Cumulocity platvormiga näiteks BACnet [4] protokolli vahendusel on neid tarku seadmeid võimalik kaugjuhtimise teel vastavalt konkreetse seadme võimalustele hallata ning monitoorida, näiteks saab ühenduse kvaliteeti jälgida ning seadet taaskäivitada. Seadmete küljes olevate andurite kogutud andmeid on võimalik näha Cumulocity Cockpit rakendusest. Lisaks on kasutajal võimalik luua oma ärireegleid, näiteks e-kirja saatmine teatud reaalajaliste sündmuste toimumisel. Kõiki Cumulocity pakutavaid võimekusi saab kasutada ka rakendusliidese (API) vahendusel. See on sobilik olukordades, kus on soov luua oma erilahendusi Cumulocity platvormi põhjal [3].

Cumulocityl on olemas standardne andmemudel platvormil olevate objektide jaoks (vt joonis 1). Objektide põhiinfo ning seosed teiste objektidega salvestatakse inventari (*Inventory*). Inventaris olevate objektidega on seotud erinevad lisaoperatsioonid, mis tagastavad valitud

objektiga seonduvaid andmeid. Nendeks on mõõtmised (*Measurements*) ja operatsioonid (*Operations*) ning sündmused (*Events*) koos alarmide (*Alarms*) ja turvalogidega (Audit logs).



Joonis 1. Cumulocity andmemudeli ülevaade [5]

Ühte inventaris olevat asja nimetatakse hallatud objektiks (*Managed Object*). See võib olla mõni füüsiline objekt, näiteks temperatuuriandur või GPS-seade. Samuti võib see tähistada loogilist alamobjektide rühma, näiteks ruumi, mille sees on mitu erinevat füüsilist mõõteandurit [5]. Joonisel 2 on näha ühe manageeritud objekti näidis.

```
{
    "id": "12345",
    "name": "Smart switch",
    "type": "ge_45609",
    "c8y_Relay": {
        "state" : "OPEN"
    }
}
```

Joonis 2. Üks manageeritud objekt [5]

Üldjuhul on igal manageeritud objektil unikaalne ID-kood, nimi, tüüp, viimati uuendamise aeg ning seda objekti iseloomustavad fragmendid. Objekti fragmendid on atribuudid, kuhu on võimalik salvestada objekti kohta metaandmeid. Fragmentide soovituslik nimetamise stiil on *c8y_<fragmendi_nimi>*. Nii moodustub üksteisele viitavate manageeritud objektide vahel hierarhiline struktuur, kus ühe objekti põhiandmed on mitmetasandiliselt ühes kohas koos.

Kui kasutajal on vaja integreerida Cumulocity funktsionaalsus oma lahendustesse, saab ta seda teha Cumulocity rakendusliidesega (Cumulocity API). Samuti kasutatakse seda kasutajaliidest TÜ Delta keskuse värkvõrgu andmete platvormile saatmiseks.

1.2.2 Cumulocity API

Cumulocity IoT platvormi API kirjeldus kasutab OpenAPI spetsifikatsiooni ning on loodud kasutades REST (*representational state transfer*) metoodikat ning HTTP-d (*Hypertext Transfer Protocol*). Tagastatavad andmed on vaikimisi JSON (*JavaScript Object Notation*) formaadis. Liidese kasutamiseks peab kasutaja ennast autentima, mida on võimalik teha kasutades *Basic, OAI-Secure* (soovitatud), *SSO* või *JWT* (aegunud ning pole enam soovituslik kasutada) metoodikaid [6]. API funktsionaalsust on Cumulocitys kirjeldatud põhjalikult, seetõttu on selle kasutamine mugav ning lihtne.

Cumulocity API-l on palju funktsionaalsusi ning objekte saab platvormil nii luua kui ka pärida. API kaudu on võimalik hallata platvormil olevaid kasutajaid (*Users*) ning üürnikke (*Tenants*). Üürnik on eraldiseisev andmeruum, milles olevad kasutajad ning ühendatud seadmed on vaikimisi täielikult eraldatud teistest üürnikest. Samuti on funktsionaalsused alarmide ja sündmuste haldamiseks ning operatsioonide tegemiseks [6]. Üks operatsioon on näiteks seadme taaskäivitamine.

Cumulocity API-t kasutatakse loodavas lahenduses tagasüsteemide ning Cumulocity vaheliseks suhtluseks. Käesoleva töö kontekstis on olulised funktsionaalsused manageeritud objektide, objektide andmefragmentide ning mõõtetulemuste pärimised. Neid funktsionaalsusi kasutatakse töö järgnevates etappides loodavas lahenduses. Lahendust luues kasutatakse TÜ Delta keskuse andmeid. Järgnevas alajaotises räägitakse selle keskuse värkvõrgu andmemudelist lähemalt.

1.2.3 TÜ Delta keskuse andmemudel

TÜ Delta keskuse Cumulocity platvormi kirjeldav alampeatükk põhineb Tartu Ülikooli arvutiteaduse instituudi dr Pelle Jakovitsiga toimunud intervjuul.

Delta keskuse ruumiautomaatika andmete hoidmiseks valiti Cumulocity IoT platvorm, mis võeti kasutusele koostöös Teliaga. Cumulocity platvormi kasutab ka Tartu linn oma targa linna andmete hoidmiseks. Platvorm töötab TÜ HPC keskuses (Teadusarvutuste keskus) OpenStack pilve operatsioonisüsteemil põhinevas pilves. 2023. aasta aprillis oli Delta keskuse värkvõrgu andmete maht 7-30 GB, sõltuvalt sellest, kas vaadata andmeid toorkujul või JSON-formaadis. Keskusel on üle 500 hoone keskkonna ning automaatikaga seotud

anduri, mis koguvad andmeid põhiliselt temperatuuride, süsihappegaasi kontsentratsioonide ning päikeseenergia tootlikkuse kohta.

Keskuse andurite ja Cumulocity vaheliseks suhtlemiseks kasutatakse BACnet (*Building Automation and Control Network*) [4] kommunikatsiooni standardit. Selle abil kuulab Pythoni BACneti klient maja automaatikasüsteemi kaudu andurite väärtuste muutumise sündmuseid ning saadab nende väärtused MQTT sõnumivahetuse teenust kasutades Apache Nifi teenusele. Apache Nifi on andmete edastamiseks ning teisendamiseks mõeldud rakendus, mis muudab andmed sobivasse formaati ning edastab need Cumulocity platvormile, kust neid saab Cumulocity API-ga alla laadida. Samuti edastatakse andmed influxDB [7] andmebaasi, kus luuakse saadud informatsioonist erinevaid visualiseeringuid, et näiteks päikeseenergia tootmisest paremini aru saada. Delta keskuse andmete liikumine on graafiliselt kirjeldatud lisas 1.

Kuigi Cumulocity platvormil on palju võimalusi, siis hetkel kasutatakse Delta keskuses seda platvormi vaid andmete kogumiseks ehk andurid on ühendatud passiivselt ning seadmete taaskäivitamist platvormi kaudu ei toetata. Samuti on ruumiautomaatika andmete kogumise eesmärk lisaks ruumide monitoorimisele eelkõige õppeotstarbeline.

Kõik Delta keskuses olevad andurid ei ole veel Cumulocity süsteemiga ühendatud. Praegu võib platvormilt leida temperatuuri, süsihappegaasi, päikeseenergia, ventilatsiooni ning muid andmeid. Ajaliselt on andmeid 2,5 aastasest perioodist, erinevad andurid tagastavad andmeid erineva intervalliga, seega andmeridade tihedus varieerub seadmelt seadmele.

Delta keskuse anduritega on paar probleemi. Üks neist on seotud andurite nimedega. Kuna nimi (nt AK01'Vent'SV2'PreHcl või AK01'Mbus'Elarv'Mtr114) ei anna täpselt edasi anduri sisu ning objektide hierarhia peegeldab ruumiautomaatika juhtseadmete hierarhiat, siis praegu pole Cumulocity kokpiti rakenduses (veebipõhine seadmete haldusrakendus) võimalik aru saada, mis seadmega on tegemist, kus see asub või millist tüüpi andmeid sealt on võimalik kätte saada. Lisas 2 on näidatud Cumulocity kokpiti rakenduses andmepunktide valimine visualiseerimiseks. Loodavas lahenduses plaanitakse objektide hierarhia esitada ühetasandilisena, kus oleks kohe näha objekti andmetüüp.

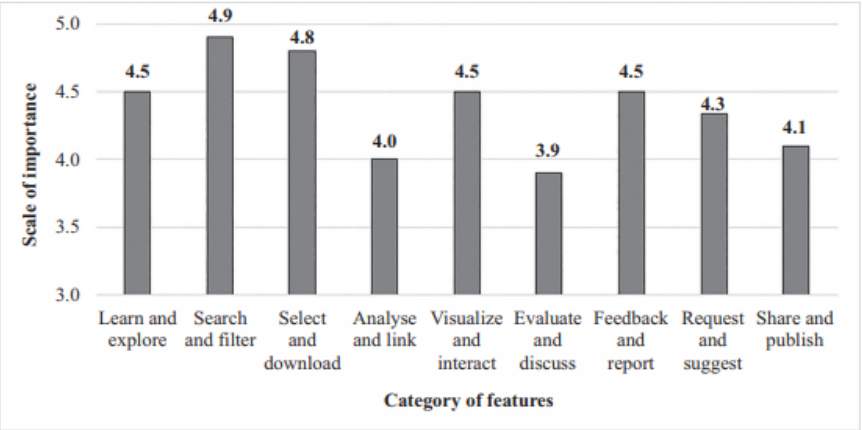
1.3 Andmeportaalide olulised aspektid

Nagu eelmise alajaotise lõpus osutati, siis andmete puhul on olulised paljud tegurid alates andmeallikate nimedest kuni nende esitusviisini. Lnenicka [8] viitab, et kuna avaliku sektori andmed ehk avaandmed, sh värkvõrkude omad, muutuvad tänapäeval üha mitmekesisemaks ning et nende hulk kasvab, siis järjest olulisemaks muutub ka andmete analüüsimine, sest saadud info põhjal on võimalik optimeerida tegevusi ja võtta vastu otsuseid [8]. Seetõttu võiks tähelepanu pöörata, mis on andmete töötlemisel oluline.

Andmete kasutamist soodustavad metaandmed, andmete kättesaadavus, tähtsamate võimaluste olemasolu ja kasutusmugavus. Lnenicka [8] sõnul tähendab kvaliteetne andmebaas seda, et algandmetele on lisatud ka metaandmed ehk andmed andmete enda kohta, näiteks nende loomise geograafiline asukoht, lühikirjeldus või muu, mis on oluline. Metaandmete ülesanne on kirjeldada andmeid tuleviku tarbeks, kuna andmebaaside praegused loojad ei tea ette kõiki võimalikke kasutusvorme, mis tulevikus võivad tekkida [8].

Avalikud andmed peavad olema ka kättesaadavad. Lnenicka [8] toob välja, et üks avaandmete põhiprobleeme on see, et soovitud andmestikke on raske üles leida. Tavalised otsingumootorid tema sõnul selleks ei sobi, kuna need on suunatud otsima terminite, mitte metaandmete põhjal. Seetõttu on riigid ise loonud avaandmete portaale, mis oskavad otsimisel arvestada ka metaandmetega ning kuhu saab andmekogusid või otseteid andmekogude juurde üles laadida.

Selleks et andmeportaal oleks kasutajate jaoks efektiivne, peavad sellel olema teatud tunnused. Lnenicka jt [9] on uurinud oma töös, millised tunnused on andmeportaalidele kõige omasemad ning kui olulised need ekspertide hinnangul on. Ühtekokku 82 ingliskeelsest portaalist valiti välja edasiseks uurimiseks 48 tunnust. Leitud tunnused jagati Delphi protsessi kasutades üheksaks kategooriaks (joonis 3).



Joonis 3. Tunnuste kategooriad [9: 1028]

Leiti, et kõige olulisemad kategooriad olid Otsi ja filtreeri (*Search and filter*) ning Vali ja laadi alla (*Select and download*). Nendele järgnesid Õpi ja uuri (*Learn and explore*), Visualiseeri ja interakteeru (*Visualize and interact*) ning Tagasiside ja teata probleemist (*Feedback and report*).

