TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Silver Laius

Integratsiooniteek Cumulocity IoT platvormi ajalooliste andmete analüüsiks Pythonis

Bakalaureusetöö (9 EAP)

Juhendaja: Jakob Mass, MSc

Tartu 2019

Integratsiooniteek Cumulocity IoT platvormi ajalooliste andmete analüüsiks Pythonis

Lühikokkuvõte:
Antud töös loodi Pythonis teek, millega saab Cumulocity IoT platvormi REST API kaudu ajaloolisi andmeid pärida. Teegi eesmärgiks on hõlbsustada ja kiiremaks teha Cumulocity'st andmete pärimist ning nende lihtsasti töödeldavale kujule viimist. Teegi loomisel oli fookus suurte andmemahtude pärimisel suure jõudlusega. Teeki kasutati 2019. aasta Metallica kontserti liiklusloendurite andmeanalüüsi andmete saamiseks Tartu linna Cumulocity platvormilt. Lisaks on töös välja toodud andmeanalüüsi töövoog ning tulemused, mis teid kaudu ning mis kellaajal kontserdikülastajad saabusid ning lahkusid Tartust.

Võtmesõnad:
Python, Pandas, Cumulocity, IoT

CERCS: P170 Arvutiteadus, süsteemid

Integration library for analyzing historical data from the Cumulocity IoT platform in Python

Abstract:
In this thesis, a library was created in Python, which can be used to query historical data from the REST API of the Cumulocity IoT platform. The purpose of the library is to make it easier and faster to retrieve data from Cumulocity and convert it into a easy-to-process format. When creating the library, the focus was on retrieving large amounts of data with the highest performance. The library was used to obtain traffic counter data for analyzing the 2019 Metallica concert. In addition, this thesis presents the data analysis workflow and the results, which roads and at what time the concert audience arrived at and left Tartu.

Keywords: Python, Pandas, Cumulocity, IoT

CERCS: P170 Computer science, systems

Sisukord

1 Sissejuhatus

Aina aktuaalsemaks muutuvad meedias globaalse soojenemise tagajärjel tekkinud probleemid ning taastuvate energiallikate eelistamine. Suurel määral on hakatud pöörduma lahenduste poole, mis kasutavad või toodavad taastuvat energiat. Evangelos Theodoridi jt. uurimustöös[1] on välja toodud, et infrastruktuur on muutunud linnastumise tagajärjel keerulisemaks ja selle efektiivsemaks toimimiseks vajab see pidevat haldust ja ülevaatust. Infrastruktuuri süsteemi kuuluvad näiteks transpordisüsteemid, veevõrgud ja elektrivõrgud. Lisatakse, et iga infrastruktuuri osa koosneb heterogeensetest pärismaailma seadmetest ja sensoritest, mis ei ole iseseisvalt koostöövõimelised. Lahenduseks tuuakse välja suurenev IoT (Asjade interneti) kasutuselevõtt, mis on toonud võimekuse teenuseid kombineerida ja nende vahel automaatselt andmeid jagada. Selgitatakse veel, et IoT keskmes on vahevara platvormid, mis aitavad vahendada andmeid seadmete ja rakenduste vahel, hallata ja konfigureerida seadmeid ning tagada kõige selle turvalisus.

Viimaste aastate jooksul on Euroopa Liidus käivitatud mitmeid IoT süsteemide realiseerimist toetavaid projekte [1]. Üheks neist on Euroopas ökoloogilise jalajälje vähendamiseks loodud projekt - SmartEnCity [2][3]. Projekti eesmärgiks on välja töötada universaalne ja süsteemne lähenemine viimaks linnade energianõudlus miinimumini ning minna üle taastuvate energialahenduste kasutamisele [3]. Programmi kolmeks majakalinnaks ehk linnadeks, kelle eeskujul süsteem välja töötatakse on Eestis Tartu, Taanis Sonderborg ning Hispaanias Vitoria-Gasteiz [2]. Tartu Linnavalitsus on projektiga liitumisest saati kogunud suurel hulgal andmeid erinevatelt seadmetelt - alates linnapiiril asuvatest liiklusloenduritest kuni linnas asetsevate mürasensoriteni [4]. Kogutud andmed salvestatakse Cumulocity IoT platvormile [5].

Cumulocity on pilves asuv graafilise veebiliidesega platvorm, mille eesmärgiks on pakkuda lihtsat ning kiiret IoT võimekusega seadmete ühendamist ja haldamist [6]. Toetatud on seadmete staatuse jälgimine, ajalooliste andmete kogumine nagu näiteks andurite mõõtmisväärtused, muutuste logimine ja alarmide salvestamine.

Cumulocity graafilise veebiliidese kaudu saab genereerida andmete visualiseerimiseks lihtsamaid graafikuid, kuid detailsemat süvenemist või eeltöötlust platvorm ei toeta. Platvormi veebiliides toetab andmete eksportimist, kuid maht on piiratud ning kogu protsess ajakulukas [7]. Cumulocity platvormi haldav firma, SoftwareAG, on arendajasõbralikuks lahenduseks loonud R keeles teegi, mis laseb kasutajal läbi REST API (*Representational State Transfer Application Programming Interface*) päringute küsida Cumulocity platvormilt ajaloolisi andmeid [8]. Pythonis Cumulocity'st ajalooliste andmete pärimiseks teeki ei eksisteeri.

Tänapäeval on Python tänu Pandase[1] teegile andmeteaduse valdkonnas R-i kõrval

[1] `https://pandas.pydata.org/`

levinuim vahend andmete töötluseks [9][10]. Lisaks Pandasele aitavad Pythoni populaarsusele kaasa ka teised statistika, masinõppe ja visualiseerimise jaoks mõeldud teegid, millest populaarseimad on välja toonud J. VanderPlas oma raamatus "Python data science handbook"[11].

Antud lõputöö eesmärk on luua Pythoni programmeerimiskeeles teek, mis teeks Cumulocity kogutud ajalooliste andmete kättesaamise võimalikult arendajasõbralikuks ja kiireks. Teegi loomisel on uuritud SoftwareAG cumulocityr teeki, mis on inspireerinud ka kasutajakogemuse loomist. Teek tehakse kättesaadavaks Pythoni paketihalduris PIP. Link koodirepositooriumile ja dokumentatsioonile on kättesaadav lisade peatükis.

Teegi arendusprotsess toimub paralleelselt Tartu linna kogutud andmeid analüüsides. Fookuses on 18. juulil 2019.a toimunud Metallica kontsert mille raames uuritakse milliseid maanteid mööda inimesed saabusid ja lahkusid kontserdilt. Uuringu tarbeks on Tartu Linnavalitsus jaganud ligipääsu Cumulocitys olevatele ajaloolistele liiklusloendurite andmetele. Need loendurid mõõdavad mööduva auto kiirust, pikkust, suunda, pikivahet eelnevalt möödunud autoga ning loendurist möödumise kellaaega. Andmete saamiseks kasutatakse jooksvalt valmivat teeki, analüüsiks ja töötluseks Pandast ning visualiseerimiseks Matplotlibi. Andmete pärimise kaudu valideeritakse teegi kasutajamugavust, vajalikku funktsionaalsust ja kiirust päriselulises situatsioonis.

Käesolev bakalaureusetöö koosneb viiest osast. Teises peatükis antakse ülevaade Cumulocity'st, Tark Tartu projektist, andmeanalüüsi levinud metoodikast ja seotud töödest. Kolmandas peatükis kirjeldatakse valminud Pythoni teegi nõudeid, otsuseid ning implementatsiooni. Neljandas peatükis tutvustatakse linna liiklusloendurite andmete analüüsi eesmärke, metoodikat ja tulemusi. Viiendas lõputöö peatükis diskuteeritakse teegis tehtud muudatuste ja teegi kasulikkuse üle. Kuues peatükk keskendub kokkuvõtte tegemisele ning tuleviku arenduste väljatoomisele.

2 Tausta ülevaade

Käesolev peatükk annab ülevaate lõputööga seotud tausta ning sarnaste lahenduste kohta.

2.1 IoT platvormid

H. Hejazi jt. poolt tehtud 2018. aasta uuringus[12] on IoT'd kirjeldatud kui seadmete, sensorite ja sõidukite võrgustikku, mis on kasutaja poolt jälgitav ja kontrollitav. IoT platvormi roll on pakkuda kasutajale süsteemset lahendust, mille külge seadmeid ühendada ning andmeid salvestada. Uuringus on mainitud, et igale platvormil olevale füüsilisele seadmele tekitatakse virtuaalne kohalolek ning seadme staatus, andmed ning konfiguratsioon on kasutajale lihtsasti kättesaadavad. Uuringus selgus, et kuna platvorme on mitmeid, leiab kasutaja enda jaoks sobiva platvormi vastavalt mudelile ja võimalikule funktsionaalsusele. Põhjalikum võrdlus erinevate platvormide toetatud funktsionaalsuste kohta on tehtud 2016. aasta uuringus[13] P. P. Ray poolt, kus selgus, et enim on levinud seadmete, andmete ja rakenduste haldusele keskendunud platvormid. Puudus on teadusele ja uurimisele keskendunud platvormidest, mis laseks teadusringkondadel paremini katseid läbi viia.

