TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Mikk Sillamaa
164488IAPB

TUDENGISATELLIIDI PROJEKTI PILTIDE ALAMSÜSTEEM

bakalaureusetöö

Juhendaja: Evelin Halling
PhD

Tallinn 2019

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Mikk Sillamaa

19.05.2019

Annotatsioon

Töö eesmärgiks on implementeerida TTÜ tudengisatelliidi missioonijuhtimistarkvarale lisafunktsionaalsus. Selle abil saab kindlaks määrata piltide tegemise parameetrid ja nende põhjal soovitud satelliidi ülelennud saada.

Käesoleva töö tulemusena valmib eelpool nimetatud lisafunktsionaalsus ning analüüs selle kohta.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 28 leheküljel, 7 peatükki, 18 joonist, 0 tabelit.

Abstract

Student satellite project's pictures subsystem

The aim of this thesis is to implement an additional functionality for TTU100 student satellite’s mission control system. The functionality will fulfill the needs of different parties who will use this.

This thesis consists of two parts. The first part has two subparts. The first subpart is a HTML form, which lets users insert parameters for making pictures with a satellite and shows maps with received fly overs information. The second subpart is also a HTML form, which shows information of picture parameters that are ready for sending to the satellite. The second part is a REST-based service that will get fly overs for specified satellite according to specified parameters.

The thesis gives a brief overview of the current mission control system. Additionally, it introduces requirements for the additional functionality and problems occured during implementing it. The last chapter is about testing the software.

The thesis is in Estonian and contains 28 pages of text, 7 chapters, 18 figures, 0 tables.

Lühendite ja mõistete sõnastik

1U	*One unit,* kuupsatelliitide standardsuurus
API	*Application programming interface*, rakendusliides
CSS	*Cascading Style Sheets*, keel veebilehe kujundamiseks
DPI	*Dots per inch*, punkti tolli kohta
ERD	*Entity Relationship Diagram,* olemisuhte diagramm
HTML	*Hypertext marking language*, hüperteksti markeerimiskeel
JSON	*Javascript object notation*, andmevahetusvorming
Kallutusnurk	Nurk pildistamise suuna ja satelliidi nadiiri vahel. *Pitch angle*
LEO	*Low Earth Orbit,* Maa-lähedane orbiit
MCS	*Mission Control System,* missioonijuhtimistarkvara
MVC	Model-View-Controller, tarkvara arhitektuurimuster
NIR	*Near infrared*, lähi-infrapuna spekter
REST	*Representational state transfer,* stiil API liideste loomiseks
RGB	*Red-Green-Blue,* värvimudel
SQL	*Structured query language*, struktureeritud päringkeel

Sisukord

Jooniste loetelu

1 Sissejuhatus

TTÜ100 tudengisatelliit on 1U (*one unit*) satelliit, mille mõõtudeks on 1x1x1 dm. See on minitehiskaaslane, mis hakkab lendama LEO (*low Earth Orbit*) orbiidil ehk Maa-lähedasel orbiidil. Selle satelliidi põhiline eesmärk on tegeleda Maa seirega. Selleks kasutatakse RGB (*Red-Green-Blue*) ning NIR (*Near infrared*) kaameraid, mis tegelevad vastavalt inimsilmale nähtava valguse piltide ning lähi-infrapuna spektris piltide tegemisega. Lisaks piltide tegemisele tegeleb satelliit ka pilditöötlusega. Side maajaamaga toimub kahel sagedusel. Andmeside piiratuse tõttu peab pilditöötluse süsteem tegelema piltide prioriseerimisega. Maa seire missiooni kõrval tegeleb satelliit ka arvutustehnika tõrkekindluse ning optilise side katsetega. [1]

Lõputöö eesmärgiks on luua eraldi Spring Booti[1] rakendus, mis on REST (*Representational state transfer*) teenus. See teenus võimaldab määrata parameetrid satelliidi piltide tegemiseks. Vastavalt etteantud parameetritele leitakse ajad, millal satelliit määratud asukohast üle lendab ja mis vastavad määratud kriteeriumitele. Lisaks on kavas tulevikus luua MCS-i (*mission control system*) backoffice-i moodulisse pilditöötluse klient, mille abil saaks pildi parameetreid kindlaks määrata ning neid sõnumitöötluse põhivoo algusesse saata.

Peatükis 2 esitatakse nõuded loodavale süsteemile.

Peatükis 3 kirjeldatakse satelliidi projektis olevat MCS-i ja tarkvaraarenduse metoodikat. Samuti tuuakse välja kasutatavad programmid, teegid ja raamistikud.

Peatükis 4 kirjeldatakse loodava süsteemi arhitektuuri ning erinevaid osasid, nende erinevused tuuakse välja tegevusdiagrammil.

Peatükis 5 tuuakse välja loodud süsteemi struktuur.

1 https://spring.io/projects/spring-boot

Peatükis 6 kirjeldatakse loodava teenuse testimist ning saadavate andmete võrdlemist teiste sarnaste programmidega.

2 Nõuded loodavale süsteemile

Loodava süsteemi nõuded on välja mõeldud selleks, et tagada lõppkasutaja soovide täitmine. Seega on need kooskõlastatud nii tiimi kui ka praeguste lõppkasutajatega. Järgnevalt on välja toodud loodava teenuse nõuded alamteenuste kaupa ja seda nii sisend- kui ka väljundandmete osas.

2.1 Nõuded loodavale teenusele

Järgnevalt on kirjeldatud nõuded loodavale REST teenusele.

2.1.1 Nõuded tagastatavale informatsioonile

Teenus peab tagastama informatsiooni ülelendude kohta.

Tagastatav informatsioon peab olema nimekiri järgnevast:

- Ülelennu algusaeg.
- Ülelennu algusaja asimuut kraadides ujukomaarvuna.
- Kuupäev ja kellaaeg, millal satelliit on saavutanud maksimaalse kõrguse, mõõdetuna merepinnalt.
- Maksimaalse kõrguse nurk kraadides ujukomaarvuna.
- Ülelennu lõpu kuupäev ja kellaaeg.
- Ülelennu lõpu asimuut kraadides ujukomaarvuna.

- Nimekiri ülelennu jooksul 30-sekundilise intervalli tagant tehtud andmete pärimistest. Üks andmete pärimine peab sisaldama endas:
    - Satellidi praeguse hetke asukohta maapinnal.
    - Satelliidi praeguse hetke kõrgust mõõdetuna merepinnast meetrites.
    - Satelliidi praeguse hetke kõrguse nurka kraadides ujukomaarvuna.
    - Satelliidi praeguse hetke asimuudi nurka kraadides ujukomaarvuna.
    - Satelliidi praeguse hetke kuupäeva ja aega.
    - Kallutusnurka (nurk pildistamise suuna ja satelliidi nadiiri vahel) kraadides kujul kraad:minut:sekund.

Teenuse poolt tagastatava informatsiooni koodinäide on lisas 1.

2.1.2 Nõuded koolidele mõeldud alamteenusele

Loodav teenus peab koolidelt aktsepteerima päringuid, mis sisaldavad järgmisi väärtusi:

- Kooli pikkus- ja laiuskraadid.
- Kas soovitakse saada ainult päevaste, öiste või mõlema aja ülelendude informatsiooni.
    - Ainult päevaste ülelendude saamiseks tuleb selle parameetri väärtuseks panna 1.
    - Ainult öiste ülelendude saamiseks tuleb selle parameetri väärtuseks panna 2.
    - Mõlema aja ülelendude saamiseks tuleb selle parameetri väärtuseks panna 3.
- Kooli parameeter tuleb väärtustada tõeseks.

  Koolide poolt saadetava päringu koodinäide on lisas 2.