Portaali ülesehitus peaks olema võimalikult mugav, et soodustada selle kasutamist. Samuti osutab Lnenicka jt [9], et portaali ülesehituses peaks arvestama erineva oskustasemega kasutajatega. Näiteks kirjeldati, et kui algajatele ei pruugi tagasisidestamine olla oluline, siis kogenenumatele kasutajatele on see vajalik, kuna nad on tavaliselt teemast rohkem huvitatud, oskavad luua materjalide põhjal paremaid uurimisküsimusi ning soovivad oma tulemusi ka avaldada. Seetõttu on nad huvitatud sellest, et algallikad oleksid võimalikult täpsed. Lisaks on oluline välja tuua, et algajate kasutajate jaoks olid kõige olulisemad kategooriad Õpi ja uuri ning Otsi ja filtreeri. Nende kategooriate võimalused peaksid olema portaali kodulehel välja toodud, et neist oleks portaali kõigile kasutajatele maksimaalselt kasu [9].

Käesoleva töö kontekstis ei kasutata esialgu kategooriaid Hinda ja arutle (*Evaluate and discuss*), Visualiseeri ja interakteeru (*Visualize and interact*) ning Analüüsi ja lingi (*Analyze and link*). Need jäävad käesoleva tööst ajapuuduse tõttu välja. Käesoleva töö raames luuakse võimekus põhiliselt kategooriatele Otsi ja filtreeri, Vali ning laadi alla ning Jaga ja avalda (*Share and publish*). Samuti luuakse võimekus lisada metaandmeid anduri külge.

1.3.1 Avaandmeportaalide ülevaade

Eesti avaandmete teabevärav

Üks võimalus, kuidas avaandmete ideest aru saada, on vaadata seni loodud andmekogusid. Näiteks Eesti avaandmete teabevärav on andmekogu, mis on mõeldud Eestiga seotud ja kõigile vabalt kättesaadavate andmete jagamiseks. Selles leidub linke avaliku sektori andmetele ning era- ja kolmanda sektori poolt jagatud litsentsitud andmetele, näiteks "Sõidukite staatused Eestis", "Vääriselupaiga tunnistusega spetsialistid" jne. 2022. aasta detsembris oli portaalis ära toodud 2220 teabevaldajat ja 1572 andmestikku energeetika, tervise, keskkonna ning paljude muude valdkondade kohta [10]. Teabevaldajaid (avaliku sektori asutused, kuid ka ettevõtted, teadusasutused jt organisatsioonid, nt Tartu Linnavalitsus [11]) on rohkem kui andmestikke, sest portaalis on registreeritud ka need asutused, kes ei ole veel ühtegi andmestikku avaldanud.

Portaalis olevad andmed on üldjuhul tekstilisel kujul või viidetena andmete algportaalidesse. Andmestike uuendamise sagedus sõltub andmete iseloomust ja nende avaldajast, kuid reeglina uuenevad selles portaalis olevad andmed aeglaselt.

Portaali avalehel on lingid alamlehtedele: andmestikud, teabevaldajad, kasutuslood, juhendid, näited uusimatest andmestikest jms. On küll loodud võimalus andmete avastamiseks nii visuaalselt kui ka otsimiseks konkreetse valdkonna või metaandmete põhjal, kuid valikute katsetamine tekitab mulje, et portaalis on probleeme, näiteks ei avane mõni kasutuslugu ("Riigiraha") või ei tööta andmete visualiseerimine.

Portaalile saab anda tagasisidet ning jälgida statistikat andmestike kohta, näiteks allalaadimiste arv ning hinnangud andmestike kohta. Lisatud on ka vahendid arutelude loomiseks ning uute andmesoovide esitamiseks, kuid nende kasutajaid on praegu väga vähe. Portaaliga on integreeritud SPARQL-i päringute ning rakendusliidese tugi [12].

Lõputöö kirjutamise ajal ei olnud teabeväravas eriti asjade internetiga seotud andmeid, kuid soovi korral on neid võimalik sinna agregeeritud kujul lisada.

Tartu linna geoHUB

Tartu linna geoHUB [13] on linna ning tema haldusüksuste ruumiandmete näitamiseks mõeldud andmeportaal ArcGIS lahenduse baasil. Eesti avaandmete teabevärav koondab

eelkõige andmeid avalikus sektoris oleva teabe kohta allalaaditavate failide või linkide kujul kolmandatesse süsteemidesse, kuid Tartu geoHUB on näide ühest avaandmete keskkonnast, mis on mõeldud konkreetse linna ruumiandmete graafiliseks kuvamiseks.

GeoHUBi andmed on organiseeritud kategooriatesse “Avaandmed” ning “Valdkonnad”, näiteks “Planeeringud”, “Tehniline infrastruktuur” ja “Transport”, mille alt antakse ligipääs valdkondade andmetega seotud portaalivälistesse rakendustesse, näiteks “Tänavavalgustuse inventari töölaud” [14], kategooria “Avaandmed” kaudu pääseb ligi Transpordi ning Linnaruumi ja heakorra andmestikele. Ühel andmestikul võib olla mitu andmekihti, mida saab interaktiivse kaardi abil kasutada [13]. Samuti on loodud võimalus kaardil kasutatavaid andmeid alla laadida erinevates failiformaatides.

Detsembris 2022 ei olnud portaalis alamlehe “Avaandmed” all veel linna andurite andmeid. Loodud on võimalus anda portaalile tagasisidet ning esitada andmesoove hetkel puuduvate andmete lisamiseks [13]. Loodava lahenduse jaoks ei saa seda kasutada, kuna eeldab, et andmetel on sisuline seos geograafilise asukohaga.

1.3.2 CKAN

Kui Cumulocity tegeleb andmete kogumise ning vahesalvestamisega, siis CKAN on mõeldud kogutud andmete näitamiseks lõppkasutajale. CKAN [15] on programmeerimiskeelega Python loodud avatud lähtekoodiga ja rohkete võimalustega andmete haldamise süsteem, mis lihtsustab avaandmete hoidmist, kasutamist ning jagamist. Portaali kasutavad paljud erinevad organisatsioonid ning riigid, teiste hulgas näiteks Taani, kes kasutab seda oma energiaandmete jagamiseks [16].

Rakendusel on palju erinevaid võimekusi. Näiteks saab olemasolevatele metaandmetele lisada ka kasutaja defineeritud välju. Samuti on olemas võimekus otsida andmeid metaandmete põhjal ning luua andmestikest graafikuid. CKANi saab lihtsal moel muuta, sellel on olemas võimalused luua rakendusele oma kasutajaliides ning luua laiendusi, mille abil on võimalik portaaliga ühendada võimekusi, mis algses portaalis puuduvad [17]. Täpsemad juhised on portaali dokumentatsioonis [18]. Portaali on võimalik kasutada juurutades see oma valitud serverisse.

1.3.3 Olemasolev lahendus

2019. aastal koostas Silver Laius bakalaureusetöö raames programmeerimiskeelele Python integratsiooniteegi, millega saab Cumulocity IoT platvormilt andmeid eksportida failidena (nt CSV-formaadis) [19]. Teek võimaldab pärida andurite mõõtetulemusi (*measurements*), sündmusi (*events*) ning seadmete infot (*managedObjects*). Vastavalt päringu tüübile võimaldab teek filtreerida infot kõikide parameetrite järgi, mida toetab vastav Cumulocity rakendusliidese (API) päring, näiteks algus- ja lõppaeg, anduri tüüp ning identifikaator.

Allalaaditud andmed hoitakse Pythoni teegi Pandas *dataframe* tüüpi objektis, mis võimaldab andmeid vajadusel edasi analüüsida või salvestada need muuhulgas CSV-failitüübina. Valminud integratsiooniteegiga analüüsiti Tartu linna liiklusloendurite andmeid ja vaadati, kuidas 18. juulil 2019. aastal toimunud Metallica kontsert mõjutas liiklusvoogu [19].

Seda konkreetset teeki loodavas lahenduses ei kasutata, kuna teegi viimane uuendus Githubis oli 2020. aasta sügisel ning veaparandusi ning uusi võimekusi pole üle kahe aasta lisatud. Mais 2023 oli teegi Githubi lehel “Issues” all kolm veateadet, millest kaks olid seal olnud juba neli kuud ning nendele polnud teegi autor tähelepanu pööranud. Seega käesoleva töö autor eeldab, et teeki enam ei arendata ning mõttekam on samalaadne lahendus luua ise. Lisaks võib teegil esineda probleeme suuremahuliste andmefailide allalaadimisega, kuna kõik andmed laetakse kõigepealt mällu ning alles seejärel kirjutatakse need kettale. See teeb küll mugavamaks kohese andmeanalüüsi, kuid võib tekitada probleeme olukordades, kus programm töötab limiteeritud mäluga konteinerites ning paralleelselt laetakse alla mitut erinevat faili.

2. Loodava lahenduse nõuded ja kasutatud tehnoloogiad

Iga suurema tarkvaralahenduse loomise puhul on vaja enne arendustöö alustamist selgeks teha, milles seisneb probleem ja millised on loodava süsteemi vajadused, et kõigil osapooltel oleks ülevaade rakendusest juba selle valmimise ajal. Järgnevas peatükis selgitatakse, miks on loodavat lahendust vaja, loetletakse lahenduse funktsionaalsed ja mittefunktsionaalsed nõuded ja selgitatakse, milliseid tehnoloogiaid lahenduses kasutatakse.

2.1 Probleemi püstitus

TÜ Delta keskuses tekib palju erinevate ruumiautomaatika andurite lugemeid. Need andmed saadetakse Cumulocity platvormile (vt lisa 1). Praegu on andmete jagamine kõigi soovijatega aeganõudev protsess, kuna neid tuleb pärida käsitsi, sest automatiseeritud protsess on puudu. Lisaks on Cumulocity kasutamiseks vaja selle platvormi kasutajat, mida pole võimalik kõigile anda. Pole ka keskkonda, kuhu olemasolevaid andmefaile salvestada ning kus saaks nende seast sobivaid faile otsida. Seetõttu on vajadus luua süsteem, mis võimaldaks kõiki eelnimetatud asju teha.

Kogu loodava süsteemi eesmärk on luua lahendus, millega oleks võimalik Cumulocity platvormil olevaid andmeid etteantud mahus alla laadida, otsida konkreetset faili juba allalaaditud failide seast ning neid kõigi soovijatega jagada. Järgnevalt on välja toodud loodava lahenduse funktsionaalsed ning mittefunktsionaalsed nõuded. Süsteemi kasutajate haldusega ja grupiga seotud nõuded on väiksema prioriteediga ning nende haldamiseks luuakse võimekus vaid API tasandil.

Lahenduse funktsionaalsusteks on Cumulocity platvormil olevate andurite andmete allalaadimine, avalikustamine ja andurite ning failide otsimine nende metaandmete järgi

2.2 Lahenduse nõuded

2.2.1 Funktsionaalsed nõuded

1. Administraator saab Cumulocityst alla laadida anduri(te) poolt tekitatud andmepunktid valitud ajaperioodis.

    a. Andmeid saab Cumulocityst pärida ühekordselt, nt andmed viimase kahe päeva kohta, või perioodiliselt, nt viimase kahe päeva andmed iga kahe päeva tagant.
2. Korraga saab Cumulocity platvormilt alla laadida mitme erineva anduri andmed samast ajaperioodist.
3. Kõik tavakasutajad saavad temale nähtavaid faile alla laadida isiklikku arvutisse.
    a. Igas allalaaditavas failis on ühe anduri andmed.
4. Administraator saab haldussüsteemist faile kustutada.
5. Administraator saab faili avalikustada tavakasutajatele nägemiseks.
    a. Administraator teeb failid tavakasutajatele kättesaadavaks.
6. Avalikustatud faile saavad näha ja kasutada kõik soovijad.
7. Kõik kasutajatüübid saavad faile otsida metaandmete järgi (nt andmete ajaperiood, anduri nimi).
8. Administraator saab süsteemi lisada andureid ning andurite gruppe.
9. Administraator saab lisada andurile metaandmeid, nt korrus, ruum jne.
10. Administraator saab anduri metaandmeid uuendada.
11. Kõik kasutajatüübid saavad andureid otsida metaandmete järgi.
12. Administraator saab teha kõiki kasutaja taseme operatsioone.
13. Andurid ning grupid lisatakse süsteemi automaatselt Cumulocity süsteemi kaudu.
14. Administraator saab andureid grupeerida.
15. Administraator saab kustutada lokaalselt loodud andurite grupi.
16. Administraator saab süsteemi lisada kasutajaid.
17. Administraator saab muuta teiste kasutajate tüüpe.
18. Administraator saab kasutajaid eemaldada.