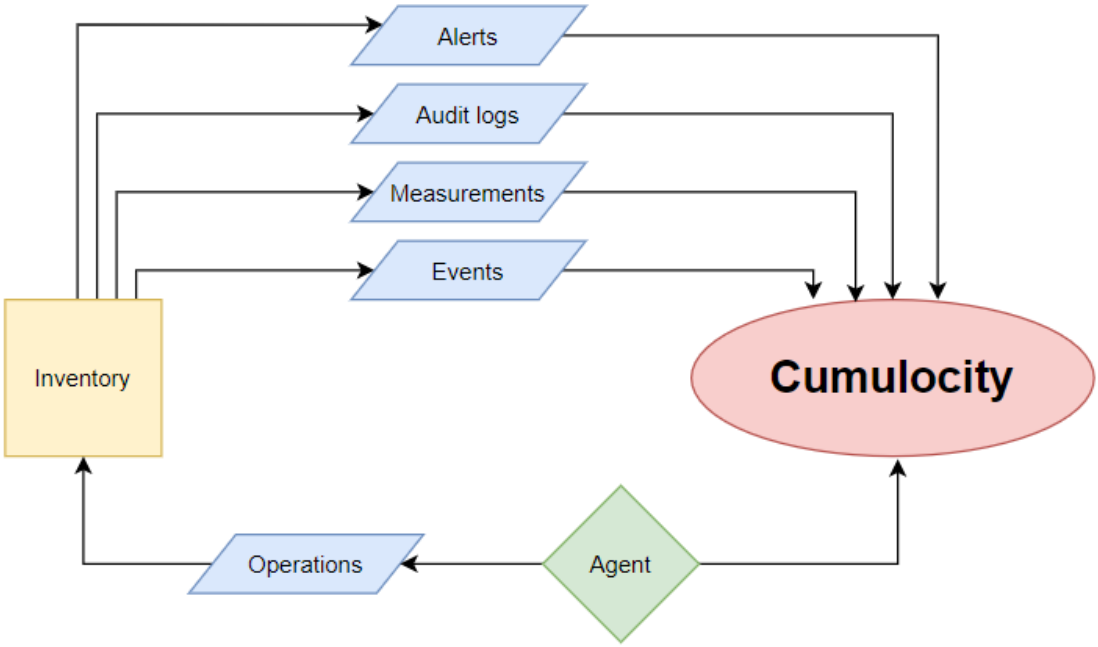
2.1.1 Cumulocity

Cumulocity [14] on pilves asuv IoT platvorm, kus iga seade ja salvestatud andmepakett on JSON (*JavaScript Object Notation*) kujul. Suhtlus seadmetega käib MQTT (*Message Queuing Telemetry Transport*) protokolli või HTTP REST API kaudu. Cumulocity andmemudeli eripäraks on seadme- ja andmestruktuuride defineerimine fragmentide abil. Igal seadmel on põhilised näitajad nagu id ja tüüp ning lisaks neile kohandatud fragmendid. Näiteks koodinäites 1 on kolmefaasiline elektrimõõduk, millel on määratud GPS koordinaadid. "c8y_Position", "c8y_ThreePhaseElectricitySensor" ja "c8y_Relay" on sellele seadmele kohandatud fragmendid, kus c8y on lühend, mis tähistab Cumulocity't.

```
{
    "id": "47635",
    "type": "elstermetering_AS220",
    "lastUpdated": "2010 11 13T18:28:36.000Z",
    "c8y_Position": {
        "alt": 67,
        "lng": 6.15173,
        "lat": 51.211977
    },
    "c8y_ThreePhaseElectricitySensor": {},
    "c8y_Relay": {
        "state": "CLOSED"
    }
}
```

Koodinäide 1. Cumulocity dokumentatsioonis [14] kirjeldatud fragment

Cumulocity dokumentatsiooni [14] kohaselt koosneb inventar kõigist hallatavatest seadmetest ning nendega seotud varadest. Inventaris oleva objekti poolt saadetud andmed on jagatud kas numbrilisteks mõõdeteks (edaspidi *measurements*) või sündmusteks (edaspidi *events*). Samuti saadavad seadmed hoiatusi (edaspidi *alerts*) ning turvalisuseteemalisi sündmusi, mida kutsutakse auditi logideks (edaspidi *audit logs*). Lisaks on võimalik seadmele jätta tööülesandeid (edaspidi *operations*). Seadmele jäetud *operations*'id jäävad Cumulocity platvormile ootama hetke, kuni seadme eest vastutav agent neid pärima ning seadmele edastama tuleb. Kogu andmevoog on ilustreeritud joonisel 1.



Joonis 1. Cumulocity andmemudel

Operations võib olla nii lihtne või raske kui seadme konfiguratsioon võimaldab. *Koodinäide 2* peal on *operation*'iks id-ga 42 seadme relee avamine.

```
{
    "deviceId": "42",
    "c8y_Relay": {
        "relayState": "OPEN"
    }
}
```

Koodinäide 2. Cumulocity dokumentatsioonis [14] kirjeldatud operation

Measurements tähistavad seadmete poolt regulaarselt saadetud näite ja statistikat, mis koosnevad näidu võtmise ajast, unikaalsetest identifikaatoritest ning fragmentidest. Igal *measurementil* on numbriline väärtus ja ühik. Koodinäide 3 annab ülevaate *measurement* JSON struktuurist.

```
{
    "time": "2011 01 02T03:04:00.000Z",
    "source": { "id": "1235", ... },
    "c8y_ThreePhaseElectricityMeasurement": {
```

```
            "A+": { "value": 435, "unit": "kWh" },
            "A ": { "value": 23, "unit": "kWh" },
            "P+": { "value": 657, "unit": "W" },
            "P ": { "value": 0, "unit": "W" },
            "A+:1": { "value": 123, "unit": "kWh" },
            "A :1": { "value": 2, "unit": "kWh" },
            "P+:1": { "value": 56, "unit": "W" },
            "P :1": { "value": 0, "unit": "W" },
            "A+:2": { "value": 231, "unit": "kWh" },
            "A :2": { "value": 23, "unit": "kWh" },
            "P+:2": { "value": 516, "unit": "W" },
            "P :2": { "value": 2, "unit": "W" },
            ...
        },
        ...
}
```

Koodinäide 3. Cumulocity dokumentatsioonis [14] kirjeldatud measurement

Events jaguneb kolmeks: *baas events*, *alarms* ning *audit logs*.

Baas events saadetakse mingisuguse ajendi peale. Näiteks kui auto möödub liiklusloendurist või seadme asukoht uueneb, nagu on toodud välja koodinäide 4 peal.

```
{
    "type": "c8y_LocationUpdate",
    "time": "2010-11-13T18:28:36.000Z",
    "text": "Location updated",
    "source": { "id": "47634", ... },
    "c8y_Position": {
        "alt": 67,
        "lng": 6.15173,
        "lat": 51.211977
    }
}
```

Koodinäide 4. Cumulocity dokumentatsioonis [14] kirjeldatud baas event

Alarms on *events*'i alamtüüp, mis vajab inimese sekkumist. Näiteks kui mõni seade ei saa teisega ühendust nagu koodinäites 5 või mõne masina temperatuur tõuseb liiga kõrgele.

```
{
    "type": "c8y_UnavailabilityAlarm",
    "time": "2010-11-13T19:28:36.000Z",
```

```
    "text": "No communication with device since 2013 11 05T15
       :23:55.284+01:00",
    "status": "CLEARED",
    "severity": "MINOR",
    "source": { "id": "47633", ... },
    "history": {
        "auditRecords": [ {
            "activity": "Alarm updated",
            "application": "devicemanagement",
            "user": "vvirtanen",
            "time": "2013 11 05T16:37:48.494+01:00",
            "changes": [ {
                "attribute": "status",
                "newValue": "CLEARED",
                "previousValue": "ACTIVE",
                "type": "com.cumulocity.model.event.
                   CumulocityAlarmStatuses"
            } ],
            ...
        } ]
        ...
    }
    ...
}
```

Koodinäide 5. Cumulocity dokumentatsioonis [14] kirjeldatud alarm

Audit logs on turvalisusega seotud *events* , mis salvestatakse hilisemaks auditeerimiseks. Näiteks Koodinäite 6 puhul on selleks kasutaja ebaõnnestunud sisselogimine.

```
{
    "type": "c8y_SecurityEvent",
    "time": "2010 11 13T18:28:36.000Z",
    "text": "Gateway login failed",
    "user": "vvirtanen",
    "application": "Resort energy management",
    "activity": "login",
    "severity": "MINOR",
    "source": { "id": "47633", ... },
    ...
}
```

Koodinäide 6. Cumulocity dokumentatsioonis [14] kirjeldatud audit log

2.2 Tark Tartu

Tartu on seotud SmartEnCity projektiga alates aastast 2016 ning eeldatavasti saavad planeeritud tegevused tehtud aastaks 2021 [3]. Kavandatud tegevusteks [4] on hoonete renoveerimine, transpordi viimine rohelise energia peale ja tänavavalgustuse energiatarbimise vähendamine kasutades kontrollsüsteemi ning sensoreid. Kogu projekti toimumise vältel renoveeritakse 22 *hruštšovka* tüüpi hoonet ning neile paigaldatakse päikesepaneelid [3]. Rohelise energia peale üleminekul on Tartu linna jaoks oluline viia aastal 2019 bussiliiklus 100% naturaalse gaasi peale [4]. Lisaks soetatakse 750 elektri- ja tavaratast ning nende jaoks paigaldatakse linna erinevatesse puntkidesse 65 parkimisplatsi. Smart Tartu projekti andmeid hoiustab linn Telia poolt hallatavale Cumulocity IoT platvormile

Kavandatud tegevuste elluviimiseks ning analüüsimiseks kogutakse andmeid renoveeritud kortermajade elektri-, vee- ning soojustarbimise kohta. Suur osatähtsus on ka transpordil ja seetõttu kogutakse Tarkadelt ratastelt, uutelt gaasibussidelt, sisse- ja väljasõiduteede sensoritelt ning elektriautode laadimiskohtadelt andmeid. Andmeid lisandub veel tänavavalgustite töötamise kohta ja palju muud [4].

2.2.1 SmartEnCity

Baskimaa Ülikooli 2019. aasta artiklis [15] tuuakse täpsemalt välja kuidas viimasel sajandil fossiilkütuste põletamise resultaadiks on ettenägematute tagajärgedega kliimamuutused. Euroopa Komisjon, eesmärgiga 2030. aastaks vähendada kasvuhoonegaaside tootmist 40%, on toetanud mitmeid projekte, mis tegelevad kasvuhoonegaaside vähendamisega. SmartEnCity [2] on Euroopa Liidu Horizon 2020 teadus- ning innovatsiooniprogrammi poolt rahastatud projekt, mille eesmärk on muuta Euroopa linnad jätkusuutlikumaks, targemaks ja ressursitõhusamaks. Projekti üheks alustalaks on Smart Zero Carbon City kontseptsioon, kus linna ökoloogiline jalajälg on viidud miinimumini ning energia tuleb täielikult taastuvatest energiaallikatest [2]. Tarkade linnade kontseptsioon keskendub enamasti innovatiivse rohelise tehnoloogia kasutuselevõtmisele [3].

2.3 Andmeanalüüs

Veebilehe coordinationtoolkit.org poolt avaldatud õpetuse [16] kohaselt jaguneb andmeanalüüsi protsess kolmeks osaks: 1) ettevalmistus, 2) andmetest arusaamine, 3) tulemuste esitlemine. Järgnevad seletused põhinevad samuti eelnevalt mainitud õpetusel [16].