Koolid saavad vastuseks alampeatükis 2.1.1 välja toodud informatsiooni ja seda päringu tegemise hetkest järgmise 14 ööpäeva kohta.

2.1.3 Nõuded android rakendustele mõeldud alamteenusele

Loodav süsteem peab android rakendustelt vastu võtma järgneva sisuga päringuid:

- Satelliidi nimi.
- Oma pikkus- ja laiuskraadid.
- Mobiili parameeter tuleb väärtustada tõeseks.

Android seadmed saavad vastuseks ainult järgmise ülelennu informatsiooni.

Android seadmete poolt saadetava päringu koodinäide on lisas 3.

2.1.4 Nõuded pilditöötluskliendile mõeldud alamteenusele

Loodav teenus peab pilditöötluskliendilt vastu võtma järgmised parameetrid:

- Satelliidi nimi.
- Oma pikkus- ja laiuskraadid kraadides.
- *Minimum slay angle* kraadides ujukomaarvuna.
    - Väärtus peab jääma vahemikku 0 kuni 90.
- Kas tahetakse teha ainult üks pilt või mitu pilti mingi intervalliga.
    - Kui tahetakse teha ainult üks pilt, siis tuleb veel kindlaks määrata:
        - Alguskuupäev.
    - Kui tahetakse teha mitu pilti, siis tuleb veel kindlaks määrata:
        - Periood, mille tagant tahetakse ülelende saada.
            - Periood võib olla tund, päev, nädal, kuu või aasta.
            - Ühe pildi tegemise puhul on perioodiks üks tund.
        - Alguskuupäev.
        - Lõpukuupäev.

- Mitu pilti tahetakse määratud kohast ühe ülelennu ajal teha.
    - Kui see parameeter on suurem nullist, tuleb veel kindlaks määrata:
        - *Sequence image target motion* kraadides ujukomaarvuna
        - *Off earth target image vector* kraadides ujukomaarvuna
- Pildi prioriteetsus täisarvuna vahemikus 0 kuni 10.
- Millise kaameraga soovitakse pilti teha.
    - Ainult RGB kaameraga piltide tegemiseks tuleb selle parameetri väärtuseks panna 1.
    - Ainult NIR kaameraga piltide tegemiseks tuleb selle parameetri väärtuseks panna 2.
    - Mõlema kaameraga pildistamiseks tuleb selle parameetri väärtuseks panna 3.
- Kas soovitakse saada ainult päevaste või öiste või mõlema aja ülelendude informatsiooni.
    - Ainult päevaste ülelendude saamiseks tuleb selle parameetri väärtuseks panna 1.
    - Ainult öiste ülelendude saamiseks tuleb selle parameetri väärtuseks panna 2.
    - Mõlema aja ülelendude saamiseks tuleb selle parameetri väärtuseks panna 3.
- Maksimaalne pilvisus ujukomaarvuna vahemikus 0 kuni 100.
- Kooli parameeter tuleb väärtustada vääraks.

Pilditöötluse klient saab päringule vastuseks informatsiooni ja seda määratud alguskuupäevast järgmise ööpäeva kohta või siis kõik ülelennud, mis jäävad algus- ning lõpukuupäeva vahele.

Pilditöötluse kliendi poolt saadetava päringu koodinäide on lisas 4.

2.2 Nõuded loodavale pilditöötluskliendile

Pilditöötlusklient koosneb kahest osast.

Esimene peab aitama kindlaks määrata pildi tegemise parameetrid.

Teine osa peab kuvama kõik saatmiseks valmis olevad piltide parameetrid.

2.2.1 Pilditöötluskliendi esimese osa nõuded

Pilditöötluskliendi esimene osa peab laskma kasutajal HTML (*hypertext marking language*) vormis sisestatada peatükis 2.1.4 välja tootud parameetrid.

Lisaks peab klient kuvama loodava REST teenuselt saadud ülelennud tabelina.

Iga ülelennu kohta peab tabelis olema:

- Ülelennu algusaeg.
- Ülelennu algusaja asimuut kraadides.
- Aeg, millal satelliit on saavutanud maksimaalse kõrguse.
- Maksimaalse kõrguse nurk kraadides.
- Ülelennu lõppaeg.
- Ülelennu lõpuaja asimuut kraadides.
- Nupp, millele klikkimisel peab vastava rea all avanema kaart, kust on näha vastava ülelennu informatsioon 30 sekundi tagant tehtud andmete pärimistest.
    - Kaardil olevad markerid peavad olema asetatud vastavatele satelliidi Maa koordinaatidele.
    - Markerile klikkimisel peab info aknas näha olema:
        - Satelliidi kõrgus merepinnast kilomeetrites.
        - Kallutusnurk.
    - Info aknas peab olema võimalik hetkel lahti oleva infopäringu kallutusnurka valida.

- Pärast kallutusnurga valimist tuleb kuvada tekst, et vastav kallutusnurk on valitud ning peab olema võimalik seda pildi parameetreid hoidvasse tabelisse loodava teenuse abil salvestada.

2.2.2 Pilditöötluskliendi teise osa nõuded

Pilditöötluskliendi teine osa peab kuvama kõik ülessaatmiseks valmisolevad piltide parameetrid tabelis.

Iga pildi parameetrite objekti kohta peab tabelis olema kuvatud peatükis 2.1.4 kirjeldatud parameetrid ja lisaks veel salvestatud kallutusnurk ning kas päring on tulnud android seadmelt.

Iga tabeli rea viimases veerus peab olema nupp.

Selle nupu vajutamisel tuleb tulevikus loodava teenuse abil genereerida vastava pildi parameetrite objektist kaheksandkood ning viimane MCS tarkvara sõnumitöötluse põhivoo algusesse saata.

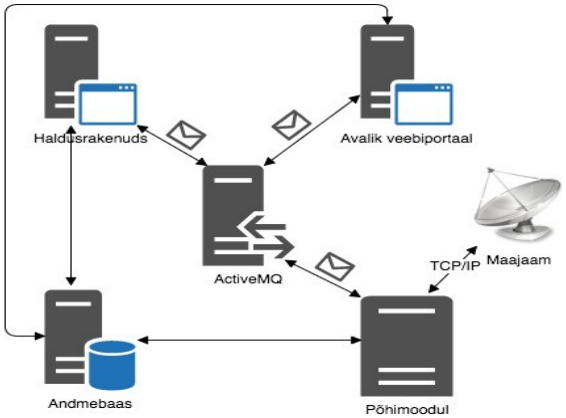
3 Missioonijuhtimistarkvara

Selles peatükis kirjeldatakse satelliidi projektis olevat MCS-i ja tarkvaraarenduse metoodikat. Samuti tuuakse välja kasutatavad programmid, teegid ja raamistikud.

3.1 Ülevaade missioonijuhtimistarkvarast

TTÜ satelliidiprojekti missioonijuhtimistarkvara on tudengite poolt arendatud tarkvara, mille arendamiseks andis tõuke asjaolu, et olemasolevad lahendused on kesised või ei rahulda kõiki vajadusi. MCS tarkvara on mõeldud selleks, et TTÜ100 tudengisatelliidile saaks mugavalt käske genereerida ning satelliidilt tulnud informatsiooni lõppkasutaja jaoks kuvada. Informatsioon liigub missioonijuhtimistarkvarasse läbi maapealse tugijaama või kolmanda osapoolte raadiojaamade. Kolmandateks osapoolteks on koolid. Tarkvara versioonihalduseks on kasutusel gitlab[1] ning keskkonna ülesseadmiseks kasutatakse dockerit[2]. [2]

Satelliidi missioonijuhtimistarkvara koosneb 5 komponendist: avalikust veebiportaalist, haldusrakendusest, põhimoodulist, sõnumivahetuse komponendist ning andmebaasist (Joonis 1). Sõnumivahetuseks kasutatakse ActiveMQ[3] programmi. [2]



Joonis 1. MCS-i arhitektuur [2]

1 https://about.gitlab.com/
2 https://www.docker.com/
3 https://activemq.apache.org/

Järgnevalt toob töö autor välja kõigi missioonijuhtimistarkvara komponentide lühikirjeldused.