2.2.2 Mittefunktsionaalsed nõuded

1. Süsteem peab kasutajat teavitama, kui mõnes kasutaja algatatud taustatöös tekkis viga.
2. Uusi faile saab Cumulocity platvormil olevate andmete põhjal luua ainult administraator.
3. Tavakasutaja ei saa teha administraatorkasutaja operatsioone.

2.3 Kasutatud tehnoloogiad

2.3.1 Arenduskeel

Loodavas lahenduses kasutatakse nii ees- kui ka tagasüsteemi jaoks TypeScript programmeerimiskeelt Node.js käituskeskkonnas. See lihtsustab mõnevõrra lahenduse arendus- ning hiljem haldusprotsessi, kuna rakenduse kõikide komponentide juures on võimalik kasutada sama ökosüsteemi tööriistu.

TypeScript kujutab endast staatiliselt tüübitud JavaScripti [20] ehk TypeScript eksisteerib ainult koodi kompileerimise ajal ning käitusajal on vaid JavaScript. JavaScript oli algselt mõeldud kasutamiseks veebilehitsejates, kuid on tänaseks kasvanud väga populaarseks ning seda kasutatakse nii ees- kui ka tagarakenduste loomiseks.

TypeScripti kasutamine aitab vähendada käitusvigu, sest ei lase koodi käivitada, kui staatiliselt on aru saada, et tüübid erinevate koodiosade vahel ei klapi. See teeb arendusprotsessi mugavamaks ning suurte projektide puhul ka programmikoodi haldamise lihtsamaks. Kuigi TypeScript on staatiliselt tüübitud, ei ole tema andmetüübid nii ranged kui on näiteks Javas, ja see muudab keele väga paindlikuks. Näiteks TypeScriptis on numbrite tähistamiseks vaid kaks tüüpi: “number” ja “bigint” ning erinevate “number” tüüpi andmete vahelised teisendused on kasutajate eest peidetud. Nii on võimalik keskenduda rohkem funktsionaalsusele ning vähem süntaksile. Et loodavas lahenduses on palju tegemist JSON-tüüpi andmetega, siis on neid TypeScriptiga mugav hallata, sest JSON teisendub otse JavaScripti objektiks.

2.3.2 Serveripoolne raamistik

NestJS on Node.js-il põhinev raamistik tagasüsteemide ehk serveripoolsete rakenduste loomiseks ning see kombineerib objektorienteeritud, funktsionaalprogrammeerimise ning funktsionaalse reaktiivprogrammeerimise (*functional reactive programming*) paradigmasid. Toetatud on nii Javascripti kui ka TypeScripti keeled. Loodavas lahenduses kasutatakse just seda raamistikku, sest see aitab hoida koodi arhitektuuri ühtlase ja selgena ning annab standardiseeritud tööriistad enamlevinud operatsioonide tegemiseks, näiteks kasutajate autentimiseks ja andmebaasiga suhtlemiseks. Selle abil on võimalik rohkem keskenduda

loodava rakenduse sisule [21]. Raamistik kasutab oma funktsionaalsuste jaoks palju annotatsioone ehk metaandmeid koodi kohta.

2.3.3 Andmebaas

Andmebaasina kasutatakse MongoDBd [22] ja sellega liidestumiseks Mongoose [23] teeki. Andmebaasis hoitakse saadaval olevat andurite, allalaetud failide ning hetkel töös olevate ja kaua kestvate taustatööde informatsiooni. MongoDB on dokumendisalvestuse andmebaas, kus ühe olemi andmeid hoitakse koos ühes dokumendis, vajadusel võivad andmed olla mitmetasandilised. Seetõttu on sõltuvalt andmemudelist vaja harva teha erinevate andmekollektsioonide (tabelite analoog) liitmisi. Kuna andmeid hoitakse BSONi ehk binaarse JSONi formaadis, siis sobivad andmebaasi kõige paremini just JSONi stiilis andmed.

Et MongoDB ei kasuta vaikeväärtusena ranget andmemudelit, siis saab ühes kollektsioonis hoida väga erineva struktuuriga dokumente. Näiteks võib ühe olemi kõiki erineva struktuuriga alamolemeid hoida samas kollektsioonis, kui neil on olemas mingi väli, mille abil neil vahet teha. See teeb ka andmemudeli uuendamise väga lihtsaks, sest näiteks uute väljade lisamiseks ei ole tarvis kasutada andmebaasi migratsioone. MongoDB sai valitud eelnevalt välja toodud asjaolude tõttu ning kuna lahenduses kasutatavad olemid on mitmetasandilised, siis on neid mugav MongoDB abil hallata. Kui kasutada ranget SQL-keelel põhinevat andmemudelit, siis oleks vaja teha väga palju tabelite liitmisi ning andmemudelist oleks keerulisem aru saada.

Mongoose ODM (*Object Document Mapping*) teeki kasutatakse selleks, et hoida arenduse ajal andmemudelit staatilisena ning et kontrollida salvestatavate andmete vastavust andmemudelile. Teek võimaldab defineerida oma andmemudeli, kontrollib objektide vastavust mudeliga ning lihtsustab andmebaasis päringute tegemist. Seejuures ei takista ta kasutamast MongoDB dünaamilisusest tulenevaid võimalusi.

2.3.4 Suhtlus komponentide vahel

Tagalahendus

Tagalahenduses kasutatakse erinevate alamkomponentide vahel suhtlemiseks RabbitMQ [24] sõnumivahenduse teenust. Sõnumivahetusteenused aitavad lahenduse komponente nõrgemalt

sidestada ehk erinevad komponendid ei pea üksteisest teadma rohkem kui suhtlemise liidest. Seega iga komponendi vaatest jääb ära teiste komponentide leidmise protsess, näiteks millisele aadressile sõnumeid saata, sest selle eest hoolitseb sõnumivahendusteenus. Samuti puhverdavad nad sõnumeid, kuni nende kohalejõudmiseni sihtpunkti, mis aitab vähendada koormust igale süsteemi alamkomponendile.

Sõnumivahendusteenustest on hetkel ühed kõige populaarsemad Apache Kafka ning RabbitMQ. Kafka on hea tööriist, millega on lisaks sõnumite vahetamisele võimalik ka juba saadetud sõnumeid uuesti algusest lugeda, kuna juba saadetud sõnumeid on vaikimisi võimalik alles hoida 168 tundi. [25]. Kuid et loodavas lahenduses on vaja ainult lihtsat ja kiiret viisi sõnumite vahetamiseks erinevate komponentide vahel, siis juba saadetud sõnumite salvestamine ning uuesti esitamine ei ole oluline. Seetõttu valiti kasutamiseks RabbitMQ.

Eesrakendus

Eesrakenduse peamiseks arendusteegiks on React.js [26], mis on Facebooki loodud avatud lähtekoodiga JavaScripti teek kasutajaliideste loomiseks, samuti on sellel tugi TypeScripti kasutamiseks. Stack Overflow 2022. aasta tarkvaraarendajate tagasiside põhjal selgus, et React.js on hetkel kõige populaarsem teek veebiliideste loomiseks [27]. Teek põhineb ideel luua taaskasutatavaid komponente, mida on võimalik kombineerida teiste komponentidega, et luua uusi erisuguseid lehti. Kuna Reacti komponendid on tegelikult JavaScripti funktsioonid, on nendes võimalik kasutada kõiki JavaScripti võimekusi. Kasutajakogemuse tõstmiseks kasutatakse JSX (*JavaScript Syntax Extension*) süntaksit, mis võimaldab JavaScripti funktsioonides tagastada otse HTML-koodi. Joonis 4 annab ühe näite React.js funktsioonist, milles on kasutatud JSX süntaksit. Eesrakenduse jaoks sai valitud see teek tema kasutusmugavuse, kiiruse ning lisade ja abimaterjalide suure hulga tõttu.

```
function VideoList({ videos, emptyHeading }) {
  const count = videos.length;
  let heading = emptyHeading;
  if (count > 0) {
    const noun = count > 1 ? 'Videos' : 'Video';
    heading = count + ' ' + noun;
  }
  return (
    <section>
      <h2>{heading}</h2>
      {videos.map(video =>
        <Video key={video.id} video={video} />
      )}
    </section>
  );
}
```

Joonis 4. React.js funktsiooni näide kasutades JSX süntaksit [26]

Ainult React.js-ist veebiliidese terviklahenduse loomiseks ei piisa, sest see ei anna häid vahendeid muu loogika tegemiseks, näiteks erinevate alamlehtede vahel navigeerimiseks, andmete pärimiseks või kasutajate autentimiseks. Need komponendid tuleb arendajal luua ise või kasutada mõnda olemasolevat lahendust. Kuna neid probleeme on juba palju kordi lahendatud, ei olnud autori arvates mõistlik hakata neid nullist ise implementeerima. Seetõttu kasutatakse muude operatsioonide lihtsustamiseks raamistikku Refine [28]. Tegemist on React.js põhise raamistikuga, mis sobib kõige paremini administraatori paneelide ning firmasiseste tööriistade tegemiseks. See raamistik lihtsustab tunduvalt päringute tegemist tagarakendusse, kasutajate autentimist, alamlehtede vahel navigeerimist ning lehtede stiliseerimist. Ühtse disaini (nupud, tekstiväljad jm) hoidmiseks kasutatakse MUI (*Material UI*) [29] komponentide teeki.

3. Arhitektuur ning tööprotsess

Järgnevalt selgitatakse, kuidas lahendust ehitati ja põhjendatakse mõningaid arhitektuurilisi valikuid. Samuti esitatakse arhitektuuri joonis koos täpsema kirjeldusega ning antakse ülevaade sellest, kuidas toimuvad taustatööd. Viimases alajaotises käsitletakse failide alla ja üleslaadimist ning nendega seotud probleeme.

3.1 Süsteemi üldkirjeldus

Esimene suurem küsimus peale nõuete analüüsimist ehk arusaamist *mida* teha on otsustamine *kuidas* oleks neid nõudeid kõige mõistlikum täita. Et hilisem arendusprotsess läheks sujuvamalt, on mõistlik kõigepealt luua kogu süsteemist mudel, mille alusel on võimalik implementeerida spetsiifilisemad komponendid. Autor kaalus lahenduse juures monoliitset ning mikroteenuste põhist hajustatut lähenemist. Monoliitne süsteem on süsteem, mille kõik osad töötavad ainult ühe protsessi sees või koosneb see erinevatest moodulitest, mida saab eraldiseisvalt arendada, kuid süsteemi juurutamiseks (*deploy*) tuleb seda teha kõikide moodulitega koos.

Seevastu mikroteenustel põhinevas süsteemis on iga teenus ülejäänud süsteemist täielikult eraldatud, neid ühendab vaid suhtlusliides. Selline eraldatus suurendab vabadust valida igale teenusele oma tehnoloogia ning suurendab süsteemi vastupidavust vigadele. Vea korral ühes mikroteenuses saavad ülejäänud teenused edasi töötada, juhul kui neil pole otseselt vaja vigasest teenusest andmeid võtta. Lihtsustub ka süsteemi skaleerimine ning juurutamine, sest iga mikroteenuse puhul on seda võimalik teha eraldi [30].