Andmeanalüüsi ettevalmistumise esimeseks etapiks on saada ligipääs analüüsitavatele andmetele ning viia need lihtsasti töödeldavale kujule nagu näiteks andmebaasi või exceli read. Seda eesmärki asub täitma loodav teek liiklusloendurite andmeanalüüsi jaoks, andes kasutajale ligipääsu platvormil olevatele andmetele ning viies need analüüsiks Pandase DataFrame kujule. Järgmiseks peab aru saama, mida iga veerg tähistab, mis ühikutes see on ning kas kõik olulised veerud, nagu näiteks kuupäevad, on olemas. Vajadusel tuleb puudulikke või erineval kujul olevaid andmeridasid töödelda, kuni neid oleks võimalik agregeerida.

Andmetest arusaamiseks peab neid iteratiivselt transformeerima, läbi vaatama, analüüsima ja tõlgendama. Kõige tavapärasemateks andmete transformeerimise viisideks on muutujate ühendamine, veergude mitmeteks muutujateks lahti kodeerimine ning muutujate standardiseerimine ja normaliseerimine. Pideval läbivaatlusel keskendutakse andmetel põhineva loo rääkimisele või probleemi lahendamisele. Kuna andmed enda eest ei räägi, vajab analüüs ja tõlgendamine inimese kogemust ning mõtlemist. Andmeanalüütikud peavad võrdlema andmeid erinevate karakteristikate põhjal ning tegema järeldusi sarnasuste ja erinevuste osas.

Tulemuste esitlemine on sama oluline kui kogu eelnev protsess ning nõuab ausust ja läbimõtlemist. Siin keskendutakse uute ja oluliste teadmiste edasiandmisele läbi graafikute ja loo rääkimise. Oluline on tunda oma publikut ning rääkida just sellest, mis otsustajatele korda läheb. Peale esitlemist ei saa unustada ka andmete säilitamist, mis on oluline tulevikus uute analüüside tegemiseks ning võrdluseks.

2.4 Seotud tööd

See peatükk annab ülevaate hetkel olemasolevatest lahendustest ning nende puudustest ja eelistest.

2.4.1 cumulocityr

Cumulocityr on SoftwareAG poolt hallatud R keelele mõeldud teek, mida saab kasutatada Cumulocity API-ga suhtlemiseks [8]. Toetatud on *events*, *measurements* ja seadmete küsimine, kus filtreerimiseks kasutatakse kuupäeva ning seadme id parameetrid [8]. Cumulocityr klient toetab ka loogilist parameetrit PARSE_JSON, mille tõeseks märkimisel parsitakse Cumulocity platvormilt saadud JSON struktuur andmeridadeks. Teek tehti kasutajatele esmakordselt kättesaadavaks 22. oktoober 2019.a. Teegi puudusteks on, et see ei võimaldata kasutajal üle 2000 rea korraga küsida ning kasutaja peab suure andmekoguse jaoks ise oma tsükleid ja andmete agregeerimist tegema. Lisaks ei ole toetatud paljud olulised filtreerimise parameetrid, mida Cumulocity REST API pakub.

2.4.2 Cumulocity graafiline veebiliides

Cumulocity veebiliides toetab andmete .csv või .xlsx kujul eksportimist [7]. Andmeid saab eksportida maksimaalselt 1 miljon rida korraga ning need saadetakse kasutajale emaili teel [7]. Dokumentatsiooni kohaselt peab kasutaja enne eksportimist valima millistelt seadmetelt ning milliseid *measuremets*'e ja *events*'e ta soovib saada. Lisaks peab täpsustama ajavahemiku. Töö autoril ei õnnestunud dokumentatsiooni järgides Tartu Linna Cumulocity kliendist päringut tehes ühtegi sündmust ega mõõtu kätte saada. Sama kogemus on olnud eelnevalt ka Tartu Linnavalitsusel.

3 Cumulocity ühenduse teek

See peatükk kirjeldab Pythonis teegi loomist ajalooliste andmete pärimiseks Cumulocity REST API käest. Tutvustatakse nõudeid ja kitsendusi, arendusprotsessi ning päringute optimiseerimist.

3.1 Nõuded ja kitsendused

Eesmärgiks on luua teek Pythonis, mis lihtsustaks Cumulocity platvormilt ajalooliste andmete küsimist. Keskendutakse andmete pärimisele, kus fookus on suurtel kogustel (n > 100 000) ning selle jõudlusel. Inspiratsiooniks on võetud cumulocityr teegi poolt kasutatavad funktsiooni nimed ning kasutajakogemus. Juurde on lisatud kõik API poolt toetatud filtreerimisparameetrid ning kiiruse optimeerimine. Loodav teek peab toetama järgnevat funktsionaalsust:

- Peab saama küsida *measurements*'idel põhinevaid andmeid.
- Peab saama küsida *events*'idel põhinevaid andmeid.
- Peab saama küsida seadmetepõhiseid andmeid.
- Peab saama täpsustada ajavahemiku alguse, kust andmeid soovitakse saada.
- Peab saama täpsustada ajavahemiku lõpu, kust andmeid soovitakse saada.
- Ajavahemikud peaksid toetama kõiki tuntud kuupäeva formaate.
- Peab saama küsida andmeid konkreetse tüübi või anduri id järgi.
- Peab saama küsida informatsiooni andurite kohta, mis on konkreetse mõõte või sündmusega seotud.
- Teek peaks olema jõudlusele optimeeritud.
- Teek peab Cumulocity poolt tagastatud JSON objektidest looma kasutajale Pandase Dataframe.

Teek luuakse toetama /measurements, /events ning /managedObjects URL-ide (*Uniform Resource Locator*) pealt andmete pärimist. Nende URL-ide pealt toetab Cumulocity API ka POST päringute saatmist ehk andmete juurde lisamist või seadmete konfigureerimist, mis ei mahu selle töö skoopi. Lisaks ei mahu töö skoopi ka teiste URL-ide pealt päringute tegemine nagu näiteks /audit, /alarm ja /application.

3.2 Päringute tegemine

Cumulocity REST API[17] lubab kasutajal pärida andmeid tema Cumulocity kliendilt kasutades HTTP päringuid. Kõik päringute poolt tagastatud tulemused on JSON kujul.

Päring koosneb kahest osast: päisetest ning URL-ist. Turvalisuse jaoks on iga Cumulocity päringuga vaja kaasa lisada volituse päised. Koodinäites 7 on näidatud päiste genereerimine Pythonis oleva teegi base64'ga.

```
from base64 import b64encode

credentials = b64encode(bytes(username + ':' + password, "utf 8")).
    decode("ascii")
headers = { 'Authorization' : 'Basic %s' %  credentials }
```

Koodinäide 7. Volituse päiste loomise näide

Lisaks on dokumentatsiooni kohaselt toetatud ka oAuth ning JWT (*JSON Web Token*) volitamine. Teine oluline osa päringust on URL, mille struktuur Cumulocity platvormil näeb välja järgnev: <KLIENDI_URL>/<KATEGOORIA>/<PARAMEETRID>. Cumulocity lehel [18] on viide Postmani collectionile, kus on erinevad päringute näidised. Postman on kasutajaliidesega rakendus, mida kasutatakse erinevate HTTP päringute tegemiseks ja testimiseks.

Kliendi URL'iks on aadress kus Cumulocity klient asub, nagu näiteks tartu.cumulocity.com. Kategooriaks võib lisaks teistele olla näiteks *measurements*, *events* või *devices*. Parameetriteks on erinevad argumendid, mis lasevad kas andmeid filtreerida tüüpide ja kuupäevade järgi või seadistavad teisi olulisi paremeetreid nagu saadetav andmete kogus.

Kõik päringute tulemused on Cumulocity poolt jagatud väiksemateks osadeks, mida kutsutakse lehekülgedeks.

Iga edukas *events* päringu tulemus on koodinäites 8 väljatoodud kujul JSON objekt.

```
{
  "self" : "...",
  "prev" : "http://...?pageSize=5&Page=1",
  "next" : "http://...?pageSize=5&Page=3",
  "statistics" : {
    "totalPages" : 7,
    "pageSize" : 5,
    "currentPage" : 2
  },
  "events" : [
    ...
  ],
}
```

Koodinäide 8. Cumulocity REST API päringu tulemuse näide dokumentatsioonist [17]

'Self', 'prev' ja 'next' selles objektis on URL'id, kus 'self' tähistab sellele konkreetsele päringule vastavat URL-i, 'next' sellele päringule vastavat järgmise lehekülje URL'i ning 'prev' päringule vastavat eelneva lehekülje URL'i. Statistics annab kasutajale ülevaate mitu lehekülge kogu päringu tulemustes on kokku, mitu rida näidatakse igal leheküljel ning mitmendal leheküljel kasutaja hetkel on. *Events* selles objektis on massiiv, mis sisaldab endas parameetritele vastavaid JSON kujul andmeid. Kuna "pageSize" on 5, on ka selles listis viis elementi.

Kuna Cumulocity kasutab lehekülgjestamise süsteemi, muutuvad päringud suurte andmehulkade puhul aeglaseks. Seda põhjusel, et andmehulga kasvades peab andmebaas hakkama vahele jätma aina suuremaid hulkasid, et järgmist lehekülge saata. Näitena võib võtta situatsiooni kus kasutaja soovib 1 kuu andmeid olles seadnud lehekülje suuruseks 2000 ning talle tuleb tagastada 100 000 andmerida. See tähendab, et Cumulocity jagab need 100 000 rida viiekümneks erinevaks leheküljeks ning kasutaja peab neid hakkama eraldi pärima. Probleemiks on see, et alates näiteks 46st leheküljest peab andmebaas vahele jätma 90 000 rida ning saatma kasutajale read 90 001 - 92 000, siis vahele jätma 92 000 rida ning saatma kasutajale read 92 001 - 94 000 jne.

Lahenduseks on esialgne 1 kuu vahemikuga päring asendada nelja 1 nädala pikkuse vahemikuga päringuga. Eeldusel, et andmete tihedus on ühtlane terve perioodi vältel, on igas päringus keskmiselt 25 000 rida. Sellise lähenemise eelis on, et kunagi ei pea andmebaas järgmiste andmete kättesaamiseks vahele jätma rohkem kui 24 000 rida. Miinuseks on see, et kui on soov hiljem kogu andmehulka korraga kasutada, peab nende nelja päringu tulemused uuesti ühte andmestruktuuri kokku paigutama.