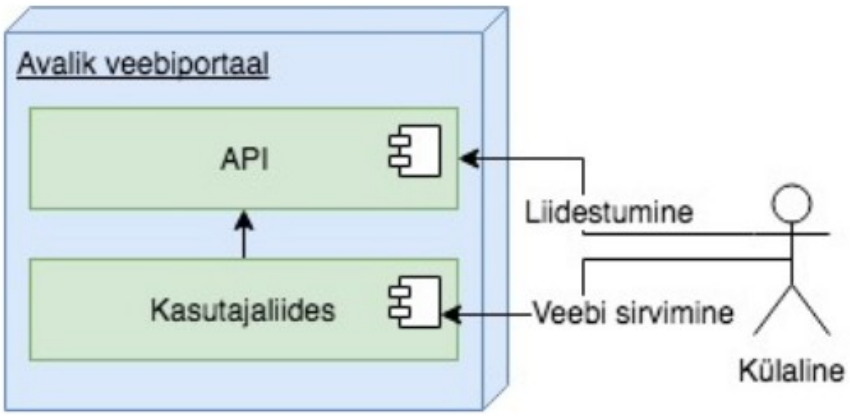
3.2 Haldusrakendus

Haldusrakendus kasutab serveripoolse rakenduse (*backend*) käivitamiseks java Hipster[1] vahendit. Serveripoolne rakendus on Spring Boot rakendus, milles lähtutakse MVC (*Model-View-Controller*) arendusmustrist. Kasutajaliides on realiseeritud Angular 6[2] raamistikuga. Kasutajaliides kasutab serveripoolse rakendusega suhtlemiseks REST-i liidest. Turvalisus on tagatud Spring Security[3] laiendi abil ehk leheküljed, kuhu igaüks ei tohi ligi pääseda, on kaitstud kasutajanime ning parooliga. Nii turvalisuse suurendamiseks kui ka tulevikus tehtavateks arendusteks on järgitud häid programmeerimistavasid. [2]

3.3 Avalik veebiportaal

Missioonijuhtimistarkvara osaks olev avalik veebiportaal tegeleb kasutajatele missiooni kulu visualiseerimisega, visualiseerides selleks erinevaid andmeid satelliidi hetkeoleku kohta. Avalik veebiportaal võimaldab kolmandatel osapooltel mugavalt endaga liidestuda. See osa missioonijuhtimistarkvarast on praeguse töö skoobis. [2]

1 https://www.jhipster.tech/
2 https://angular.io/
3 https://spring.io/projects/spring-security

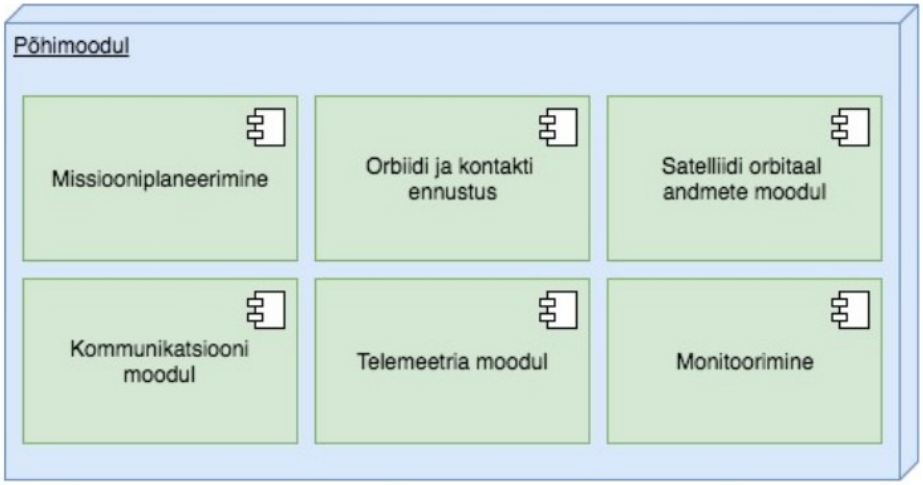


Joonis 2. Avaliku veebiportaali struktuur [2]

Avalik veebiportaal koosneb kahest osast: API-st (*application programming interface*) ning kasutajaliidesest. API ülesandeks on võimaldada kolmandatel osapooltel liidestuda missioonijuhtimistarkvaraga. API abil saab kätte erinevaid avalikke missiooniga seotud andmeid. Lisaks on võimalik kolmandatel osapooltel saata telemeetria andmeid, mis sisalduvad nende poolt kinnipüütud raadiosignaalides. Kui telemeetria andmed jõuavad missioonijuhtimissüsteemi, märgitakse need ära kui usaldusväärsest allikast tulnud andmed. Seda selleks, et võimaliku halbade kavatsustega kasutaja poolt edastatud andmed ei saaks süsteemi tööd rikkuda. [2]

3.4 Põhimoodul

Missioonijuhtimissüsteemis täidab kõige suuremaid ning oluliseimaid ülesandeid põhimoodul. See osa koosneb kuuest alammoodulist, mis suhtlevad omavahel sõnumikomponendi kaudu. Alammoodulid on järgmised:

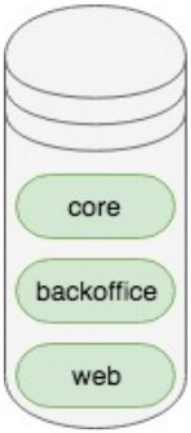


Joonis 3. Põhimooduli alammoodulid [2]

- Missiooniplaneerimise alammoodul
  - Tegeleb Maa seire missioonide planeerimisega, kasutades ülejäänud viite alammoodulit.
- Orbiidi ja kontaki ennustuse alammoodul
  - Tegeleb satelliidi järgmise kontakti täpse aja ning orbiidi väljaarvutamisega.
- Telemeetria alammoodul
  - Kirjeldab telemeetria sõnumite sisu ning dekodeerib need vastavalt iga satelliidi alamsüsteemi spetsifikatsioonile.
- Kommunikatsiooni alammmoodul
  - Tegeleb sõnumite kodeerimisega vastavalt protokollile, kodeeritud paketide maajaamale edastamisega ning satelliidi ja maajaama vahelise andmeside seansi juhtimisega.
- Satelliidi orbitaalandmete alammoodul
  - Tegeleb satelliidi andmete uuendamisega. [2]

3.5 Andmebaas

Missioonijuhtimistarkvaras on kasutusel PostgreSQL[1] andmebaasisüsteem. Paremaks andmebaasimuudatuste haldamiseks kasutatakse Liquibase[2] tarkvara. [2]



Joonis 4. Andmebaasi struktuur [2]

Erinevate moodulite andmed on erinevates andmebaasiskeemides, millest on neid vajadusel võimalik ka täitsa eraldi andmebaasidesse tõsta. [2]

3.6 Sõnumivahetuse komponent

Missioonijuhtimistarkvaras on kasutusel sõnumivahetusel põhinev arhitektuur. See tähendab seda, et süsteemis on keskne sõnumite vahetamiseks mõeldud komponent, mis vahendab sõnumeid erinevate osapoolte vahel. Sõnumivahetuskomponendi tarkvaraks on ActiveMQ, mis võimaldab arhiveerida sõnumeid ja hoida erinevate tehnoloogiate hulka väiksemana. [2]

1 https://www.postgresql.org/
2 https://www.liquibase.org/

4 Loodav süsteem

Selles peatükis on kõik loodava süsteemi kohta käiv informatsioon.