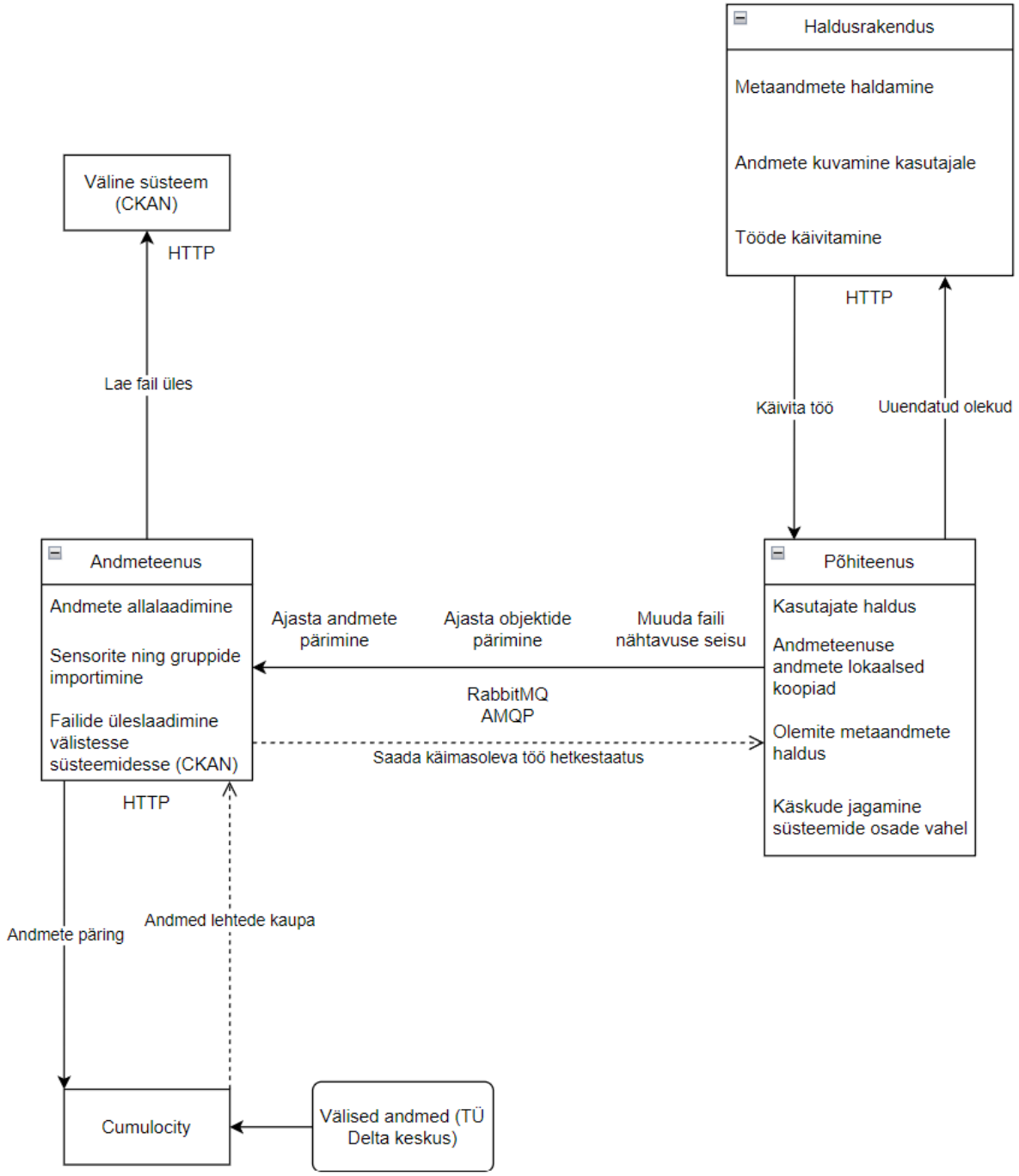
3.2 Arhitektuur

Järgmisel leheküljel joonisel 5 on kirjeldatud loodud süsteemi kõrgetasemelist toimimist. Nii sellel kui ka järgnevatel joonistel tähendab punktiirjoon asünkroonset vastust. Lahendus koosneb kolmest suurest komponendist:

1. Veebipõhine haldusrakendus
2. Põhiteenus
3. Andmeteenus.

Haldusrakendus tegeleb andmete kuvamise ning erinevate taustatööde käivitamisega. Üks taustatöö on näiteks selline, et 1. toimub ühe anduri andmete allalaadimine teatud

ajaperioodist, nt 01.05 - 03.05 või nii pikas vahemikust kui on vaja, 2. nende salvestamine faili ning 3. tulemuse kuvamine haldusteenuses. Andmeteenus suhtleb põhiteenusega asünkroonselt kasutades RabbitMQ vahendatud sõnumeid. Andmeteenus tegeleb andurite andmete allalaadimisega ning Cumulocity platvormil olevate andurite importimisega süsteemi, et kasutajatel oleks võimalik tutvuda loodud süsteemis olevate anduritega. Samuti on teenuse üks ülesanne laadida olemasolev fail kasutaja käsu peale välisesse süsteemi.



Joonis 5. Lahenduse üldjoonis

Põhiteenus seob haldusrakenduse ning andmeteenuse funktsionaalsused kokku ühtseks tervikuks. Selle jaoks käitub ta teiste teenuste vahel käskude vahendajana ning haldusrakenduse jaoks kõikide protsesside hetkeseisu hoidjana. Haldusrakendus näitab kasutajatele süsteemi olekut põhiteenuse teenuse andmete põhjal.

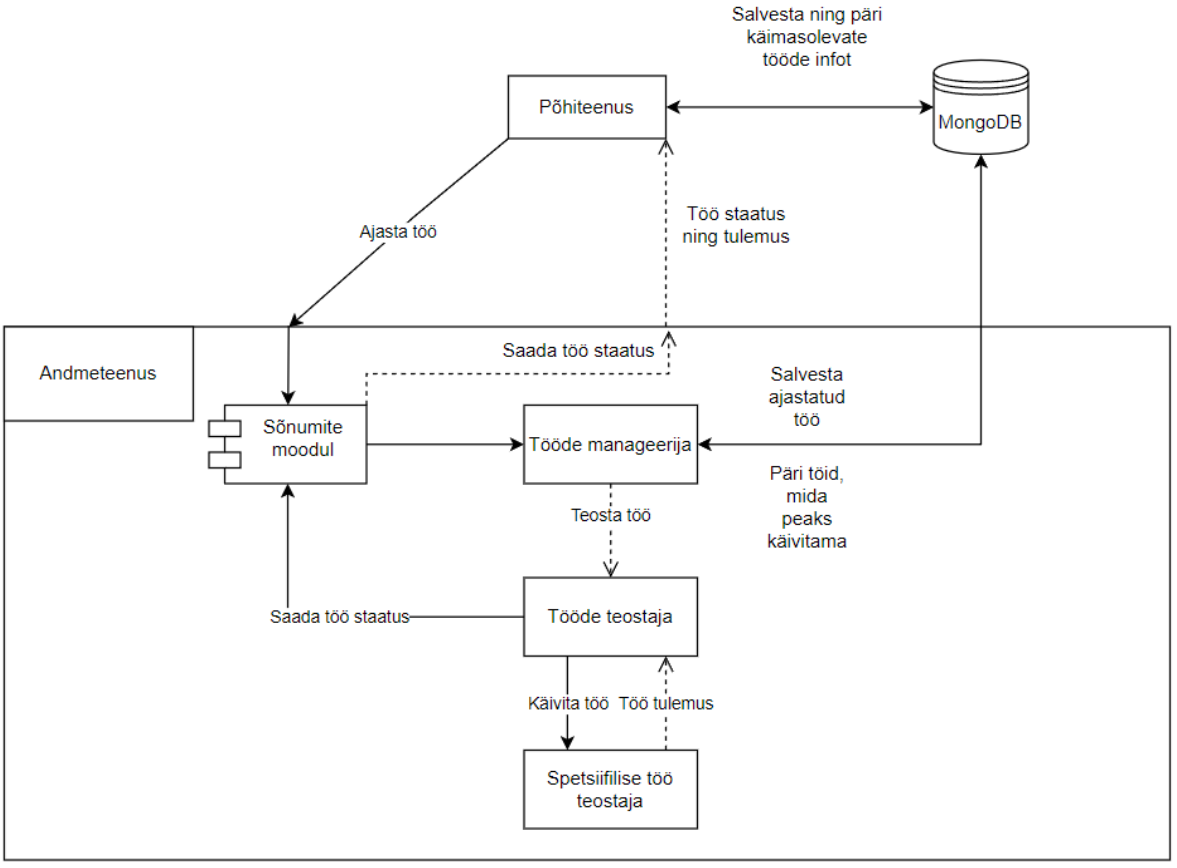
3.3 Taustatööd

Cumulocityst andmete pärimine ning nende salvestamine failideks võib võtta rohkem kui mõned sekundid, sõltuvalt päritavast andmemahust. Kuna erinevad andurid tekitavad erineval hulgal andmeid samast ajaperioodist, pole ka ette teada, kui kaua võib käivitatud töö kesta. Samuti võib olenevalt andurite arvust nende sünkroniseerimise töö (impordib andurid ning andurite grupid Cumulocity süsteemist loodud lahendusse) võtta kaua aega. Seetõttu tekkis vajadus luua süsteemis taustatöid, mis saaksid taustal töötada muid operatsioone segamata. Töö tulemus tuleb tagastada asünkroonselt.

Implementatsiooni algusfaasis oli kaalumisel kasutada ka taustatööde tegemiseks põhiteenust (ehk andmeteenust mitte kasutada). Kuna JSONi parsimine on Node.js-is sündmuste tsüklit (*event loop*) blokeeriv operatsioon, siis suuremamahuliste andmete korral tuleks seda teha eraldi lõimes, muidu ei pruugi server kasutajale alati kiirelt vastata. Probleemi lahendamiseks kaaluti algul Node.js töölõimede (*worker threads*) kasutamist. Kuna aga lõimede tugi on Node.js juurde lisatud hilisemates versioonides ning JavaScript oma olemuselt ei ole mitmetuumaline keel, siis oleks seda lähenemist kasutades läinud lahendus liialt keeruliseks. Samuti suurenes hiljem erinevate tehtavate tööde arv. Seetõttu otsustati kasutada mikroteenuste põhist lähenemist ning kõik andmetega seotud potentsiaalselt pikad taustatööd teha eraldiseisvas andmeteenuses. Vajadusel saaks käima panna ka mitu andmeteenuse koopiat paralleelselt, kuid käesolevas töös kasutati testimiseks ainult ühte koopiat. Mitme andmeteenuse puhul saadaks RabbitMQ ühe töö ühele andmeteenuse koopiale ringmeetodil (*round-robin*).

Joonisel 6 (järgmisel leheküljel) on kirjeldatud lahenduses kasutatavat taustatööde haldajat. Andmeteenuse sees olev suhtlus alamkomponentide vahel toimub kasutades JavaScripti funktsioonide väljakutseid. Kõik tööd luuakse kasutaja käsu peale põhiteenuses, kus nende staatuse info salvestatakse lokaalsesse andmebaasi ning seejärel saadetakse sõnumivahendusteenusele RabbitMQ sõnum töö ajastamise kohta.

Kõik joonisel 6 andmeteenuse kasti sees olevad funktsionaalsused kuuluvad andmeteenuse alla ning ei ole eraldi teenused. Andmeteenus kuulab tööde ajastamise sõnumeid. Uue töö puhul võtab sõnumite moodul selle vastu ning saadab selle tööde manageerijale. Tööde manageerija salvestab sissetuleva töö andmed oma andmebaasi, kusjuures andmeteenuse andmebaas on erinev põhiteenuse omast. Mõlemal teenusel on oma loogiline andmebaas, et nad saaksid olla üksteisest täielikult lahus ning ei saaks potentsiaalselt muuta teise teenuse andmeid. Vastasel juhul on oht, et teenused põimuvad liiga tugevalt, mis aga nurjab kogu eesmärgi kasutada hajusat süsteemi.