Measurements päringu tegemisel on toetatud järgnevad URL parameetrid:

- dateFrom - alguskuupäev

- dateTo - lõpukuupäev
- source - seadme id, mis selle measurementi salvestas
- measurementId - measurementi enda id
- type - measurementi tüüp
- valueFragmentType - fragmendi tüüp koos väärtusega
- valueFragmentSeries - fragmendi seeria koos väärtusega

Events päringu tegemisel on toetatud järgnevad URL parameetrid:

- dateFrom - alguskuupäev
- dateTo - lõpukuupäev
- source - seadme id, mis selle evendi salvestas
- type - evendi tüüp
- fragmentType - fragmendi tüüp

Seadmedepõhise info päringu tegemiseks on toetatud järgnevad URL parameetrid:

- deviceType - seadme tüüp
- fragmentType - fragmendi tüüp
- ids - massiiv seadmete id'dest
- text - seadme külge lisatud tekst
- query - Cumulocity enda päringute keeles [19] sõnastatud alampäring

3.3 Vastuse töötlemine

Selleks, et saada JSON struktuurist DataFrame, peab selle struktuuri tegema ühedimensiooniliseks. Koodinäites 9 on välja toodud *baas event*, mis kirjeldab asukoha uuendamist.

```
{
    "type": "c8y_LocationUpdate",
    "time": "2010 11 13T18:28:36.000Z",
    "text": "Location updated",
    "source": { "id": "47634" },
    "c8y_Position": {
        "alt": 67,
        "lng": 6.15173,
        "lat": 51.211977
    }
}
```

Koodinäide 9. Andmed mitmedimensioonilisel kujul

Kogu struktuur on ühedimensiooniline peale "c8y_Position" ja "source" fragmentide, milles on omakorda võtmed ja väärtused sees. Pandase DataFrame loomise poolt toetatud struktuuri saamiseks saab liita fragmendis sisalduvale võtme nimele eesliiteks fragmendi nime. Loetavuse jaoks lisatakse nende kahe sõna vahele eraldaja. *Koodinäide 10* puhul on selleks "_".

```
{
    "type": "c8y_LocationUpdate",
    "time": "2010 11 13T18:28:36.000Z",
    "text": "Location updated",
    "source_id": "47634",
    "c8y_Position_alt": 67,
    "c8y_Position_lng": 6.15173,
    "c8y_Position_lat": 51.211977

}
```

Koodinäide 10. Ühedimensiooniliseks viidud andmed

3.4 Disainiotsused ja kasutatud teegid

Teegi loomisel keskenduti ebavajaliku keerukuse vältimiseks sellele, et sõltuda võimalikult vähe teekidest, mis pole Pythoni programmeerimiskeelega kaasas. Cumulocity REST API päringute jaoks on kasutatud Pythoni HTTPSConnection[2] teeki, mis on kõige läbipaistvam lahendus päringute tegemiseks ning neile päiste lisamiseks. Volitus päiste kodeerimiseks kasutatakse samuti Pythoni enda b64encode[3] teeki. Kuna kogu

[2] https://docs.python.org/3/library/http.client.html
[3] https://docs.python.org/3/library/base64.html

andmepakett tuleb Cumulocity'st JSON sõne kujul, on tarvis JSON[4] teeki, mis võimaldab tagastagava JSON sõne Pythoni sõnastikuks parsida. Suurte andmemahtude tõttu on oluline ka kasutajale tagasisidet pakkuda kui kaugel allalaetav andmete kogum on. Selleks kasutatakse sys[5] teegi stdout.write ja stdout.flush meetodeid, et kasutaja konsoolis progressi riba ja infot näidata ilma, et iga kord peaks uut rida printima. Kuna Cumulocity aksepteerib ainult ISO 8601 formaadis kuupäevasid, kasutatakse kasutaja sisestatud kuupäevade parsimiseks ja ISO 8601 formaati teisendamiseks dateutil[6] teegi parser funktsiooni ning ajatsooni määramiseks tz funktsiooni. Selleks, et päringuid saaks optimeerida dünaamiliselt, on päringute optimeerimiseks kasutatud datetime[7] teegi timedelta parameetrit, kus kasutaja saab ise valida väärtuse. Viimaks on kasutatud Pandase teeki, et kasutajale saadetud andmetest dataframe tekitada.

3.5 Teegi kasutamine

Kogu loodud teegi funktsionaalsus on koondatud klassi nimega CumulocityConnection. Klassi kasutamiseks peab kasutaja esialgu looma klassi isendi oma cumulocity kliendile vastava infoga. Koodinäide 11 illustreerib seda protsessi Pythonis.

```
username = "kasutajanimi"
password = "parool"
url = "tenant.cumulocity.com"

connection = CumulocityConnection(url, username, password)
```

Koodinäide 11. Teegiga ühenduse loomise näide

Toetatud on koodinäites 12 välja toodud meetodid *measurements*'ide, *events*'ide ning seadmete info küsimiseks.

```
measurement_data = connection.get_measurements(
            value_fragment_type="c8y_YourMetric",
            date_from="07 11 19  03:00:00",
            date_to="07 13 19  04:00:00",
        )

event_data = connection.get_events(
            device_id="194442191",
            date_from="07 30 19  12:00:00",
```

[4] https://docs.python.org/3/library/json.html
[5] https://docs.python.org/3/library/sys.html
[6] https://dateutil.readthedocs.io/en/stable/
[7] https://docs.python.org/3/library/datetime.html

```
        date_to="07 30 19  13:00:00",
    )

ids=[117925736,117925737,117925738,117925739]
device_data = connection.get_devices(ids=ids, page_size=1000)
```

Koodinäide 12. Teegiga andmete pärimise näited

Toetatud on kõigi Cumulocity REST API poolt pakutavad parameetrid, mis toodi välja alampeatükis 3.2. Lisaks on implementeeritud kõigi laialt levinud kuupäevaformaatide konverteerimine cumulocity'le sobivale kujule.

```
Downloading events batch  1 of 244:
[====================] 100.00%
Downloading events batch  2 of 244:
[====================] 100.00%
Downloading events batch  3 of 244:
[====================] 100.00%
Downloading events batch  4 of 244:
[====================] 100.00%
Downloading events batch  5 of 244:
[===========          ] 55.56%
```

Joonis 2. Kasutajale progressi näitamine konsoolis

Peale päringu alustamist hakatakse kasutajale konsooli printima infot progressi kohta nagu näidatud joonisel 2. *Batch* (partii) tähistab mitmendat lõiku ajavahemikust hetkel päritakse ning protsendid tähistavad lehekülgi. Kui kokku on etteantud ajavahemikus 5 lehekülge andmeid ning esimene lehekülg saab päritud, näidatakse progressi 20%, ehk 1/5 kui teine lehekülg saab päritud näidatakse 40% ehk 2/5 jne. Optimiseerimise jaoks päringu ajavahemikku jagamist väiksemateks lõikudeks kirjeldatakse täpsemalt järgmises optimiseerimise alampeatükis.

Alati ei pruugi ühendus Cumulocity'ga õnnestuda või päringute parameetrid ei pruugi olla paikapidavad. Sellisel juhul kuvatakse kasutajale välja tema tehtud päringu URL, Cumulocity poolt tagastatud HTTP staatuse kood ning tagastatud errori sisu. See annab kasutajale silumiseks parema ülevaate, mis valesti võis minna.

3.6 Optimiseerimine

Päringute optimiseerimiseks on *measurements*'i ja *events*'i pärimise funktsioonidele võimalik anda kaasa *timedelta* väärtus. *Timedelta* on datetime teegi üks funktsioonidest,

mis toetab järgnevaid parameetreid: days, seconds, microseconds, milliseconds, minutes, hours, weeks. Selle parameeteri kaasa andmine lõikab esialgse dateFrom ja dateTo kuupäevade vahemiku timedelta parameetrile etteantud pikkustele vastavateks lõikudeks. Näiteks esialgselt etteantud vahemik 22.01.2019 - 25.01.2019 tehakse timedelta(days=1) korral kolmeks erinevaks päringuks: 22.01.2019 - 23.01.2019, 23.01.2019 - 24.01.2019, 24.01.2019 - 25.01.2019. See aitab vältida liialt suureks kasvavat vahele jäetavate andmete hulka ning sellest tulenevaid aeglaseid päringuid. Päringute tulemused lisatakse kokku üheks listiks.

Koodinäites 13 näidatud päringut katsetati kümnel korral erinevate *timedelta* väärtustega ning tulemused salvestati. Kümne korra tulemustest leiti keskmine ning see kajastub järgnevates graafikutes. See päring tagastab Tartu linna Cumulocity'st 985 504 liiklusloendurite salvestatud *events*'i. Keskmiselt on andmetihetus 95 000 rida päeva kohta, kus põhiosa andmetest on salvestatud päevasel ajal. Lehekülje suuruseks on valitud 2000 rida.

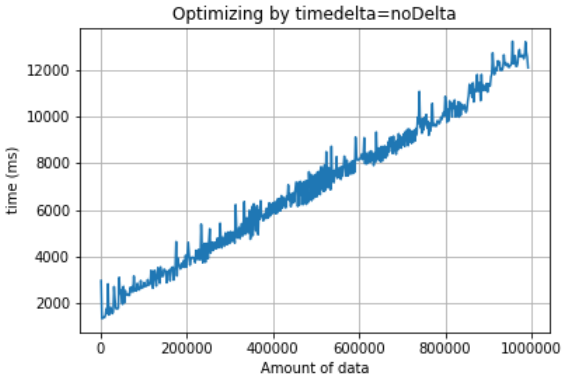
```
from datetime import timedelta

vehicle_data = connection.get_events(
            event_type="avc_VehicleEvent",
            date_from="07 01 19 00:00:00",
            date_to="07 11 19 12:00:00",
            page_size=2000,
            timedelta=muutuv_delta
        )
```

Koodinäide 13. Optimiseerimise testimiseks kasutatud päring

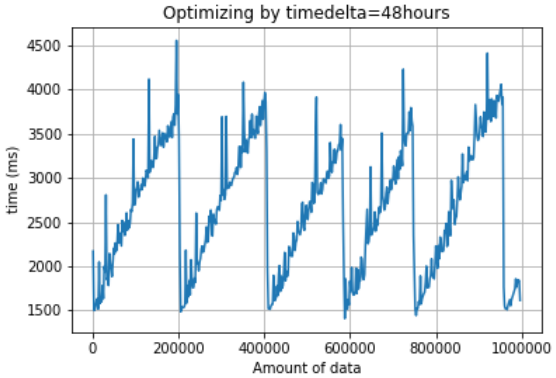
Joonistel 3 - 6 on Y-teljel välja toodud iga päringu kestvus millisekundites ning X-teljel andmete hulk. Graafikul tähistab üks andmepunkt ühte päritud lehekülge ehk siis 2000 andmerida. Näiteks: vaadates X-teljel numbrit 200 000, tähendab see, et tehti päring, kus leheküljestamise tõttu jäeti vahele 200 000 rida ning edastati kasutajale read 200 001 - 202 000. Suured hüpped päringute pikkuses tulenevad sellest, et mõnes ajavahemikus on vähem andmeid kui teistes. Näiteks öösel vahemikus 00:00 kuni 07:00 on loendurid salvestanud kümnetes kordades vähem andmeid kui päevaste tipptundide ajal.