4.1 Üldine informatsioon

Antud töö koosneb kahest osast. Mõlema osa nõuded on välja toodud peatükis 2.

Arvestades asjaolu, et senine haldusrakendus kasutab kasutajaliideses Angular 6 raamistikku ning serveripoolses rakenduses Spring Booti, on ka antud töös neid raamistikke kasutatud.

Esimene osa tööst on pilditöötluse klient, mis on lisatud haldusrakenduse kasutajaliidesesse. Esimene osa kliendist tagab selle, et haldusrakenduses saab mugavalt kindlaks määrata soovitava pildi parameetrid ning need loodava REST teenuse abil andmebaasi salvestada. Teine osa pilditöötluskliendist tagab saatmisvalmis pildi parameetrite informatsiooni näitamise tabelina ning võimaldab tulevikus REST teenuse abil saata soovitud parameetreid kaheksandkoodiks konverteerituna MCS tarkvara sõnumitöötluse põhivoo algusesse.

Teine osa tööst on eraldiseisev REST teenus, mis annab informatsiooni vastavalt päringuga etteantud parameetritele. Loodava REST teenuse eesmärgiks on vastavalt saadud parameetritele leida kasutaja jaoks vajalikud ülelennud. Ülelendude leidmiseks kasutatakse Pythoni[1] PyEphemi[2] teeki. Javal[3] põhineva serveripoolse rakenduse ning Pythonis kirjutatud teegi vahel liigutatakse infot läbi sokkeli (*socket*), mis pannakse käima loodava REST teenuse Spring rakenduse käivitumisel.

Seda süsteemi on missioonijuhtimistarkvarasse vaja, kuna muidu ei saaks lõppkasutajad TTÜ100 tudengisatelliidile piltide tegemiseks käske saata. Teenus tagab ka selle, et

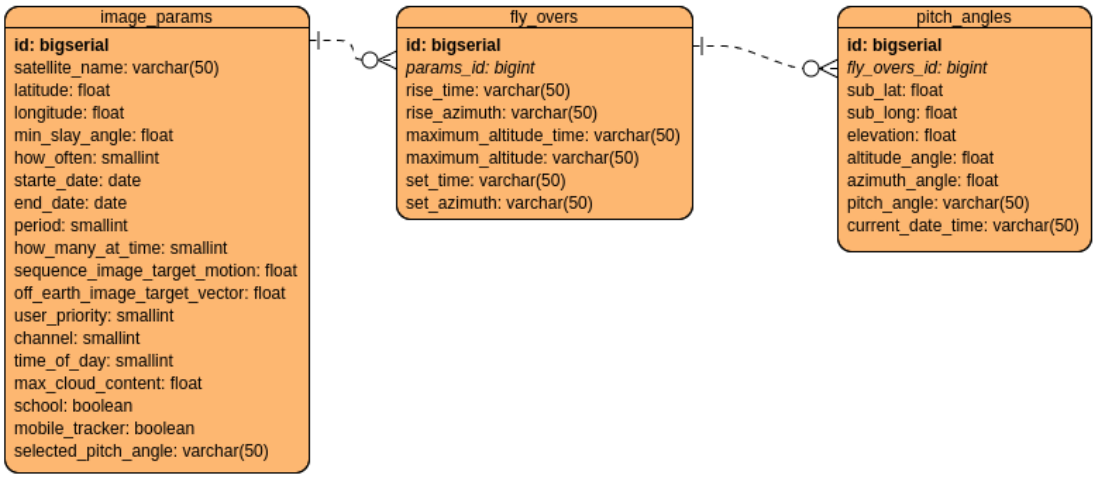
1 https://www.python.org/
2 https://pypi.org/project/pyephem/
3 https://www.java.com/en/

satelliidi meeskonna liikmed saavad näha, millistest kohtadest maakeral satelliit üle lendab ning selle alusel kindlaks määrata soovitud kallutusnurga.

Uute arenduste lisamisel ei muudeta olemasolevat arhitektuuri, vaid täiendatakse seda. Kõik muudatused tuuakse välja antud dokumendis. Uute arenduste lisamisel lähtutakse nii töös [2] välja tootud põhimõttest kasutada avatud lähtekoodiga tarkvara kui ka antud töös peatükis 2 kirjeldatud nõuetest.

4.2 Kasutatavad andmebaasi tabelid

Loodava teenuse jaoks on tarvilikud kolm andmebaasi tabelit: *image_params, fly_overs ning pitch_angles*. *Image_params* tabeli primaarvõti on ühendatud *fly_overs* tabeli võõrvõtmega ning *fly_overs* tabeli primaarvõti on ühendatud *pitch_angles* tabeli võõrvõtmega. Võtmete muutmisel ei tehta midagi, küll aga primaarvõtmega rea kustutamisel kustutatakse sellega seotud võõrvõtmega veerud alamtabelist. Järgnevalt toob töö autor välja eelpool välja toodud tabelite ERD (*Entity Relationship Diagram*) diagrami.



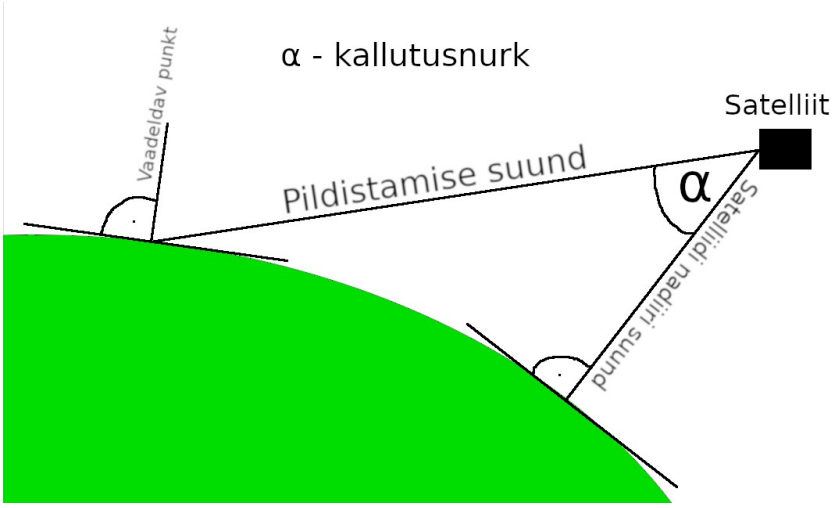
Joonis 5. Kasutatavad tabelid

4.3 PyEphemi tutvustus

PyEphem on Pythoni teek, mis on mõeldud kõrge-täpsusega astronoomia kalkulatsioonide tegemiseks. See põhineb populaarse XEphem-nimelise astronoomia rakendusega samadel alustel. Nimi „ephem” on lühivorm sõnast „ephemeris”, mis tähistab tabelit, kust saab pärida infot erinevate planeetide, asteroidide ning komeetide asukohtade kohta erinevatel kuupäevadel. [3]

4.4 Kallutusnurga arvutamine

Järgnevalt on kirjeldatud kallutusnurga arvutamise protsessi. Kallutusnurga olulisus satellidile saadetavate parameetrite hulgas seisneb selles, et selle alusel saab määrata, mis nurga alt satelliit soovitud objekti pildistab. Kallutusnurga arvutamiseks kasutatakse vaadeldava punkti koordinaate, satelliidi koordinaate planeedil Maa ja satellidi kõrgust merepinnast mõõdetuna meetrites. Esmalt teisendatakse kõik need parameetrid [4] meetodi abil Descartesi koordinaatideks. Seejärel leitakse kaks vektorit (satelliidilt tema Maa koordinaatideni ning satelliidilt vaadeldava objektini) ning nendevaheline nurk, milleks ongi soovitud kallutusnurk. Kallutusnurka illustreeriv joonis on järgnev.