Joonis 6. Tööde struktuur

Tööde manageerija, mis on üks alammoodul andmeteenuses, esitab perioodiliselt andmebaasi päringu uute tööde teostamise vajaduse kohta. Töid päritakse loogiliselt eraldiseisvast andmeteenuse andmebaasist, et see ei oleks kuidagi seotud põhiteenuse poolt defineeritud andmemudeliga. Nii on võimalik andme- ning põhiteenust arendada ning uusi versioone juurutada eraldi. Töö on vaja teostada siis, kui töö alustamise aeg on kätte jõudnud ning seda

juba ei teostata. Sõltuvalt töö tüübist käivitatakse töö perioodiliselt. Käivitatud töö saadetakse tööde teostajale JavaScripti funktsioonide väljakutsete kaudu, mis teostab töö ning saadab seejärel töö staatuse tööde moodulit kasutades tagasi. Vea tekkimisel saadetakse samuti töö staatuse sõnum, et teavitada sellest kasutajat. Nii on võimalik tööde tegijat paremini skaleerida kasutades näiteks Kubernetest. Rakenduse skaleerimine jääb käesolevast tööst välja ning on jäetud tulevikutööks.

3.4 Failide alla ning üleslaadimine

Kuna lahendust võib potentsiaalselt kasutada mitu kasutajat korraga, võib juhtuda, et mitme anduri andmeid laaditakse alla korraga. Suure jõudlusega serveri puhul ei pruugi see probleeme tekitada, kuid väikese mälulimiidiga konteinerite korral võib see halvimal juhul tekitada *out-of-memory* tüüpi veateateid. Seetõttu tuleks võimalusel suuremate failide alla või üleslaadimiseks kasutada voogedastust (*streams*) või pärida andmeid väikeste tükkide kaupa ning salvestada need samasse faili.

Valmis failide üleslaadimiseks kasutatakse voogedastust ning Cumulocityst andmete pärimiseks väikeste tükkide kaupa andmete pärimist. Cumulocity API võimaldab pärimisel määrata vastuse lehekülje suuruse ehk mitu erinevat andmerida korraga tagastada. Kui määrata lehe mahuks näiteks 50 andmerida, on võimalik viia leheküljel olevad andmed sobivasse formaati, kirjutada need faili ning seejärel korrata protsessi kuni andmete lõppemiseni. Seejuures hoitakse mälukasutus miinimumi juures, sest korraga on vaja mälus hoida vaid ühe lehe jagu andmeid. Samasugust protsessi kasutatakse nii mõõtetulemuste kui ka andurite metaandmete pärimiseks.

Andurite mõõtetulemusi on võimalik Cumulocity API kaudu alla laadida vaikeväärtusena JSON-formaadis, kuid toetatud on ka CSV- ning XLSX-tüüpi failide allalaadimine [6]. Siin tekkis autoril allalaadimisega seoses probleem. Nimelt tagastatakse kasutajale allalaaditud andmefailid loodud lahenduses CSV-formaadis, kuid need failid saadakse sel viisil, et mõõtetulemused laaditakse alla JSON- formaadis ning seejärel teisendatakse CSV-formaati. Kui on soov lisada faili lisaks mõõtetulemustele ka enda defineeritud tulpasid, nt link mõõtetulemusele või objektile Cumulocity süsteemis, siis sel juhul on mõistlikum kasutada lehtede kaupa JSON-formaadis andmete pärimist. Nii on võimalik täpselt kontrollida kogu allalaadimise ning andmete salvestamise protsessi. Kui aga Cumulocity tulpade struktuur

(*time*, *source*, *device_name*, *fragment.series*, *value*, *unit*) sobib, siis on võimalik CSV-fail otse Cumulocity süsteemist alla laadida sel viisil, et päringut tehes lisatakse päisesse väärtus "Accept: text/csv".

Juhul kui andmeridu on väga palju, siis ei tagastata andmeid kohe. Esialgu tagastatakse kasutajale ainult loodava andmefaili nimi ning faili ettevalmistamist jätkatakse taustatööna. Kui fail on valmis, on see võimalik eraldi päringuga alla laadida. Kuna autor ei leidnud andmete allalaadimise moodulit implementeerides infot selle kohta, mida peab edasi tegema olukorras, kus andmeridade arv muutub Cumulocity jaoks liiga suureks, et see ühe päringuga tagastada, siis otsustati kasutada JSON-formaadis andmete pärimist. Seega oleks otse CSV-tüübis andmete allalaadimine Cumulocityst efektiivsem loodud lahenduse jaoks kui see JSON-tüübis andmete põhjal luua. Siiski on sellist lähenemist võimalik kasutada vaid mõõtetulemuste pärimiseks. Cumulocity objektide metaandmete pärimiseks, mille põhjal on loodud lahenduses võimalik näha kasutatavaid andureid, tuleb kasutada JSON-formaadis andmete pärimist.

4. Valminud lahendus

Järgnevas peatükis antakse ülevaade valminud lahenduse kolmest põhilisest alamteenusest, andmeteenusest, põhiteenusest ja veebipõhisest haldusrakendusest. Seejärel kirjeldatakse, mida on võimalik praeguseks hetkeks valminud lahendusega teha. Käsitletakse ka lahenduse valideerimist ja võimalusi lahenduse edasi arendamiseks.

4.1 Lahenduse ülevaade

Valminud lahendus koosneb kolmest alamteenusest: andmeteenus, põhiteenus ning veebipõhine haldusrakendus. Kasutajavaatest on nendest kõige olulisem just viimane, sest selle kaudu interakteerub kasutaja süsteemiga. Järgnevalt kirjeldatakse haldusrakenduse tööd.

Haldusrakenduse kasutamiseks on vajalik lahenduse kasutaja olemasolu. Konto võib olla administraatorkasutaja või tavakasutaja tüüpi. Administraatorkasutaja saab süsteemis teha kõike ehk nii vaadata olemasolevat infot kui ka luua uusi objekte, nt uusi andmefaile. Tavakasutajal on ainult vaatamisõigused. Kuna tavakasutaja ei saa teha süsteemi seisu muutvaid operatsioone, ei ole sellel kasutajatüübil vajalik Cumulocity konto olemasolu. Administraatorkasutajat on küll võimalik luua ilma Cumulocity kasutajata, kuid uute andmete allalaadimiseks on vajalik selle olemasolu. Administraatorkasutaja on mõeldud uute andmefailide ettevalmistamiseks ning tavakasutaja juhuks, kui tema ainus vajadus on loodud lahenduses olemas olevatele andmetele ligi pääseda.

Haldusrakendus on jagatud neljaks suureks alamosaks: *Sensors*, *Files*, *Tasks* ja *Groups*. Need omakorda on jagatud kahte tulpa. Vasakus tulbas on võimalik valida soovitud filtreerimised ning paremal, valitud jaotise listivaates, on igas reas objekti põhiline info (vt lisa 3).

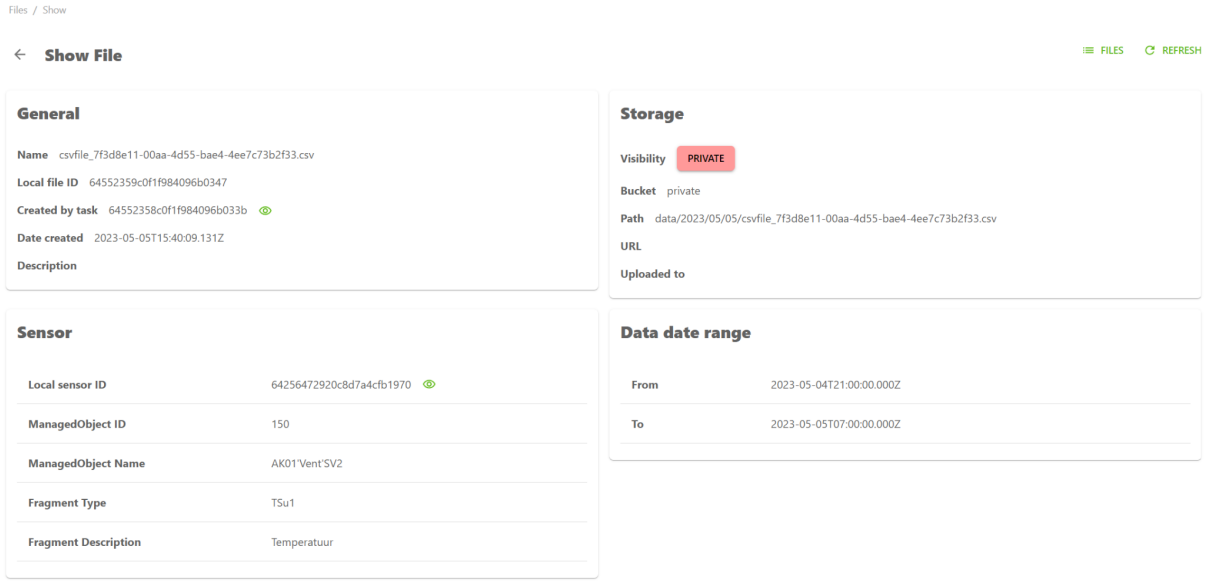
Alajaotise *Sensors* all (vaikimisi avaleht) on võimalik näha lahendusse imporditud andureid (vt lisa 3). Lahenduses moodustub üks andur Cumulocity atribuutide manageeritud objekti id (*managedObjectId*) ja väärtusfragmendi tüübi (*valueFragmentType*) paarist.

Ühel manageeritud objektil (vt joonis 2) võib Cumulocity süsteemis olla mitu erinevat väärtuse fragmenti. Joonisel 7 on kirjeldatud ühe manageeritud objekti väärtusfragmente.

```
{
    "c8y_SupportedMeasurements": [
        "DP1",
        "TSu",
        "DP2",
        "TSu2",
        "DP",
        "TSu1"
    ]
}
```


Joonis 7. Ühe manageeritud objekti väärtusfragmendid

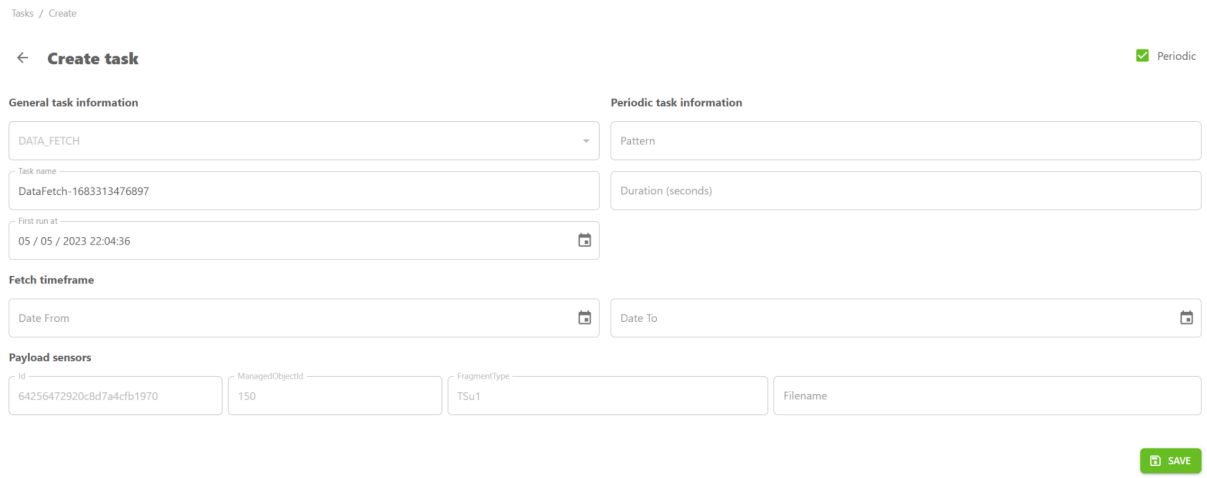
Üks väärtusfragment (nt TSu, temperatuur TÜ Delta keskuses) näitab andmetüüpi, mille mõõtetulemusi on potentsiaalselt võimalik alla laadida. Samuti on sõltuvalt objektist võimalik vaadata objekti spetsiifilist infot - erinevatel objektitüüpidel on erinevad spetsiifilised väljad.