Esmalt tehti päring teha ilma timedelta'ت kasutamata. Kuigi esimesed leheküljed võtsid alla kahe sekundi aega, on joonisel 3 näha, et nendele järgnevate lehekülgede pärimise aeg kasvab lineaarselt keskmiselt 1 sekundi võrra iga 100 000 kande kohta. Viimased leheküljed, alates 900 000 andmerea kättesaamisest, võtavad igaüks üle kümne sekundi aega.



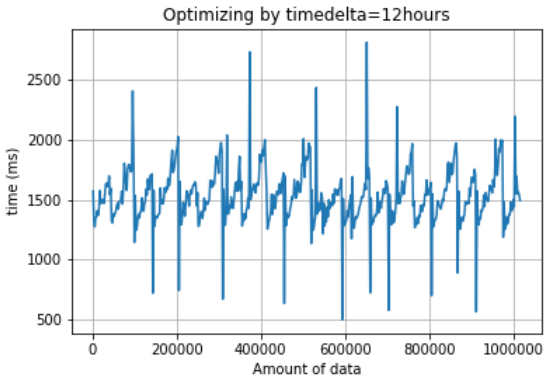
Joonis 3. Päringu tegemine ilma timedelta parameetrita

Teiseks tehti sama päring kasutades timedelta parameetriks 48 tundi ehk 2 päeva. Kogu 10 päeva ja 12 tunnine ajavahemik on eraldatud kuueks kahepäevase vahemikuga päringuks. Joonisel 4 on näha kuidas enne uut ajavahemikku tõusis lehekülje tagastamise aeg 4 sekundini.



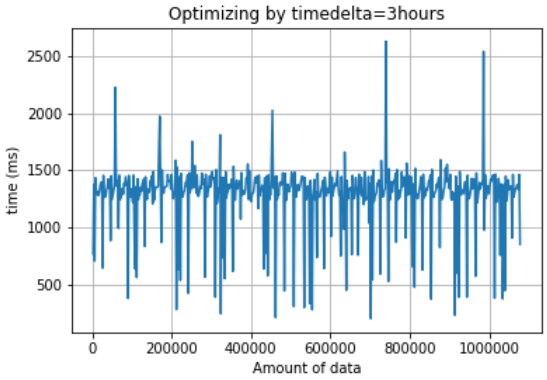
Joonis 4. Päringu tegemine kui timedelta on 48 tundi

Kolmandaks tehti sama päring kasutades timedelta väärtuseks 12 tundi. Joonis 5 illustreerib kuidas lehekülje tagastamise aeg langes mõne erandiga maksimaalselt 2 sekundini.



Joonis 5. Päringu tegemine kui timedelta on 12 tundi

Viimaseks tehti sama päring timedelta väärtusega 3 tundi. Nagu joonise 6 peal on näha, et päringu tegemise aeg on üpris konstantne olenemata andmemahust.

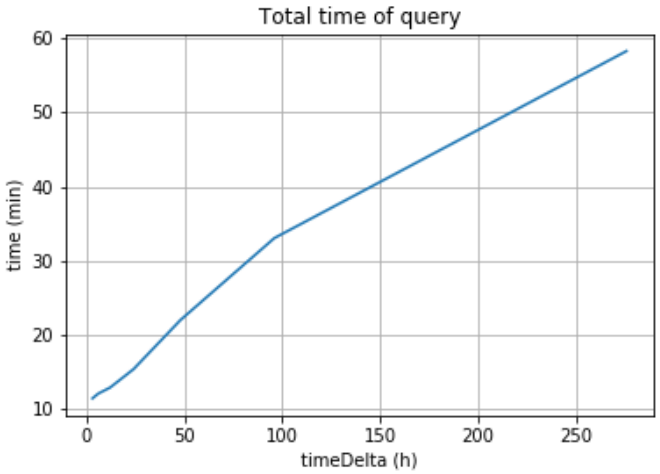


Joonis 6. Päringu tegemine kui timedelta on 3 tundi

Tabelis 1 on välja toodud eelnevate päringute timedelta väärtus ning igale väärtusele vastav kogu andmehulga pärimise aeg. Päringu ajavahemiku väiksemateks osadeks jagamise suurim eelis on see, et iga lehekülje pärimise aeg kasvab ainult mingi kindla ülemise piirini, enne kui alustatakse uuesti nullist. Kui parameetrit mitte kasutada, siis iga järgnev päring läheb aina aeglasemaks, kuni lõpuks võivad tekkida probleemid kliendiga ühenduse osas.

Tabel 1. Timedelta väärtused ja päringute kiirused

timedelta väärtus	päringu kiirus	keskmine lehekülje tagastamise aeg
Ilma parameetrita	58 minutit	7.1 sekundit
48h	22 minutit	2.7 sekundit
12h	13 minutit	1.6 sekundit
3h	11 minutit	1.4 sekundit



Joonis 7. Päringute aegade ja timedelta kasutuse võrdlus

Joonis 7 näitab kuidas timedelta parameetri väärtus ning kogu andmehulga pärimise kiirus on lineaarses korrelatsioonis. Mida väiksem on parameeter, seda kiirem on kogu andmehulga pärimine. Hetkest, mil iga ajavahemiku sisse jääb alla 10 000 andmerea, pole parameetri vähendamine enam kasulik, kuna iga päring tehakse konstantse ajaga. See tähendab, et kui ajavahemikus on rohkem kui 5 lehekülge (eeldusel et lehekülje suurus on 2000), saab seda kiirendada kasutades timedeltat.

4 Liiklusloendurite andmete analüüs

Selles peatükis kirjeldatakse Tartu linna liiklusloendurite ajalooliste andmete analüüsi. Tuuaks välja, mis eesmärgil analüüsi tehakse, mis kujul andmed on, kuidas nendega töötati ning mis tulemusteni jõuti. Keskendutakse Metallica kontserdile, mis toimus 18. juulil 2019.a.

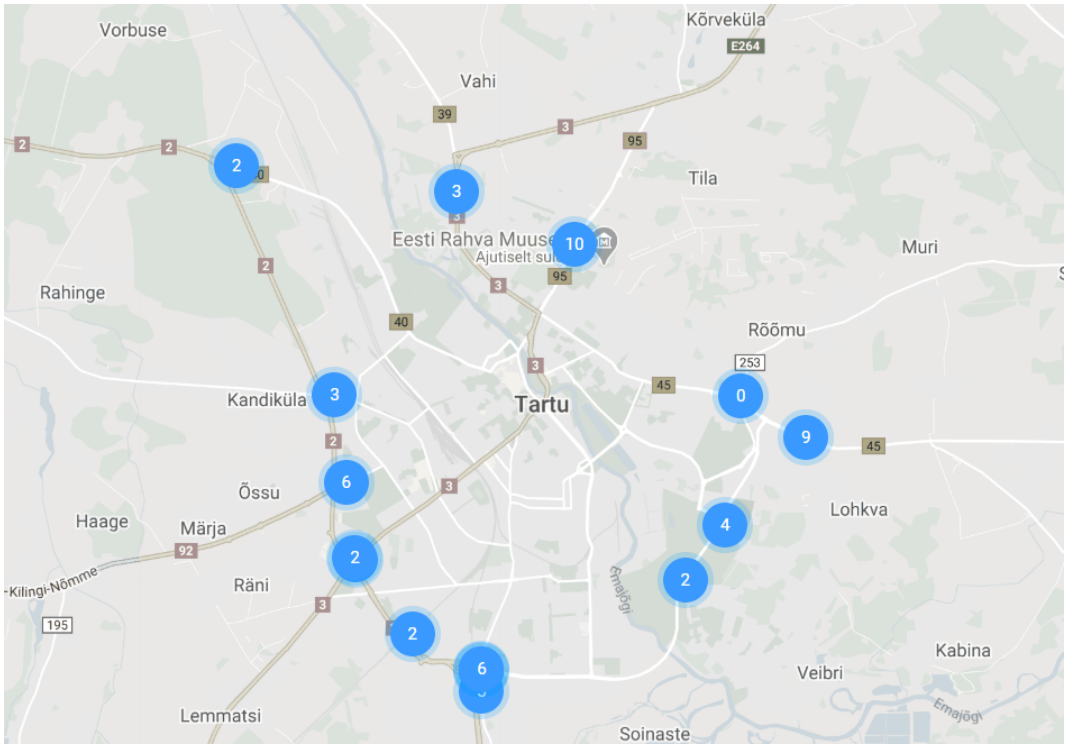
4.1 Eesmärgid

Oluliseks eesmärgiks on loodud teeki kasutada päriselulises andmeanalüüsi situatsioonis ning valideerida selle jõudlust ning kasutusmugavust. Lisaks on Tartu Linnavalitsus välja pakkunud uurimisküsimused, millele otsitakse andmeanalüüsi käigus vastuseid. Küsimused on järgnevad:

- Millist maanteed mööda sõitsid inimesed Tartusse enne Metallica kontserdit?
- Mis kellaaegadel hakati Metallica kontserdi jaoks kogunema Tartusse?
- Millist maanteed mööda sõitsid inimesed Tartust välja peale Metallica kontserdit?
- Mis kellaaegadel väljuti Tartust peale Metallica kontserdit välja?

4.2 Andmete ülevaade

Andmete allikaks on 16 liiklusloendurit, mis on paigutatud ümber Tartu linna sissesõidu maanteedele. Kõik loendurid registreerivad nii sisse kui ka väljasõite mõne erandiga. Eranditeks on Ringtee andurid, mida on kolm. Ringteel loeb esimene andur ainult sissesõite, teine loeb ainult väljasõite ning kolmas loeb mõlemat. Põhjuseks on erinevate teede eraldi jälgimine. Teiseks erandiks on Riia maantee loendurid, mida on kaks tükki. Üks loeb sissesõite ning teine väljasõite. Joonis 8 kirjeldab loendurite asukohta ning joonisel andurite peal olevad numbrid näitavad viimaste minutite liiklustihedust.