Joonis 6. Kallutusnurga joonis

4.5 Kahe punkti vahelise kauguse arvutamine

Töö käigus oli tarvis arvutada maakera kahe punkti vahelist kaugust. Selleks uuris autor nii Euclidean kauguse kui ka Haversine kauguse valemit.

Euclidean kauguse valemit kasutatakse koolides kahe punkti vahelise kauguse arvutamiseks. Kahjuks ei võta antud valem arvesse Maa raadiust, seetõttu ei saanud seda kasutada. Valem on selline: $d(A,B)=\sqrt{(x_b-x_a)^2+(y_b-y_a)^2}$. [5]

Haversine kauguse valemit kasutatakse planeedil Maa olevate punktide vahemaa arvutamiseks. Selle suureks plussiks on asjaolu, et võetakse arvesse ka Maa raadiust. Valem on selline: $d=2rsin^{-1}(\sqrt{\sin^2(\frac{\phi_2-\phi_1}{2})+\cos(\phi_1)\cos(\phi_2)\sin^2(\frac{\psi_2-\psi_1}{2})})$ [5]

Kus d tähistab vahemaad kahe punkti vahel, r maakera raadiust, ψ on pikkuskraad, φ on laiuskraad. Kuna selles valemis sisaldub ka maa raadiuse parameeter, otsustas töö autor kasutada seda valemit. [5]

4.6 Andmevood ülelende leidvas pythoni failis

Serveripoolse rakenduse käivitumisel pöördutakse ülelende leidva pythoni faili poole. Selles failis käivitatakse sokkel, mis jääb kuulama, kas keegi talle andmeid saadab. Andmete kättesaamise järel uuendatakse satelliitide kohta käivat informatsiooni. Selleks tehakse GET päring veebilehele https://www.tle.info/data/TLE.ZIP. Sealt allalaetud .zip fail pakitakse lahti. Saadud TLE.txt failist loetakse satellidi nime alusel välja vastava satellidi informatsioon. Seejärel luuakse ephemi vaatlusobjekt, mille alusel hakatakse ülelende leidma. Järgneb ülelendude andmete pärimine, kus tehakse esmalt kindlaks, kellelt päring tuli. Koolidelt tulnud päringu puhul leitakse järgmise 14 ööpäeva ülelennud. Android seadmete puhul leitakse ainult järgmise ülelennu andmed. Pilditöötluse kliendi päringule vastatakse kas algus- ja lõpukuupäeva vahele jäävate ülelendudega või kui lõpukuupäeva pole kindlaks määratud, leitakse järgmise ööpäeva ülelennud. Pärast iga ülelennu informatsiooni leidmist vaadatakse, kas saab vastavalt ajalistele parameetritele veel ülelende leida. Kui see on võimalik, siis muudetakse ka vaatlusobjekti aega etteantud intervalli võrra. Vaikimisi intervalliks on üks tund.

Vastavalt püstitatud nõuetele küsitakse pyephemi teegilt ühe ülelennu kohta horisondile tõusmise aega, horisondile tõusmise ajal olevat asimuuti, maksimaalse kõrgusnurga aega, maksimaalset kõrgusnurka, horisondi taha minemise aega, horisondi taha minemise ajal olevat asimuuti ning päikese loojumise ja päikese tõusmise aegu. Lisaks küsitakse iga 30 sekundi tagant andmeid satellidi kõrguse kohta tema hetke kõrgust merepinnast mõõdetuna meetrites, satellidi koordinaate Maa peal, kõrgusnurka, asimuuti ning praegust aega. PyEphemiga andmete pärimiste koodinäited on lisades 5 kuni 7.

4.7 Andmevood loodavas teenuses

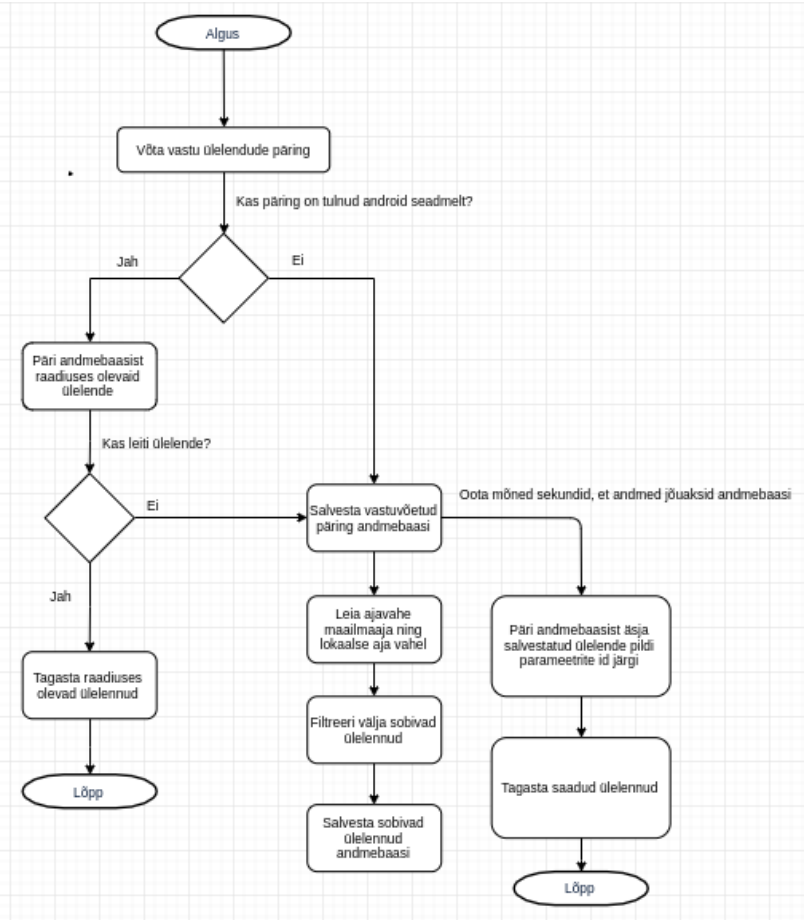
Loodav REST teenus hakkab tööle, kui lõppkasutaja teeb sinna päringu. Lõppkasutajateks võib olla nii kolmandad osapooled ehk koolid, android rakendused kui ka loodav pilditöötlusklient. Pärast päringuandmete kohalejõudmist loetakse neid.

Android seadmelt või loodavalt pilditöötluskliendilt tulnud päringu puhul leitakse ülelennud järgnevalt: kontrollerisse saadetud piltide parameetrid salvestatakse *core* andmebaasis olevasse tabelisse *image_params*. Vastuseks saadakse äsja salvestatud pildi parameetrite id. Järgnevalt saadetakse pildi parameetrid JSON (*javascript object notation*) objektina ActiveMQ järjekorda. Seda järjekorda kuulab ka otseselt Pythoni failiga suhtlev teenusfail, mis võtab järjekorrast andmed ning saadab need sokkeli kaudu Pythoni faili. Pythoni failist tagasi tulnud JSON kujul ülelennud võetakse teenusfailis vastu. Arvestades asjaolu, et pyephem annab ajad maailmaaja järgi, leitakse kohaliku aja ning maailmaaja tundide vahe. Järgnevalt filtreeritakse saadud ülelendudest vastavalt saadud sisendparameetritele välja kas ainult päevased, ainult öised või mõlema aja ülelennud. Päevasteks ülelendudeks peetakse ülelende, mis jäävad PyEphemilt saadud päikesetõusu ja päikeseloojangu vahele ning ühe kuupäeva sisse. Kõiki ülejäänud ülelende peetakse öisteks. Sobivad ülelennud pannakse jälle ActiveMQ järjekorda, kust nad kontrolleris välja võetakse ning *fly_overs* tabelisse salvestatakse. Iga ülelennu juures olevad kindla intervalli tagant tehtud päringud salvestatakse *pitch_angles* tabelisse. Lisaks salvestatakse *pitch_angles* tabelisse välja arvutatud kallutusnurk, mille kohta on täpsem info peatükis 4.4**.** Kontrolleris oodatakse natuke aega, enne kui tehakse päring salvestatud pildiparameetrite id-le vastavate ülelendude

andmete küsimiseks andmebaasi tabelitest *fly-overs ja pitch_angles*. Seda selleks, et kõik andmed jõutaks andmebaasi salvestada.