Joonis 8. Ühe faili täpne info

Näiteks joonis 8 näitab ühe faili kohta täpset infot, nt andmete ajavahemik ning faili salvestuse asukoht. Samuti on lingitud seotud objektid, mille täpse info lehele on võimalik minna rohelisele silma ikoonile vajutades.

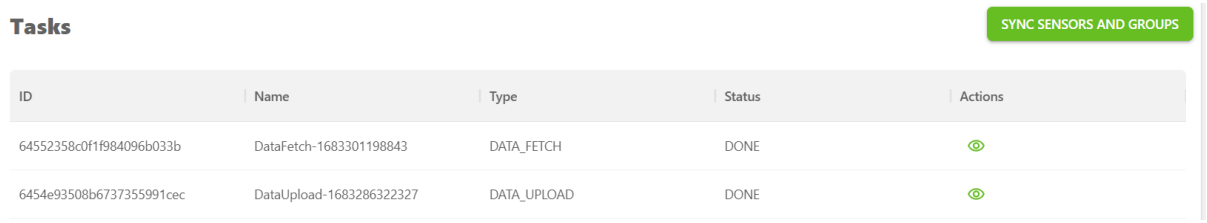
Vaatest *Sensors* (lisa 3) on võimalik alustada anduri mõõtetulemuste allalaadimist niimoodi, et valitakse soovitud andur ning vajutatakse "FETCH SENSOR DATA" nupule. Seejärel jõutakse järgmisesse vaatesse, mis on joonisel 9.



Joonis 9. Anduri mõõtetulemuste allatõmbamise seadistamine

Enne andmete allatõmbamist (vt joonis 9) saab seadistada soovitud ajavahemiku, failinime, töö nime ning perioodilisuse info CRON-mustri abil nt 0 */5 * * * *, mis tähendab, et tõmba uus fail alla iga 5 minuti tagant tõstes ajaperioodi igal sammul edasi.

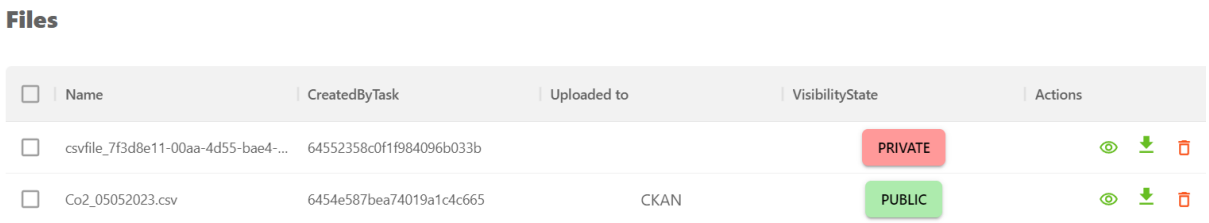
Hetkel töös olevaid allalaadimisi on võimalik vaadata alajaotisest *Tasks* (vt joonis 10), kus kuvatakse töö nimi, tüüp ning olek. Näiteks anduri mõõtetulemuste allalaadimise töö detailvaates kuvatakse allalaaditava faili info ning vajadusel ka veateade töö kohta. Veateate puhul liigub töö olekusse "FAILED".



Joonis 10. Taustatööde listivaade

Kui töö on valmis ehk liikunud olekusse "DONE", on allalaetud fail kuvatud jaotises "Files" (vt joonis 8 ning 11). Samuti on failide listivaatest võimalik valmis fail alla laadida, neid

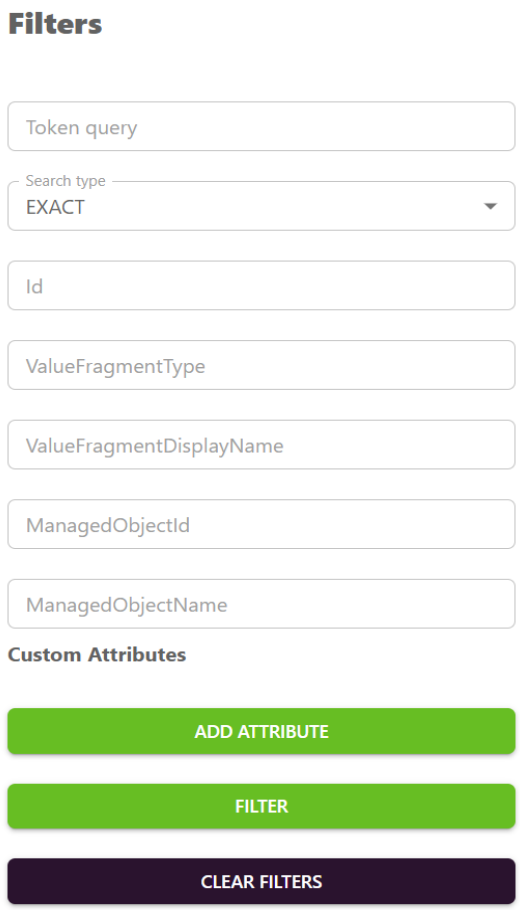
kustutada, muuta nähtavuse staatust (tavakasutajatele kuvatakse vaid avalikud failid) ning laadida fail üles välisesse süsteemi.



Joonis 11. Failide listivaade

Hetkel on integratsioon loodud vaid CKANi süsteemiga, kuid tulevikus on võimalik implementeerida tugi ka teistele välistele süsteemidele. Samuti eeldab autor loodud lahenduses, et CKANi süsteem on juba vajadusel ilustatud ning lisatud juurde vajalikud otsimisväljad. Hetkel pandi süsteem töötama vaikeväärtustel.

Kuna süsteemis võib olla palju erinevaid objekte ning objektidel palju erinevaid atribuute (näiteks testimisel leidis süsteem TÜ Delta keskuses süsteemist 526 erinevat andurit), siis on vaja võimekust erinevaid objekte (andureid, faile jne) otsida. Selleks on igas alajaotises antud vastavate objektide spetsiifilised filtreerimisvalikud. Näiteks joonisel 12 on välja toodud andurite spetsiifilised filtreerimise valikud.



Joonis 12. Andurite filtreerimise valikud

Kuna kasutajad ei pruugi alati teada atribuutide täpseid nimesid, nt võib Delta keskuse andmestikust leida väärtusfragmendi “KickFnct1.OpSta” või manageeritud objekti nimega “JSAK01'JahSolm'FanCoi2'P13”, siis võeti selliste atribuutide nimede järgi otsimiseks kasutusele kolme tüüpi otsingurežiimid. Otsingu vaikeväärtus on alati täpne otsing ehk “EXACT”. Lisaks on võimalik kasutada “PREFIX” tüüpi otsingut, sellise valiku puhul otsitakse soovitud väljadelt mittetõstutundliku prefiksotsinguga. Kolmas võimalus on kasutada otsimiseks režiimi “TOKEN”, mis kasutab MongoDB tekstiindeksit [31]. See on andmebaasi MongoDB sisseehitatud viis täistekstiotsinguks. “TOKEN” otsingurežiimis otsitakse ManagedObjectName, ValueFragmentType ning ValueFragmentDisplayName väljadelt.

Suurte andmemahtude korral on tekstiindeks kiirem kui prefiksotsing, sest prefiksotsing ei saa kasutada MongoDB indekseid ning seetõttu tuleb otsinguks kogu andmebaasi

kollektsioon läbi vaadata. Kahjuks ei tööta MongoDB tekstiindeks hästi Delta keskuse andmetega, kuna see ei oska teksti tükeldada ülakoma juurest. Andurite andmetest otsides on võimalik tekstiindeksit kasutada nende väljade korral: *ValueFragmentType, ValueFragmentDisplayName* ja *ManagedObjectName*. Edasiarendusena oleks tulevikus siin võimalik kasutada mõnda spetsiaalselt otsinguteks mõeldud lahendust, näiteks Apache Solr või Elasticsearch.

4.2 Võimekuste kirjeldus

Järgnevalt antakse ülevaade sellest, mida on hetkel võimalik valminud lahendusega teha. Lahendusega on võimalik alla laadida Cumulocity platvormil olevate andurite mõõtetulemusi valitud ajaperioodist. Vajadusel on võimalik alla laadida mitme anduri mõõtetulemused korraga. Piirang on, et kõikide andurite mõõtetulemustele rakendatakse sama ajaperiood ühe töö kontekstis. Mõõtetulemused laaditakse alla CSV-formaadis failidena. Alla laaditud faile on võimalik avalikustada loodud süsteemis ning laadida üles välistesse süsteemidesse. Lokaalselt lahenduses avalikustamine tähendab seda, et ka tavakasutajatel on võimalik allalaaditud failidele ligi pääseda. Tavakasutajad (ehk ilma administraatori õigusteta kasutajad) näevad sisse logides vaid avalikustatud faile “Files” listivaates. Lahenduse valmimise hetkel oli loodud integratsioon ühe välise süsteemiga (CKAN), kuid tulevikus on võimalik implementeerida tugi ka muudele välistele süsteemidele, kuhu vajadusel valmis fail üles laadida. Andmed laaditakse alla selle kasutaja Cumulocity kontoga, kes töö käivitas.

Kasutajate töö lihtsustamiseks on võimalik automaatselt importida Cumulocity platvormilt andurid ning andurite grupid. Lahenduses juba olemasolevate objekte (andurid, failid, taustatööd ning grupid) leidmiseks on neid kõiki võimalik filtreerida objekti spetsiifiliste atribuutide järgi. Näiteks lisaks muule on andureid võimalik otsida Cumulocity väärtusfragmentide ning manageeritud objektide nimede järgi ning olemasolevaid faile nendes olevate andmete algus- ning lõpuaja järgi. Otsida saab ka osaliselt ehk prefiks otsinguga. Kuna TÜ Delta keskuse andurite nimedest ei ole kohe võimalik üheselt aru saada anduri olemusest ning asukohast, siis loodi võimalus anduritele lisada ka enda defineeritud atribuute (*custom attributes*). Nende abil on võimalik andurile lisada näiteks ruumi ning korruse informatsiooni. Enda defineeritud atribuutide järgi on andureid võimalik ka filtreerida. Samuti on anduritele võimalus lisada kirjeldus ning väärtusfragmendi (*valueFragmentType*) kirjeldus andmetüübi kiiremaks hoomamiseks. Juba lisatud atribuute

on ka võimalik muuta. Kuna iga objekti kohta on rohkem informatsiooni, kui mahub listivaatesse, siis täpne informatsioon on välja toodud eraldi objekti spetsiifilises vaates (vt joonis 8).

Failide vaates on võimalik süsteemis olemas olevaid faile alla laadida, neid süsteemist kustutada ning vaadata nende kohta täpsemat informatsiooni. Tööde vaade on mõeldud hetkel töös olevate taustatööde jälgimiseks (näiteks kas töö on juba lõpuni jõudnud või mitte). Tööde vaatest toimub ka andurite ning andurite gruppide importimine Cumulocity platvormilt. Töö loomisel saab määrata ka spetsiifilisemat informatsiooni, näiteks enne mõõtetulemuste allalaadimist on võimalik seadistada andmete alguse ning lõpu kuupäevad, määrata allalaaditavatele failidele nimed ning määrata, kas andmeid peaks alla laadima perioodiliselt. Perioodilisuse intervalli saab määrata *cron* mustri abil, samuti saab määrata igal intervalli sammul alla laaditavate andmete mahu (mitu sekundit ajas tagasi).

4.3 Valideerimine ning järgnevad sammud

4.3.1 Valideerimine

Lahendust valideeriti TÜ Delta keskuse Cumulocity platvormi põhjal. Lahenduse funktsionaalsuse testimiseks kasutatakse automaattestide ning manuaalse testimise kombinatsiooni. Kahjuks ei jõudnud autor luua automaatteste kõikidele lahenduse funktsionaalsustele ning need jäävad tulevikutöödeks, kuna töö kujunes eeldatust mahukamaks. Põhiteenuse funktsioonid on kaetud 41.07% ulatuses testidega ning andmeteenus 12.5% (Jest teegi hinnangul). Järgnevalt on välja toodud läbi viidud manuaalsed testid. Kõikide testide puhul on atribuudid nimetatud nii, nagu nad on nimetatud lahenduse haldusrakenduses.