Joonis 8. Liiklusloendurite asukohad Telia keskkonna [20] visualisatsiooni andmetel

Loendurite poolt tagastatavad andmed on koodinäide 14 poolt kirjeldatud struktuuriga evendid. Koodinäide 14 pole täielik, kuna loetavuse jaoks on ’self’ URL-id lühendatud.

```
{
  'creationTime': '2019-07-01T16:44:33.057+03:00',
  'source': {
    'name': '14 AVC controller (Narva mnt)',
    'self': 'https://.../inventory/managedObjects/117925749',
    'id': '117925749'
  },
  'type': 'avc_VehicleEvent',
  'self': 'https://.../event/events/126962687',
  'time': '2019-06-29T21:00:01.748Z',
  'id': '126962687',
  'text': 'Vehicle detected',
  'avc_Vehicle': {
     'occupancy': 240,
     'vehicle_class': 1,
     'gap': 206200,
     'length': 3.8,
     'lane_id': 2,
```

```
        'vehicle_id': 6319529,
        'speed': 56
    }
}
```

Koodinäide 14. Liiklusloenduri event

Selle analüüsi kõige olulisemateks osadeks on 'avc_Vehicle' fragment ning 'time'. Tähtis on mainida, et 'time' on alati UTC ajatsoonis ning see märgib evendi saatmise kuupäeva ja kellaaega. Kogu info loendurist möödasõidu kohta on 'avc_VehicleEvent' fragmendis, mida tabel 2 kirjeldab.

Tabel 2. Loenduri poolt tagastatud evendi fragmendi kirjeldus

nimi	selgitus
occupancy	aeg millisekundites kui kaua sõiduk loenduri vaateväljas püsis
vehicle_class	sõidukile määratud klass.
gap	vahe eelnevalt möödunud sõidukiga
length	sõiduki pikkus
lane_id	rada millel sõiduk loendurist möödudes sõitis
vehicle_id	automaatselt genereeritud id igale sõidukile
speed	kiirus millega sõiduk loendurist möödus km/h

Tabelis 3 on välja toodud kuidas sõidukile määratakse klass vastavalt selle pikkusele.

Tabel 3. Sõiduki klassi määramine

klass	pikkus	sõiduki tüüp
1	lühem kui 5.5m	sõiduauto
2	5.5m - 16.4m	kaubik
3	pikem kui 16.4m	rekka või buss

Lisaks loendurite poolt tagastatavatele andmetele on oluline teada, millistele sõiduradadele vastavad sissesõidud ning millistele väljasõidud. Selleks on eraldi CSV fail, mis kirjeldab loendurite ja nende sõiduradade sisse- ning väljasõite. Lisade peatükis II on tabelina välja toodud selle CSV sisu, kus -1 tähistab linnast väljasõitu ning 1 tähistab linna sissesõitu.

4.3 Andmeanalüüsi töövoog

Andmeanalüüs viidi läbi kasutades Jupyter notebooki[8]. Esmalt päriti selles lõputöös loodud teeki kasutades kogu juunikuu andmed. Neid kasutatakse selleks, et leida tavapärase nädala keskmine liiklustihedus. Teiseks päriti kogu Metallica nädala andmed, et saada ülevaade kontserdile eelnevate ja järgnevate päevade liiklusest. Kontsert toimus 18. juulil, neljapäeval ning andmeid päriti 15.-21. juuli kohta. Suundade määramiseks oli tarvis importida eelnevalt mainitud CSV fail loendurite ja nende sõiduradade suundadega. Andmeanalüüsiks tehtud Jupyter Notebooki fail on kaasa lisatud koos selle lõputööga.

Eeltöötluse vältel kasutab töö autor Pandase sisseehitatud funktsioone. Lihtsamaks grupeerimiseks peab igale andmereale juurde lisama sõidusuuna vastvalt loendurile ja sõidurajale. Selleks ühendatakse päritud andmed ning sõidusuundade CSV faili read anduri nime ning lane_id veeru järgi. Kuna aeg on UTC ajatsoonis, peab lisama igale ajahetkele juurde 3 tundi selleks, et ajatsoonid ühilduksid. Nüüd kui igal evendil on suund määratud ning ajatsoon õige, saab neid Pandase group_by funktsiooniga lihtsasti grupeerida ja kokku loendada tunni aja kaupa. Metallica nädala jaoks on seda lihtsam teha, kuna kogu andmestik koosnebki vaid ühest nädalast. Juunikuu jaoks on tarvis leida iga nädalapäeva tunnile vastavad keskmised. Peab tegema kindlaks, et ka mõnede loendurite ühesuunalisest lugemisest või puuduvatest andmetest tulenevad 0 väärtused oleksid andmestikus olemas. Grupeerimise tulemus on näidatud *Joonis 9* peal.

[8] `https://jupyter.org/`

	hour	direction	weekday	radar	size
0	0	-1	Friday	01 AVC controller (Ilmatsalu)	251
1	0	-1	Friday	02 AVC controller (Viljandi mnt)	316
2	0	-1	Friday	03 AVC controller (Riia mnt LS)	0
3	0	-1	Friday	04 AVC controller (Riia mnt LV)	780
4	0	-1	Friday	05 AVC controller (Roopa)	50
5	0	-1	Friday	06 AVC controller (Võru)	198
6	0	-1	Friday	07 AVC controller (Ringtee 1)	82
7	0	-1	Friday	08 AVC controller (Ringtee 2)	368
8	0	-1	Friday	09 AVC controller (Ringtee 3)	0
9	0	-1	Friday	10 AVC controller (Ihaste)	71
10	0	-1	Friday	11 AVC controller (Lammi)	34
11	0	-1	Friday	12 AVC controller (Räpina mnt)	322
12	0	-1	Friday	13 AVC controller (Rõõmu)	53
13	0	-1	Friday	14 AVC controller (Narva mnt)	279
14	0	-1	Friday	15 AVC controller (Aruküla)	406
15	0	-1	Friday	16 AVC controller (Tiksoja)	324
16	0	-1	Monday	01 AVC controller (Ilmatsalu)	18
17	0	-1	Monday	02 AVC controller (Viljandi mnt)	16
18	0	-1	Monday	03 AVC controller (Riia mnt LS)	0
19	0	-1	Monday	04 AVC controller (Riia mnt LV)	54

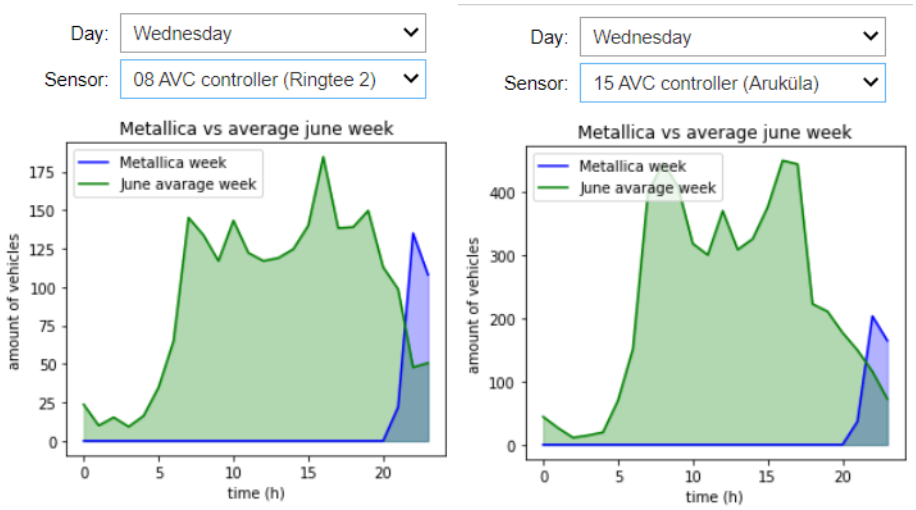
Joonis 9. Grupeeritud liiklusloendurite andmed

Jupyter notebookis kasutatakse interaktiivsete graafikute loomiseks widgeteid. Need on erinevad komponendid alates rippmenüüdest kuni liuguriteni. Widgeti väärtuse muutumisel käivitatakse talle määratud funktsioon uue muutunud väärtusega. Selles töös on kasutatud radar graafikuid ning nende interaktiivseks päeva valimiseks rippmenüüd ning tunni valimiseks liugurit.

Kuna radargraafik keskendub kõikide loendurite võrdlusele ühe tunni lõikes, ei anna see head ülevaadet konkreetse loenduri tervest päevast. Parema ülevaate saamiseks kasutatakse lisaks joongraafikut koos kahe interaktiivse rippmenüüga, kust saab valida loendurit ning päeva.

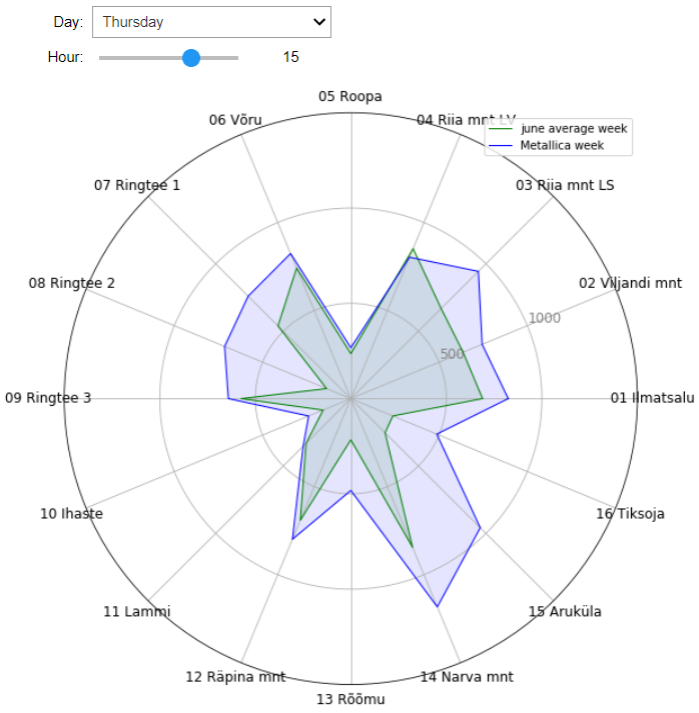
4.4 Tulemused

Metallica kontsert algas ametlikult kell 21:00 ning lõppes 23:30. Analüüsi jaoks vaadati läbi terve nädal ning selles alampeatükis tuuakse välja suurimad leitud erinevused võrreldes keskmise juuni nädalaga.