Mõnelt koolilt tulnud päringu puhul on ülelendude leidmise protsess järgmine: esmalt tehakse *image_params* tabeli pihta päring, mis otsib päringuga saadetud koordinaatidest 50 km raadiusesse jäävaid teisi pildi parameetreid. Selleks on andmebaasis funktsioon [6], mis kasutab Haversine kauguse valemit [5]. Sellest valemist on lähemalt kirjutatud peatükis 4.5**.** Juhul, kui leitakse raadiuses olevaid pildi parameetreid, millele vastavad ka ülelennud, tagastatakse need ülelennud ning päringus olevaid andmeid andmebaasi ei salvestata. Vastasel juhul, kui raadiuse sisse ei jää ühtegi ülelendu, leitakse ülelennud nii, nagu tehakse seda android seadmete ning loodava pilditöötluskliendi jaoks.

Järgnevalt on välja toodud ka andmevoo tegevusdiagramm.



Joonis 7. Andmevood loodavas süsteemis

4.8 Java ja pythoni vahelise ühenduse alternatiiv

Antud töö tegemisel oli autoril probleem, mille abil panna omavahel suhtlema Javal põhinev serveripoolne rakendus ning ülelende küsiv Pythoni fail. Võimalusi oli kaks.

Esimeseks võimaluseks oli Java platvormil töötav ning Java klassides kasutatav Jython[1], mis on Pythoni keele implementatsioon. Jython võimaldab mugavalt Javast Pythoni failidele parameetreid ette anda, saadud tulemust tagasi saada ning kuvada. [7]

Teiseks võimaluseks oli luua Java kliendi ja Pythoni serveri vaheline sokkel. Pythoni server käivitub, kui serveripoolne rakendus käivitatakse ning kuulab, kas serveripoolsest rakendusest mingeid andmeid tuleb. Pärast ülelendude andmete välja filtreerimist saadetakse need tagasi serverirakendusele.

Arvestades asjaolu, et pyEphem jookseb C[2] teegi peal, mida Jython ei toeta, otsustas töö autor sokkeli lahenduse kasuks. C teegi toetuse olemasolu korral oleks töö autor kindlasti Jythonit kasutanud, kuna siis oleks ühendust lihtsam arendada olnud.

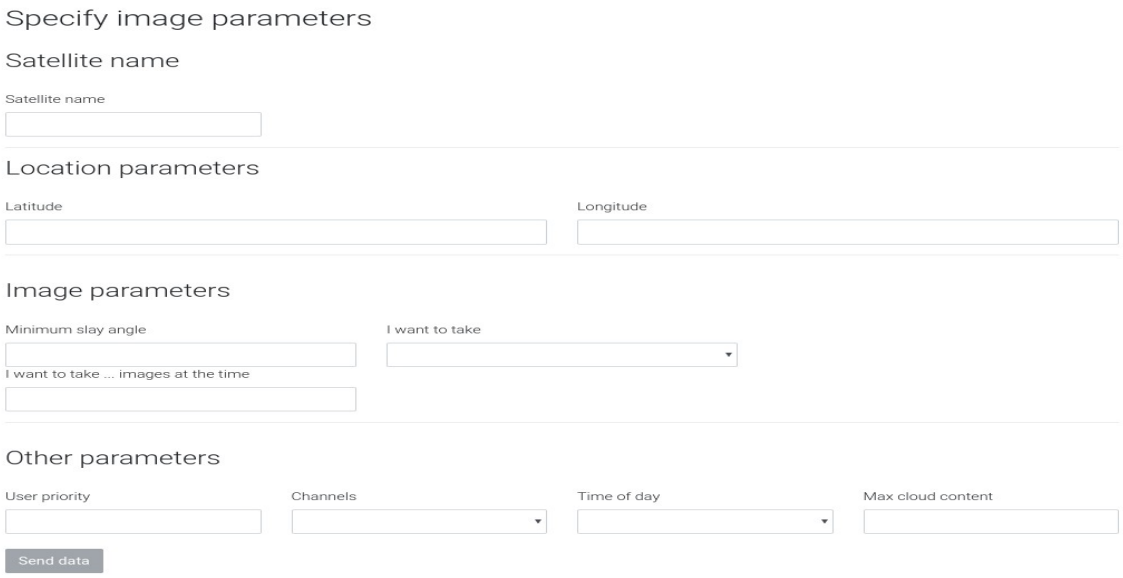
4.9 Loodav piltitöötlusklient

4.9.1 Esimene osa

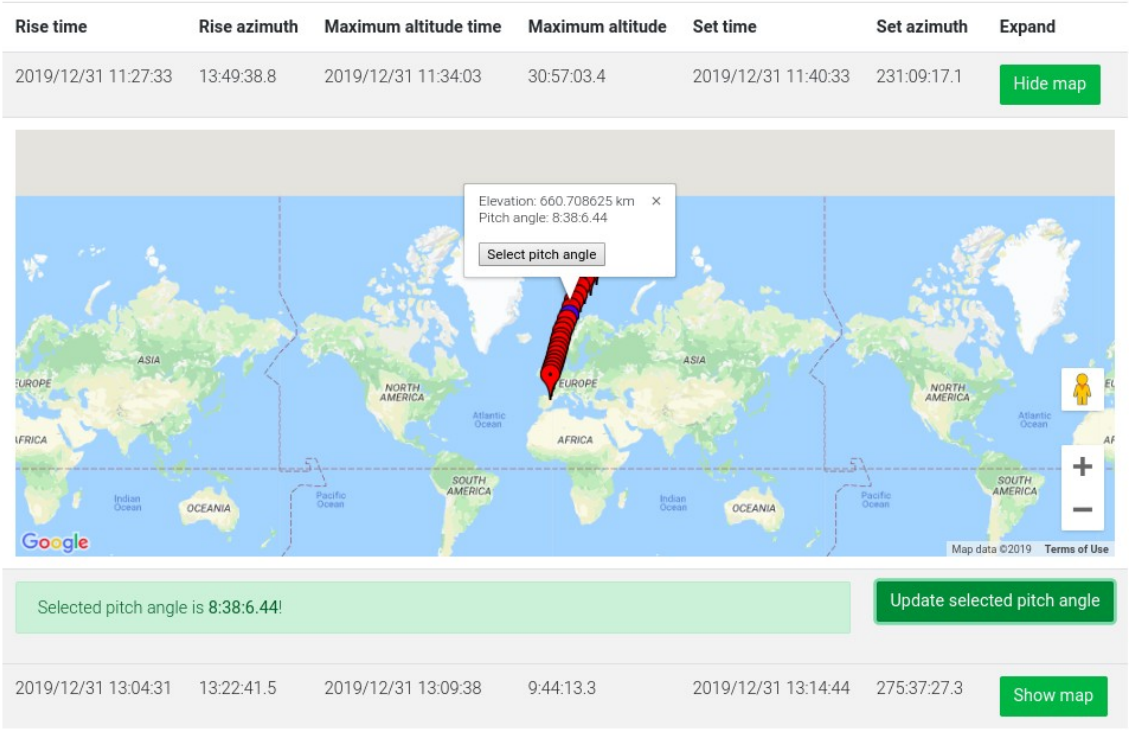
Esimene osa kliendist on HTML vorm, mis küsib kasutajalt peatükis 2 esitatud parameetreid ning valideerib neid vastavalt esitatud nõuetele. Kui kõik on korras, saadetakse sisestatud andmed REST päringuga loodavale teenusele. Mõne sekundi möödudes tuleb teenuselt vastus ning kui see ei sisalda andmeid leitud ülelendude kohta, kuvatakse vastav teade. Vastasel juhul, kui vastus sisaldab ülelendude andmeid, kuvatakse need vormi all olevas tabelis. Tabeli iga rea viimane veerg sisaldab nuppu, millele vajutamisel kuvatakse valitud rea all kaart. Kaardi loomiseks on kasutatud Angulari agm/core[3] komponenti. Kaardil kuvatakse iga 30 sekundi tagant päritud informatsioon markeritena. Ühel markeril kuvatakse satelliidi kõrgus maapinnast kilomeetrites, kallutusnurk ning nupp, millele vajutades määratakse Typescriptis[4] valitud kallutusnurk ära ning kuvatakse vastav teade. Teate kõrvale lisatakse nupp,