Test 1
Pealkiri: Ühekordselt anduri andmed laetakse korrektselt alla failina
Eeldused: Kasutaja on sisse logitud administraatori kasutajaga
Andmed:

- ManagedObject=150
- FragmentType=TSu1
- Date From=”02/05/2023 10”
- Date To=”03/05/2023 10”
- Task name=”Andmetöö 1”

- First run at=Vaikeväärtus
- Filename=Andmefail1

Sammud:

- “Files” alajaotises valida andur ning vajutada “FETCH SENSOR DATA” nupul. Täita lahtrid eelmises punktis mainitud andmetega. Vajutada “SAVE” nupul.

Eeldatav tulemus:

- Kasutaja viiakse “Tasks” ekraanile
- Töö nimega ”Andmetöö 1” jõuab loetud sekundite aja jooksul (võib mõnevõrra erineda vastavalt hetke interneti kiirusele) olekusse “DONE”.
- “Files” alajaotisse tekib uus rida failinimega “Andmefail1.csv” olekus “PRIVATE”.

Tegelik tulemus:

- Saavutati soovitud tulemus

Test 2

Pealkiri: Avalikku faili on võimalik alla laadida

Eeldused:

- Kasutaja on sisse logitud administraatori kasutajaga
- Olemas on fail testist 1

Sammud:

- “Files” alajaotises valida fail nimega “Andmefail1.csv” ning vajutada allalaadimise nupule (nool). Ei tohi midagi juhtuda
- Vajutada nupule “PRIVATE”. Nupp peab näitama laadivat olekut (keerlev ring) ning minema olekusse “PUBLIC” (võib võtta aega).
- Vajutada uuesti allalaadimise nupule.
- Vajutades uuesti nupule “PUBLIC” viia fail tagasi privaatsesse olekusse
- Vajutada uuesti allalaadimise nupule

Eeldatav tulemus:

- Faili olekus “PUBLIC” allalaadimise nupule vajutades peab süsteem laadima faili alla kohalikku arvutisse.
- Olekus “PRIVATE” allalaadimise nupule vajutades ei tohi midagi juhtuda

Tegelik tulemus:

- Saavutati soovitud tulemus

Test 3

Pealkiri: Alla laaditud failis on korrektne arv andmeridu

Kirjeldus: Võrreldakse lahenduse tekitatud faili ning Cumulocity süsteemi pakutavate andmeridade arvu

Eeldused:

- Kasutaja on sisse logitud administraatori kasutajaga
- Kohalikus arvutis on olemas allalaetud fail “Andmefail1.csv”

Sammud:

- Lugeda mõne tekstitöötluse rakendusega (nt Notepad++) “Andmefail1.csv” ridade arv.
- Tõmmata alla sama anduri andmed otse Cumulocity API kaudu järgneva päringuga (Täpsem dokumentatsioon Cumulocity API alajaotises “Retrieve all measurements”)

```
curl --location
'[TENANT_URL]/measurement/measurements?dateFrom=2023-05-02T07%3
A00%3A00.000Z&dateTo=2023-05-03T07%3A00%3A00.000Z&source=150
&valueFragmentType=TSu1' \
--header 'Accept: text/csv' \
--header 'Authorization: Basic [API_TOKEN]'
```

Eeldatav tulemus:

- Failis “Andmefail1.csv” on 94 rida
- Cumulocity API tagastas 94 rida

Tegelik tulemus:

- Saavutati eeldatav tulemus

4.3.2 Järgnevad sammud

Eelnevas peatükis kirjeldati lahenduses hetkel olemasolevaid võimekusi, mida on võimalik kasutada läbi veebipõhise haldusrakenduse. Loodud lahenduse rakendusliidese kaudu on aga võimalik kasutada rohkem funktsionaalsusi. Lahenduse API on välja toodud c8y-core-service teenuse dokumentatsioonis. Näiteks süsteemi saab uut kasutajat luua ning olemasolevat kasutajat kustutada vaid API abil. API kaudu on võimalik hallata ka lahenduses olevaid gruppe, mille abil saab luua uue grupi juba lahendusse imporditud andurite põhjal, loodud gruppidesse lisada juurde andureid ning ka gruppe kustutada. Läbi haldusrakenduse on võimalik hetkel ainult vaadata olemasolevaid gruppe ning nendes olevate andurite mõõtetulemusi alla laadida. Samuti on võimalik läbi API luua käsitsi andureid. Nende puhul eeldatakse, et kasutaja sisestab sellise anduri andmed, mis on ka Cumulocity platvormil olemas. Neid võimekusi ei jõudnud autor süsteemi luua ning jäävad järgnevateks sammudeks mida arendada.

Autor ei jõudnud luua ka korraliku integratsiooni CKANi põhise süsteemiga. Süsteemi testimisel kasutati selle vaikeversiooni, milles ei olnud veel täpsemat eesliidest otsimiseks näiteks andmete ajavahemike järgi. Siiski edastatakse need andmed üleslaadimisel CKANi ning nende järgi otsida sisseehitatud tekstipõhise otsinguga. Failid laetakse hetkel CKANi üles kasutades jäika struktuuri ehk kõik sama väärtusfragmendiga failid lisatakse ühte gruppi ning iga faili jaoks tehakse uus andmestik (*dataset*), mille alla lisatakse ressurss üleslaetava

failiga. Iga faili jaoks luuakse uus andmestik, kuna vaikeseadistusega toetab CKAN rohkete metaandmete lisamist vaid andmestikule. Metaandmed võetakse hetkel loodud süsteemist andurite küljest. Hetkel pole näiteks võimalik seadistada üleslaetavale failile lisa metaandmeid, mille tugi oleks tulevikuarendusena hea lisada.

Kokkuvõte

Töö esimeses peatükis anti ülevaade Cumulocity IoT platvormist, selle API toimimisest ning TÜ Delta keskuse Cumulocity andmemudelist, samuti kirjeldati olemasolevat lahendust ning selle mitte kasutamise põhjuseid. Töö teises peatükis käsitleti loodava lahenduse funktsionaalseid ja mittefunktsionaalseid nõudeid ning kasutatud tehnoloogiaid ja põhjendati nende valikut. Kolmandas peatükis anti ülevaade loodud lahenduse arhitektuurist ning mõningatest probleemidest andmete allalaadimisel. Viimases peatükis kirjeldati valminud lahenduse komponente ning arutleti lahenduse valideerimise ning järgnevate sammude üle.

Bakalaureusetöö eesmärgiks oli luua Cumulocity platvormil olevate andurite andmete eksportimise ning avalikustamise lahendus, sest sellel hetkel ei olnud mugavat viisi Cumulocity platvormilt andurite mõõtetulemuste jagamiseks. Seetõttu loodi veebipõhine haldussüsteem koos tagasüsteemidega, millega saab importida andurid ja andurite grupid Cumulocity platvormilt süsteemi. Seejärel saab otsida andurite ja gruppide seast vajalikud andurid. Soovitud andurite mõõtetulemused saab alla laadida valitud ajaperioodist, samuti saab avalikustada alla laaditud andurite andmeid süsteemi tavakasutajatele ning andmefaile võib soovi korral eksportida ka CKANi põhisesse andmeportaali. Ühel kasutajal on korraga võimalik alla laadida mitme erineva anduri andmeid. Seda on võimalik teha nii ühe korra kui ka automatiseerida perioodiliseks. Samuti loodi võimalus defineerida anduritele lisametaandmeid ning näha ja kasutada Cumulocity platvormil olevaid andurite gruppe.

Järgnevate arendustöödena on vaja täiustada lahenduse andmete avalikustamise funktsionaalsust ning integratsiooni CKANi ning muude väliste süsteemidega, kuna autor jõudis CKANi funktsionaalsustest tegeleda vaid vaikesätetega. Samuti on automaattestide katvus hetkel veel vähene ning seda tuleb edasiarenduste käigus suurendada. Edaspidi on võimalik haldusrakendusse lisada ka need võimekused, mis on hetkel saadaval vaid loodud lahenduse rakendusliidese kaudu.

Viidatud kirjandus

[1] Hanes, D., Salgueiro, G., Grossetete, P., Barton, R., Henry J. IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things. Cisco Press. 2017. https://nibmehub.com/opac-service/pdf/read/IoT%20Fundamentals.pdf

[2] Euroopa Parlamendi ja nõukogu direktiiv (EL) 2019/1024, avaandmete ja avaliku sektori valduses oleva teabe taaskasutamise kohta (uuesti sõnastatud). 20.06.2019. https://eur-lex.europa.eu/legal-content/ET/TXT/?uri=CELEX:32019L1024&qid=1683097809916 (30.04.2023)

[3] Cumulocity, Introduction to our IoT platform. https://cumulocity.com/guides/concepts/introduction/ (26.02.2023)

[4] BACnet. https://bacnet.org/ (16.04.2023)

[5] Cumulocity IoT's domain model. https://cumulocity.com/guides/concepts/domain-model/ (26.02.2023)

[6] Cumulocity IoT API. https://cumulocity.com/api/10.14.0/ (09.04.2023)

[7] InfluxDB. https://www.influxdata.com/ (16.04.2023)

[8] Lnenicka, Martin. An In-Depth Analysis of Open Data Portals as an Emerging Public E-Service. *World Academy of Science, Engineering and Technology International Journal of Humanities and Social Sciences.* 2015, Vol. 9, No. 2, pp. 592-593. doi.org/10.5281/zenodo.1099708 (25.04.2023)

[9] Lněnička, M., Machova, R., Volejníková, J., Linhartová, V., Knezackova, R. and Hub, M. Enhancing Transparency Through Open Government Data: the Case of Data Portals and Their Features and Capabilities. *Online Information Review*, 2021, Vol. 45, No. 6, pp. 1021-1038. https://doi-org.ezproxy.utlib.ut.ee/10.1108/OIR-05-2020-0204 (25.04.2023)

[10] Eesti avaandmete teabevärav. https://avaandmed.eesti.ee/ (07.12.2022)

[11] Eesti avaandmete teabevärav, Tartu Linnavalitsus. https://avaandmed.eesti.ee/information-holders/tartu-linnavalitsus (07.12.2022)

[12] Eesti avaandmete teabevärav, üldjuhend. https://avaandmed.eesti.ee/instructions/teabevarava-uldjuhend (07.12.2022)

[13] Tartu linna geoHUB. https://geohub.tartulv.ee/ (09.12.2022)

[14] Tartu linna geoHUB, Tehniline infrastruktuur https://geohub.tartulv.ee/pages/tehnotaristu (15.04.2023)

[15] CKAN. https://github.com/ckan/ckan (10.12.2022)

[16] CKAN, Showcase. https://ckan.org/showcase (10.12.2022)

[17] CKAN, Features. https://ckan.org/features (10.12.2022)

[18] CKAN, Docs. https://docs.ckan.org/en/2.9/contents.html (10.12.2022)

[19] Laius, S. Integratsiooniteek Cumulocity IoT platvormi ajalooliste andmete analüüsiks Pythonis, TÜ arvutiteaduse instituudi bakalaureusetöö. 2019. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=69682&year=2020 (04.12.2022)

[20] TypeScript. https://www.typescriptlang.org/ (26.04.2023)

[21] NestJS. https://docs.nestjs.com/ (26.04.2023)

[22] MongoDB. https://www.mongodb.com/ (27.04.2023)

[23] Mongoose. https://mongoosejs.com/ (27.04.2023

[24] RabbitMQ. https://www.rabbitmq.com/ (27.04.2023)

[25] Apache Kafka, Documentation. https://kafka.apache.org/documentation/ (29.04.2023)

[26] React. https://react.dev/ (29.04.2023)

[27] Stack Overflow Developer Survey. 2022. https://survey.stackoverflow.co/2022/#most-popular-technologies-webframe (29.04.2023)

[28] Refine. https://refine.dev/docs/ (29.04.2023)