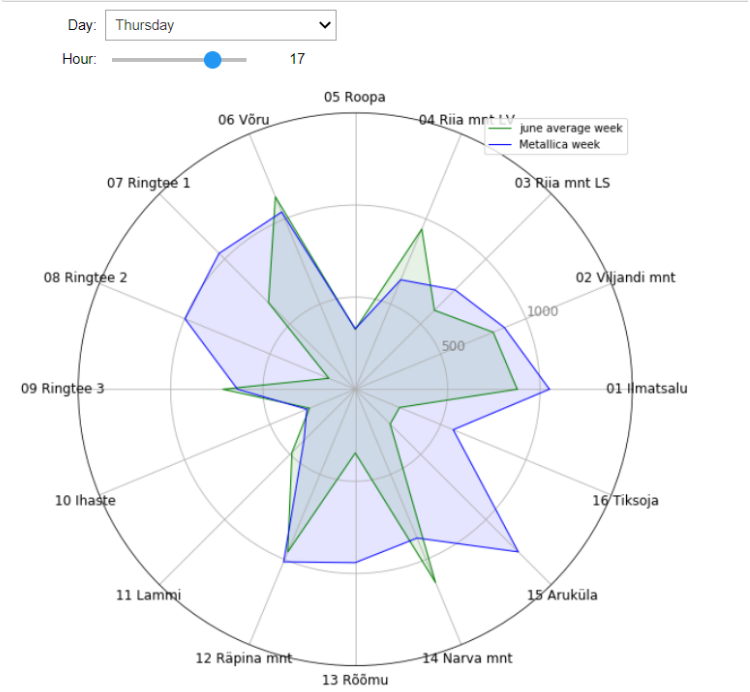
Joonis 10. Ringtee 2 ja Aruküla loendurid

Andmete kontrollimisel selgus, et 15. juuli päevast kuni 17. juuli õhtuni puuduvad andmeid Aruküla ja Ringtee 2 loenduritelt. 17. juulil, peale kella 22:00, kui andmed hakkasid saabuma, on neid palju rohkem kui tavanädalal. Seda illustreerib ka joonis 10. Sellest võib järeldada, et nende loenduritega võis sel nädalal olla mingisugune probleem ning need kas ei töötanud või ei saatnud oma andmeid õige kuupäevaga Cumulocity'sse. Hiljem kinnitas ka Tartu Linnavalitsus loendurite rikke sel perioodil. Neid loendurite andmeid analüüsis ei arvestata.



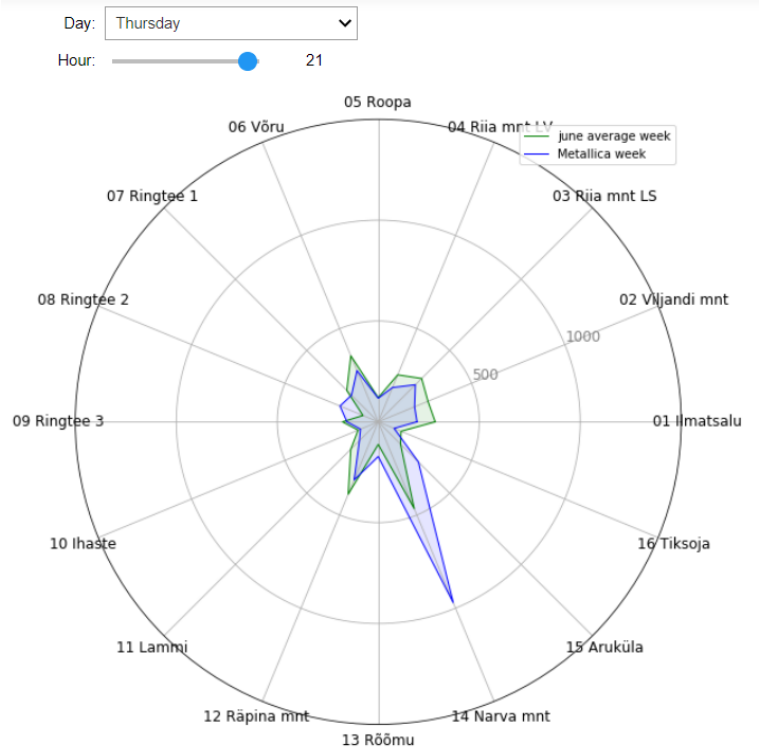
Joonis 11. Metallica nädal neljapäeval kell 15.00-15.59

Esimesed suuremad erinevused liiklustiheduses kerkisid esile Metallica kontserdipäeval ehk neljapäeval, 18. juulil ajavahemikus 14:00 - 18:00. *Joonis 11* järgi on kella 15:00 ajal liiklus kõigil loenduritel peale Roopa, Ihaste, Lammi ja Ringtee 3 tavapärasega võrreldes kasvanud. Suurim liiklustihedus jagunes Narva mnt, Tiksoja, Riia mnt LS(Linna Sissesõit), Võru ja Ringtee 1 vahel.



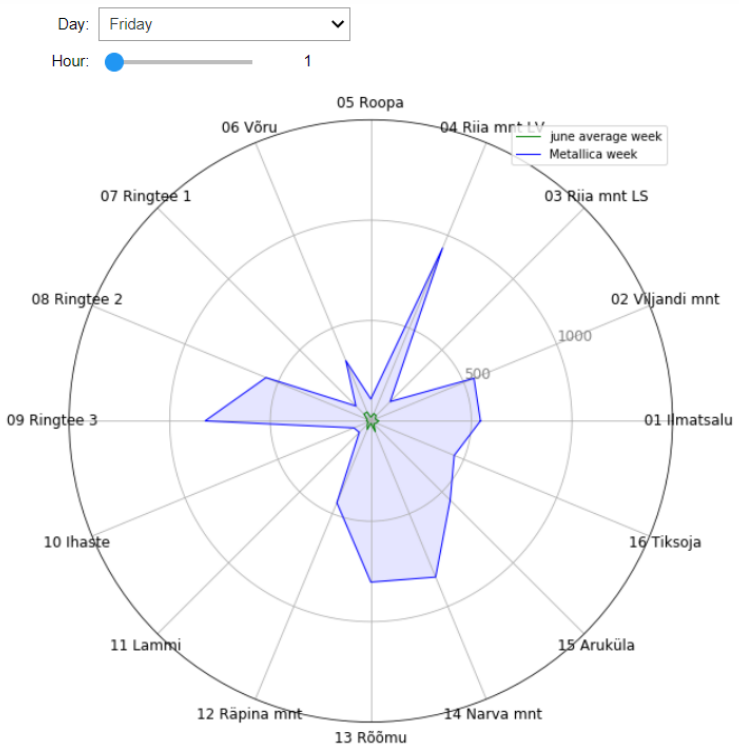
Joonis 12. Metallica nädal neljapäeval kell 17.00-17.59

Joonis 12 pealt on näha, et kella 17ks oli suurem osa liikluskoormusest Rõõmu, Tiksoja ja Ringtee 1 peal. Riia mnt LV (Linnast Väljasõit) ja Narva mnt koormus olid selleks hetkeks vähenenud alla tavapärase.



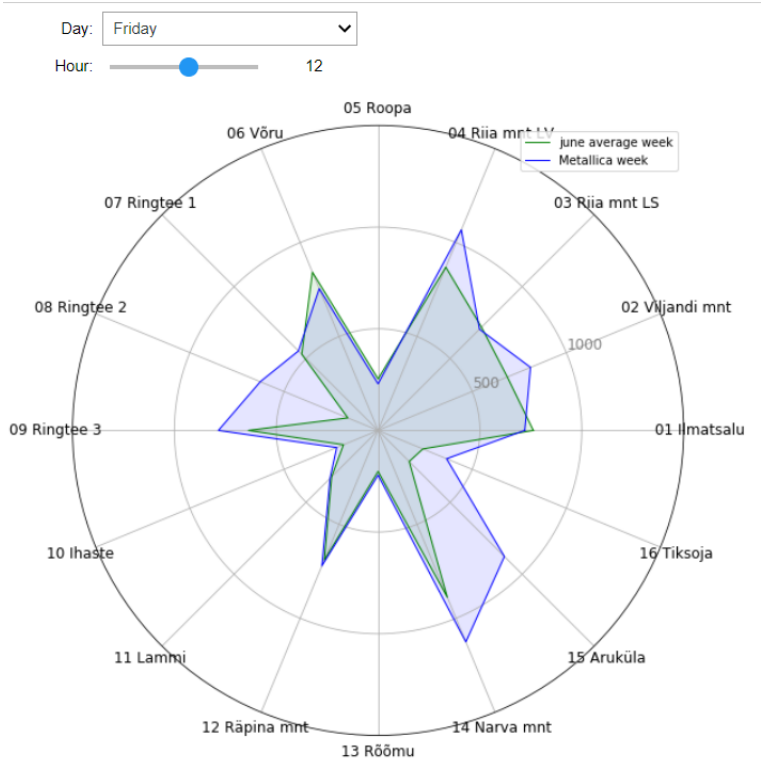
Joonis 13. Metallica nädal neljapäeval kell 21.00-21.59

Kella 21ks, kui Metallica kontsert ametlikult algas, oli *Joonis 13* järgi liiklus üpris sarnane tavapärasele, ühe erandiga. Nimelt olid Narva mnt loenduri näidud keskmisest kaks korda kõrgemad. Tartu Linnavalitus kinnitas hiljem, et olid suunanud liikluse selleks hetkeks Narva mnt peale.



Joonis 14. Metallica nädal reedel kell 01.00-01.59

Jooniselt 14 on näha, et suurem osa kontsertikülastajatest alustas Tartu linnast väljasõitu reedel, 19.juulil südaöö paiku. Lahkujad kasutasid suuremas osas Ringtee 3, Riia mnt, Narva mnt ja Rõõmu sõiduteid.



Joonis 15. Metallica nädal reedel kell 12.00-12.59

Silmapaistev oli ka kontserdijärgse päeva keskpäev, kus Ringtee 3, Narva mnt jt. anduritel oli tavapärasest märgatavalt suurem koormus. Erinevus on nähtav *Joonis 15* peal. Arvatavasti alustasid ööseks Tartusse jäänud kontserdikülastajad sel ajal oma sõitu linnast välja .