1 https://www.jython.org/
2 https://www.geeksforgeeks.org/c-language-set-1-introduction/
3 https://angular-maps.com/api-docs/agm-core/index.html
4 https://www.typescriptlang.org/

millele vajutamisel salvestatakse valitud kallutusnurk loodava teenuse abil andmebaasi vastava pildi parameetrite objekti juurde. Järgnevalt toob töö autor välja pildid vormist ning kuvatavast tabelist, kus on äsja valitud üks kindel kallutusnurk.



Joonis 8. Kuvatõmmis loodavast HTML vormist

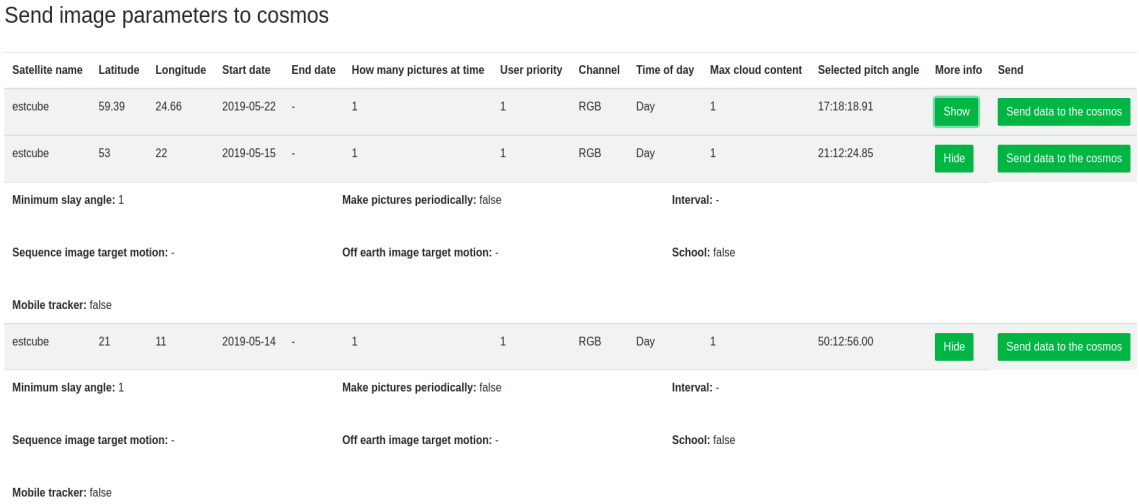


Joonis 9. Kuvatõmmis kuvatavast tabelist

4.9.2 Teine osa

Pilditöötluskliendi teiseks osaks on tabel, kust saab näha informatsiooni valmis olevate pildi parameetrite objektide kohta. Objekt on valmis, kui sellele on valitud sobiv kallutusnurk. Tabeli iga rea viimasel veerul on nupp, millele vajutamisel saadetakse valitud pildi parameetrid tulevikus serveripoolsesse rakendusse, kust neist tulevikus genereeritakse sobiv kaheksandkood, mis omakorda saadetakse MCS tarkvara sõnumitöötluse põhivoo algusesse.

Eelpool nimetatud põhivoog edastab andmeid MCS tarkvara ning satelliidi vahel. Kättesaadud kaheksandkoodi alusel teeb satelliit pildid ning saadab need tagasi maale Bayeri maatriksina. Tulevikus on plaanis integreerida juba olemas olev Bayeri maatriksit pildiks konverteeriv lahendus pilditöötluskliendi teise osa külge. Seejärel saab iga pildi parameetrite rea alla tekitada tehtud piltide nimekirja. Järgnevalt toob töö autor välja pildi kirjeldatud tabelist.



Joonis 10. Valmis olevate pildi parameetrite tabel

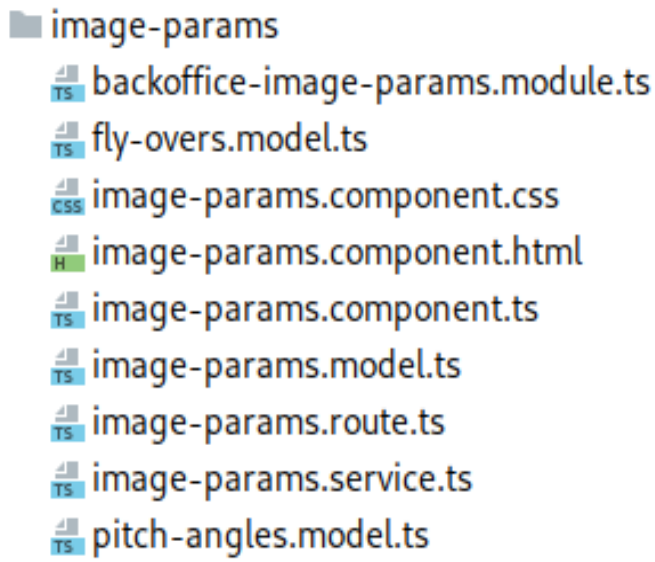
5 Loodava süsteemi struktuur

Selles peatükis tuuakse välja nii loodud teenuse kui ka pilditöötluskliendi struktuurid.

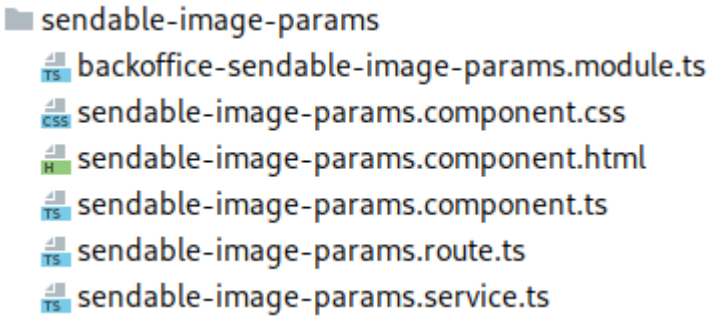
5.1 Loodava pilditöötluskliendi struktuur

Loodav pilditöötlusklient koosneb kahest Angulari komponendist: *image-params* ning *sendable-image-params*. Mõlemad komponendid on seotud missioonijuhtimis-süsteemiga läbi vastavate moodulite importimise *commanding* moodulisse.