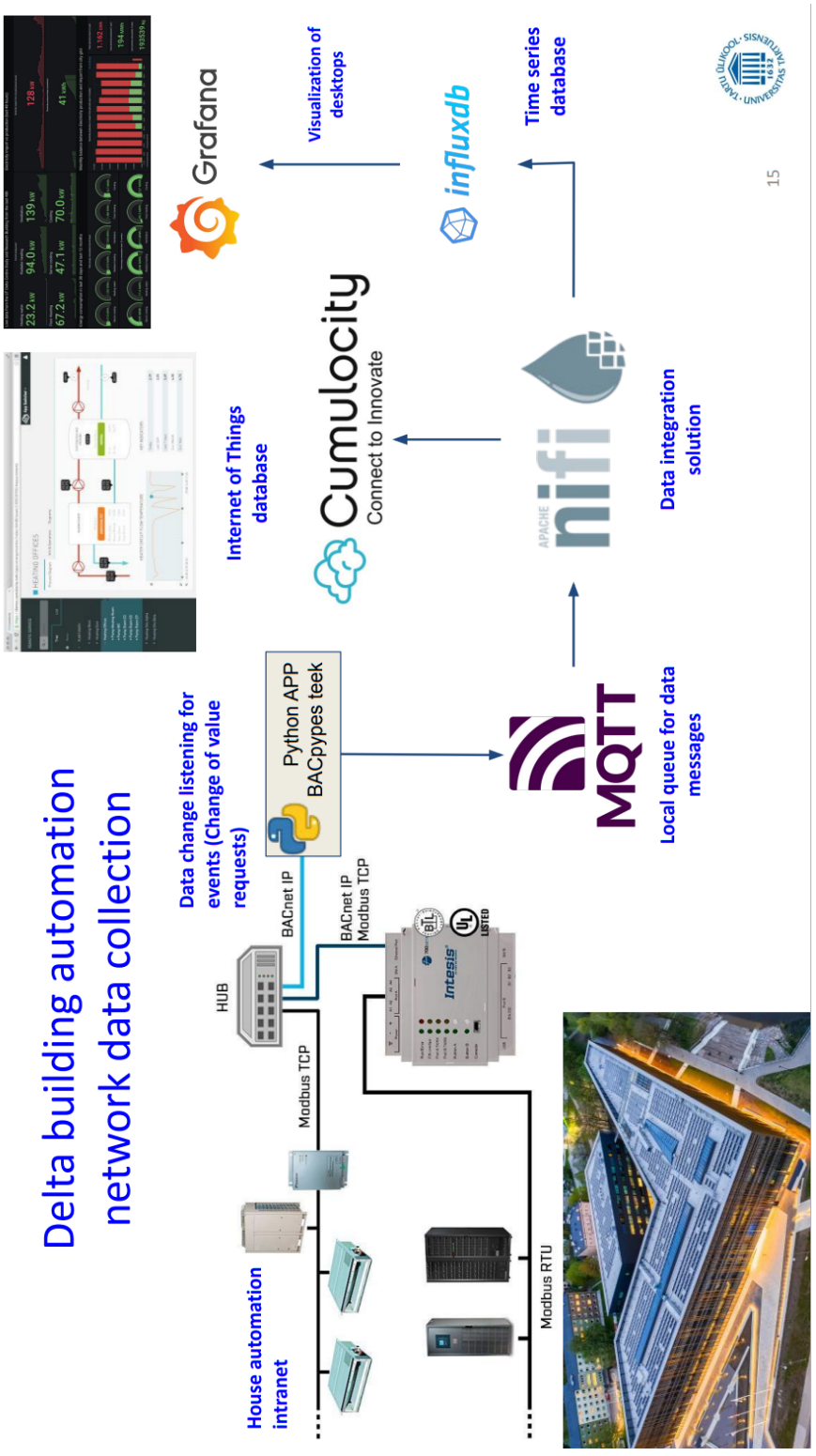
[29] MUI, Overview. https://mui.com/material-ui/getting-started/overview/ (29.04.2023)

[30] Newman, Sam. Building Microservices, Designing Fine-Grained Systems. Canada: O'Reilly Media, Inc. 2021, pp. 14-25.

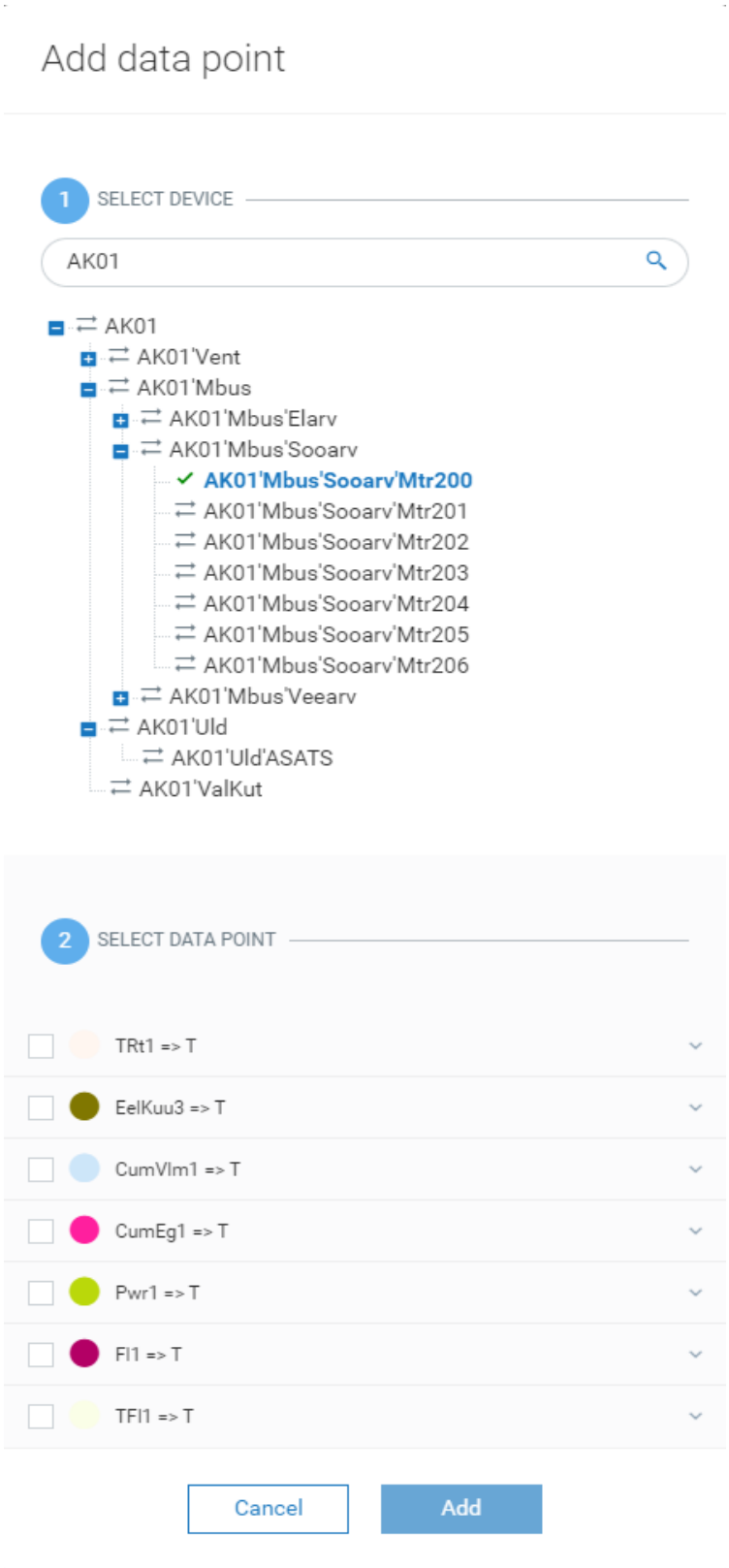
[31] MongoDB. Text Indexes. https://www.mongodb.com/docs/manual/core/index-text/ (05.05.2023)

Lisad

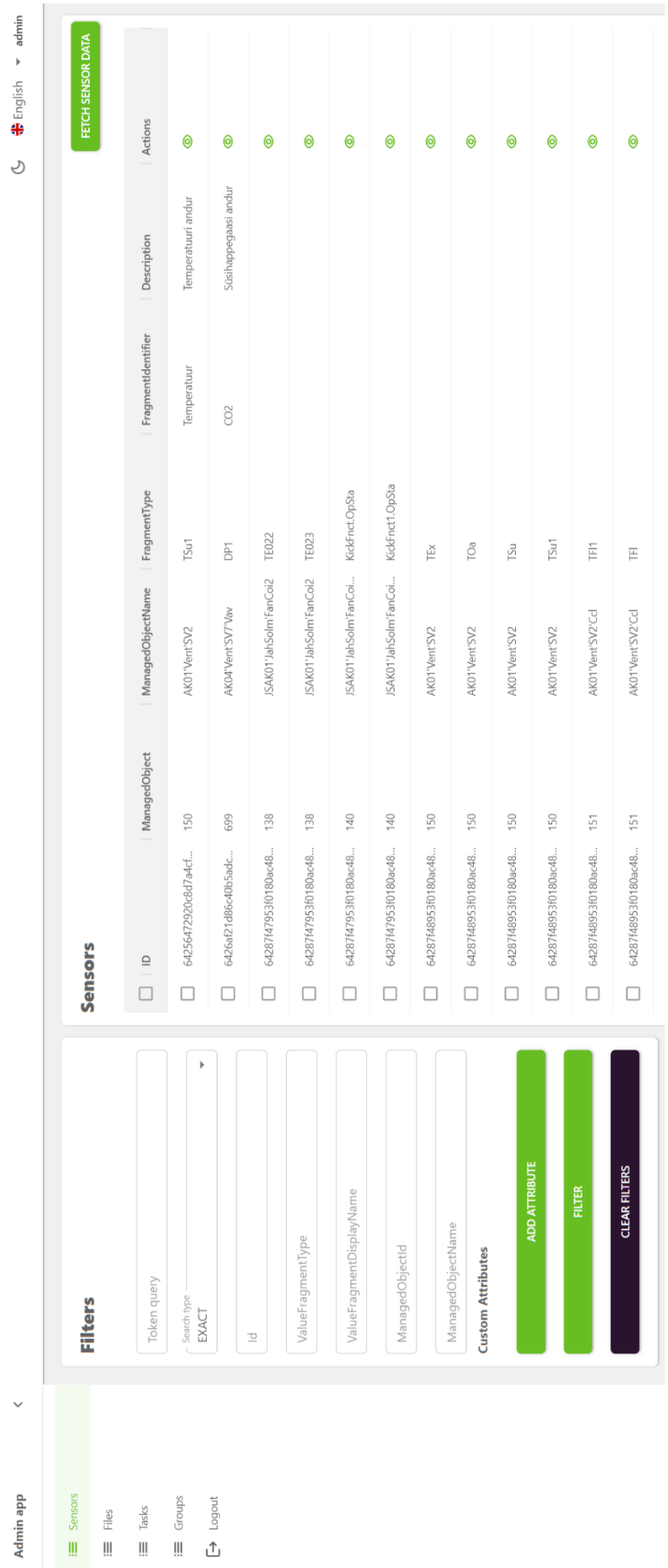
Lisa 1. Andmete liikumine TÜ Delta keskuses, seminar "TÜ Targa maja lahendused



Lisa 2. Delta keskuse andmepunktide avastamine Cumulocity kokpitis



Lisa 3. Haldusrakenduse avaleht



Lisa 4. Lähtekood

Valminud lahenduse lähtekood on saadaval Githubis aadressidel https://github.com/martenka/c8y-core-service, https://github.com/martenka/c8y-admin ja https://github.com/martenka/c8y-data-service . Kuna lahendus vajab testimiseks Cumulocity platvormi kasutajat, siis selle puudmisel tuleks ühendust võtta töö autoriga e-kirja teel marten.kahu@ut.ee või juhendaja dr Pelle Jakovitsiga pelle.jakovitsh@ut.ee .

Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Märten Kahu,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

   Lahendus avaandmete jagamiseks Cumulocity IoT platvormilt

   mille juhendaja on Pelle Jakovits,

   reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Märten Kahu

08.05.2023