Kõige vähem olid Metallica kontserdist mõjutatud Ihaste, Lammi ja Roopa sissesõidud. Liikluse tihenemist oli esmalt märgata kontserdipäeval kell 15 kuid see normaliseerus kontserdi alguseks. Enamus Metallica kontserdi külastajatest lahkus Tartust kohe peale kontserti reedel, 19. juulil ajavahemikus 00:00-02:59. Peale kontserti Tartu linnas ööbinud külastajad lahkusid reedel, 19. juulil umbes keskpäeval.

5 Diskussioon

Andmete analüüsimise käigus tehti teegile mitu juurdearendust.

Esimeseks oli kasutajale progressi info näitamine. Varasemalt ei olnud aru saada kas andmeid päritakse või kood on kokku jooksnud. See andis hea ülevaate kui kiiresti pärimine käib ning kui suure andmekogusega on tegemist.

Peale progressi näitamise lisamist tuli esile, et iga lehekülje pärimine muutub aina aeglasemaks. Tekkis vajadus tegeleda optimeerimisega. Lisati juurde timedelta parameeter ning testiti milline väärtus on kõige optimaalsem.

Kolmandaks tekkis vajadus vigade haldusega tegelda. Vigase parameetri kasutamisel ei tagastatud esialgu kasutajale midagi peale errori sisu, mis oli tihti üpris üldine. Paremaks tagasisideks prinditakse nüüd errori tekkimisel kasutajale Cumulocity tagastatud HTTP staatuse kood, päringu URL ja errori sisu.

Andmeanalüütikule, kes soovib teha Cumulocity IoT platvormil asuvate andmetega analüüsi, pakub teek mitmeid kasulikke funktsionaalsusi. Mõned neist tulevad esile iga päringuga ning teised vaid suurte andmekogustega.

Kasutaja peab vaid täpsustama ajavahemiku ning mõne filtreerimise parameetri nagu tüüp või seadme id. Selle peale kodeeritakse päised, pannakse kokku päringu jaoks URL, tehakse päring, tasandatakse tagastatud andmete struktuur ühedimensiooniliseks ning teisendatakse andmed DataFrame kujule. Eraldi nende sammude tegemine ilma teeki kasutamata võib kujuneda ajakulukaks protsessiks olenevalt andmete struktuurist. Näiteks tasandamine nõuab juba rekursiivse funktsiooni loomist.

Suurte andmehulkade korral tehakse kasutaja eest ära kogu lehekülgede pärimine ning delta parameetri olemasolul ka kiiruse optimiseerimine ehk päringu mitmeks väiksemaks ajavahemikuks jagamine. Lisaks agregeeritakse kõikide ajavahemike tulemused kokku üheks DataFrameks.

6 Kokkuvõte

Töös kirjeldati teegi loomist Pythonis, mis keskendub Cumulocity REST API kaudu ajalooliste andmete pärimisele ning päritud andmete Pandase DataFrame kujule viimisele. Teek on kättesaadavaks tehtud ka Pythoni package manageris.

Nõuded teegile tulid osaliselt Tartu Linnavalitsuselt ning osaliselt andmeanalüüsi protsessi käigus esilekerkinud vajadustest. Toodi välja viis kuidas Cumulocity REST API kasutades saab suure andmekoguse pärimise teha kiiremaks, kui päring teha mitme erineva osana.

Lisaks kirjeldati andmeanalüüsi töövoogu ning 2019. aasta 18. juulil toimunud Metallica kontserti liiklusloendurite andmeanalüüsi tulemusi. Töö tulemustes selgus, et suurem osa kontserdi külastajatest sisenes Tartu linna kontserdi päeval kella 15 paiku ning enamus külastajatest lahkus kohe peale kontsertit, südaööl.

6.1 Tuleviku arendused

Teek toetab hetkel ajalooliste *measurements*'ide, *events*'ide ja seadmete pärimist. Teeki saab edasi arendada lisades sinna juurde toe pärida ka audit logisid ja alarme. Lisaks toetab Cumulocity API reaalajas andmete pärimist MQTT kaudu, mis oleks kasutajale kasulik.

Hetkel on teek fokuseeritud andmete pärimisele ning sellega ei saa Cumulocity'sse lisada andmeid, konfigureerida seadmeid ega teha muid muudatusi. API tugi eelnevalt kirjeldatud tegevustele on Cumulocity's olemas ning nende implementeermisel teeki annaks see kasutajale täieliku kontrolli oma kliendi üle.

Selles töös tehti kogu testimine käsitsi. Tulevikus on võimalik tekitada teegi edasiseks arenduseks automaattestid, mis jooksutatakse läbi enne iga uuenduse repositooriumisse lisamist. Automaattestid suurendavad kasutaja usaldust teegi vastu.

Viidatud kirjandus

[1] E. Theodoridis, G. Mylonas, and I. Chatzigiannakis, "Developing an iot smart city framework," *IISA 2013*, pp. 1–6, 2013. https://ieeexplore.ieee.org/abstract/document/6623710.

[2] "Smartencity about." https://smartencity.eu/about/. Vaadatud kuupäev: 2019-12-04.

[3] "Retrofitting soviet-era apartment buildings with 'smart city' features: The h2020 smartencity project in tartu, estonia." https://link.springer.com/chapter/10.1007/978-3-030-23392-1_17. Vaadatud kuupäev: 2020-04-01.

[4] "Tark tartu kavandatud tegevused." http://tarktartu.ee/avaleht/kavandatud-tegevused/. Vaadatud kuupäev: 2019-12-04.

[5] "Tartu – from hrustsovkas to smartkovskas." https://smartencity.eu/about/lighthouse-cities/tartu-estonia/. Vaadatud kuupäev: 2019-12-04.

[6] "Cumulocity iot." https://www.softwareag.cloud/site/product/cumulocity-iot.html#/. Vaadatud kuupäev: 2019-12-04.

[7] "Exporting data with reports from cumulocity." https://cumulocity.com/guides/users-guide/cockpit/#reports. Vaadatud kuupäev: 2020-03-27.

[8] "Softwareag - cumulocityr." https://github.com/SoftwareAG/cumulocityr. Vaadatud kuupäev: 2020-03-27.

[9] "pandas: a foundational python library for data analysis and statistics." https://www.dlr.de/sc/portaldata/15/resources/dokumente/pyhpc2011/submissions/pyhpc2011_submission_9.pdf. Vaadatud kuupäev: 2020-04-01.

[10] M. Gallagher and R. Trendafilov, "R vs. python: Ease of use and numerical accuracy," *Journal of Business and Accounting*, vol. 11, 2018.

[11] J. VanderPlas, *Python Data Science Handbook: Essential Tools for Working with Data*. 1st Edition, O'Reilly', 2016.

[12] H. Hejazi, H. Rajab, T. Cinkler, and L. Lengyel, "Survey of platforms for massive iot," *2018 IEEE International Conference on Future IoT Technologies (Future IoT)*, 1 2018. https://ieeexplore.ieee.org/abstract/document/8325598.

[13] P. Ray, P., "A survey of iot cloud platforms," *Future Computing and Informatics Journal*, pp. 35–46, 12 2016. https://www.sciencedirect.com/science/article/pii/S2314728816300149.

[14] "Cumulocity domain model." https://cumulocity.com/guides/concepts/domain-model/. Vaadatud kuupäev: 2020-03-27.

[15] K. Urrutia-Azcona, S.Štendorf-Sørensen, P. Molina-Costa, and I. Flores-Abascal, "Smart zero carbon city:key factors towards smart urban decarbonisation," *DYNA volume 94*, pp. 676–683, 11 2019. https://www.revistadyna.com/search/smart-zero-carbon-city-key-factors-towards-smart-urban-decarbonization.

[16] "Data flow." http://www.coordinationtoolkit.org/wp-content/uploads/130907-Data-flow.pdf. Vaadatud kuupäev: 2020-03-29.

[17] "Cumulocity rest implementation." https://cumulocity.com/guides/reference/rest-implementation/#http-usage. Vaadatud kuupäev: 2020-04-06.

[18] "Using cumulocity rest interface." https://cumulocity.com/guides/microservice-sdk/rest/. Vaadatud kuupäev: 2020-04-07.

[19] "Cumulocity inventory reference guide." https://cumulocity.com/guides/reference/inventory/#query-language. Vaadatud kuupäev: 2020-05-05.

[20] "Telia live platform." https://tarktartu.telia.ee. Vaadatud kuupäev: 2020-05-05.

Lisad

I. Koodirepositoorium

Töös valminud teek ja selle dokumentatsioon on kättesaadav lingilt `https://github.com/SilverLaius/cumulocitypython`. Kuna teegi kasutamine eeldab andmete olemasolu Cumulocity IoT platvormil, siis testimise vajaduse tekkimisel peaks ühendust võtma autoriga e-maili teel silver.laius@gmail.com.

II. Loendureid ja nende sõidusuunasid kirjeldav tabel

Tabel 4. Loendurid ja nende sõidusuunad

name	lane_1	lane_2	lane_3	lane_4
01 AVC controller (Ilmatsalu)	-1	1	0	0
02 AVC controller (Viljandi mnt)	-1	1	0	0
03 AVC controller (Riia mnt LS)	1	1	0	0
04 AVC controller (Riia mnt LV)	-1	-1	0	0
05 AVC controller (Roopa)	1	-1	0	0
06 AVC controller (Võru)	-1	1	0	0
07 AVC controller (Ringtee 1)	-1	-1	0	0
08 AVC controller (Ringtee 2)	-1	1	0	0
09 AVC controller (Ringtee 3)	1	1	0	0
10 AVC controller (Ihaste)	1	-1	0	0
11 AVC controller (Lammi)	-1	1	0	0
12 AVC controller (Räpina mnt)	1	1	-1	-1
13 AVC controller (Rõõmu)	-1	1	0	0
14 AVC controller (Narva mnt)	-1	1	0	0
15 AVC controller (Aruküla)	1	-1	0	0
16 AVC controller (Tiksoja)	-1	1	0	0

III. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, **Silver Laius**,

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

   Integratsiooniteek Cumulocity IoT platvormi ajalooliste andmete analüüsiks Pythonis,

   mille juhendaja on Jakob Mass,

   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.

Silver Laius
07.05.2020