Nii *image-params* komponent kui ka *sendable-image-parms* koosneb järgmistest failidest: mooduli fail, mis määrab kindlaks selles moodulis oleva komponendi ning teenuse, komponendis kasutatav css fail, komponendis kasutatav html fail, komponendi fail ning teekondade (*paths*) kindlaks määramise fail ja teenuse fail, mis suhtleb serveripoolse rakendusega. Lisaks on *image-params* komponendil ka *fly-overs* objekti andmefail, *image-params* objekti andmefail ja *pitch-angles* objekti andmefail.



Joonis 11. *Image-params* komponendi struktuur

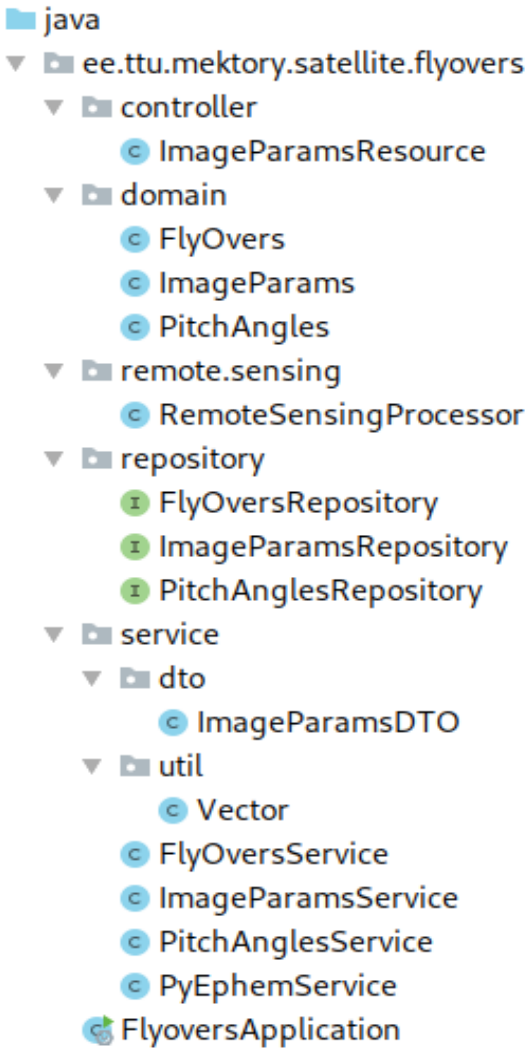


Joonis 12. *Sendable-image-params* komponendi struktuur

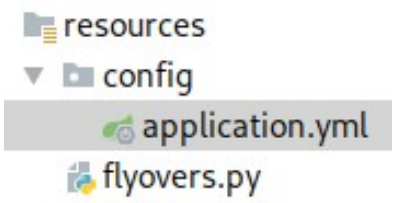
5.2 Loodava teenuse struktuur

Loodav *REST* teenus jaotub järgmisteks osadeks:

- *Controller* sisaldab faili, kus on kirjeldatud serveripoolse rakenduse lõpp-punktid (*endpoints*), kuhu pilditöötlusklient saab teenuste kaudu andmeid saata.
- *Domain* sisaldab andmefaile, kus on kirjeldatud andmebaasi tabelite veerud.
- *Remote.sensing* sisaldab faili, mille abil saab tulevikus satelliidilt tulnud Bayeri maatriksi pildiks konverteerida. See funktsionaalsus realiseeritakse tulevikus.
- *Repository* sisaldab otseselt andmebaasiga suhtlevaid faile.
- *Service* sisaldab *repository* failidest tulnud andmeid töötlevaid, töötluseks vajalikke ning pythoni-java-vahelise suhtlusega tegelevaid faile.
- *Config* sisaldab teenuse konfiguratsiooni hoidvat faili.
- Lisaks on ka Pythoni fail, kus päritakse satelliidi andmeid kasutades pyephemi teeki.



Joonis 13. Loodava teenuse struktuur



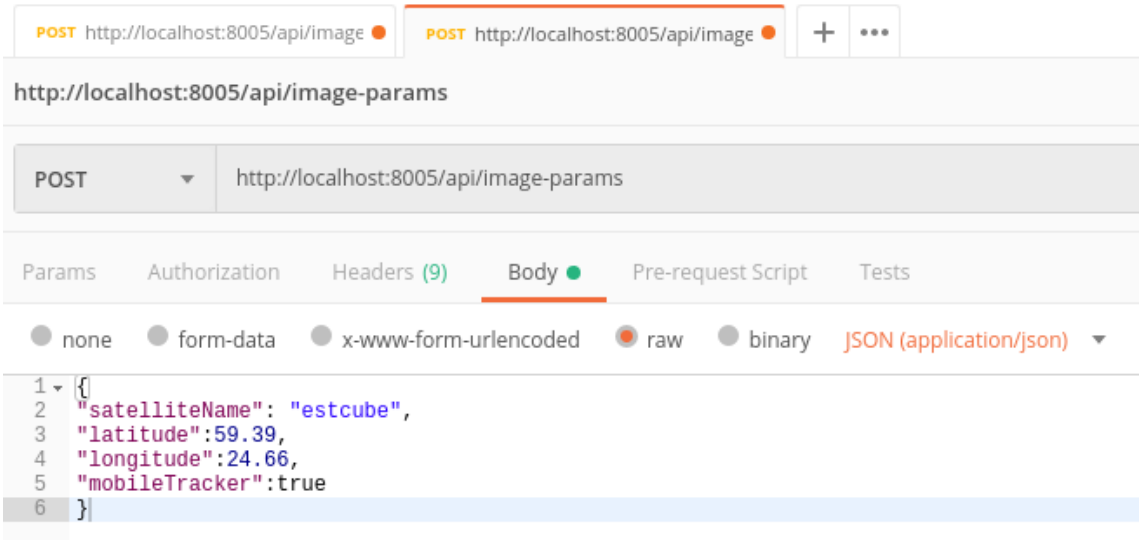
Joonis 14. Loodava teenuse seadete struktuur

6 Testimine

Arvestades asjaolu, et TTÜ100 tudengisatelliit ei ole veel kosmosesse tõusnud, tehakse kõik testimised kasutades Estcube 1 satellliiti.

6.1 Manuaalne testimine

Loodud teenust on manuaalselt testitud. Selleks on kasutatud nii API-de arendamist lihtsustavat programmi Postman[1] kui ka loodud pilditöötlusklienti. Mõlemat moodi testimisel on jälgitud, kas vastuseks saadav informatsioon on kooskõlas edastatud päringus kindlaks määratud parameetritega. Järgnevalt toob töö autor välja kaks pilti, mis illustreerivad Postmani programmis tehtavat päringut ning saadavat vastust.



Joonis 15. Näide postmanis tehtavast päringust

1 https://www.getpostman.com/

Joonis 16. Näite postmanis tehtud päringu tulemusest

6.2 Automaattestimine

Koodis on järgmiseid tegevusi teevad meetodid kaetud ühiktestidega:

- Loodud pilditöötlusteenuse vormist tulnud kuupäevade teisendamine õigesse formaati.
- Kallutusnurga teisendamine ujukomaarvu kujult kraad:minut:sekund kujule.
- Vektorite pikkuste arvutamine ning kahe vektori vahelise nurga arvutamine.

Kuupäevade teisendamine õigele kujule on tähtis, sest PyEphemi teek aktsepteerib kuupäevi ainult kujul aasta/kuu/päev tund:minut:sekund. Valesti arvutatud kallutusnurga puhul poleks võimalik soovitud objekte kosmosest õige nurga alt pildistada. Vektoritega seotud tehete kontrollimine on oluline, et tagada tõesed algandmed kallutusnurga kalkuleerimiseks. Testimiseks on kasutatud Java jaoks mõeldud testimisraamistikku Junit[1].

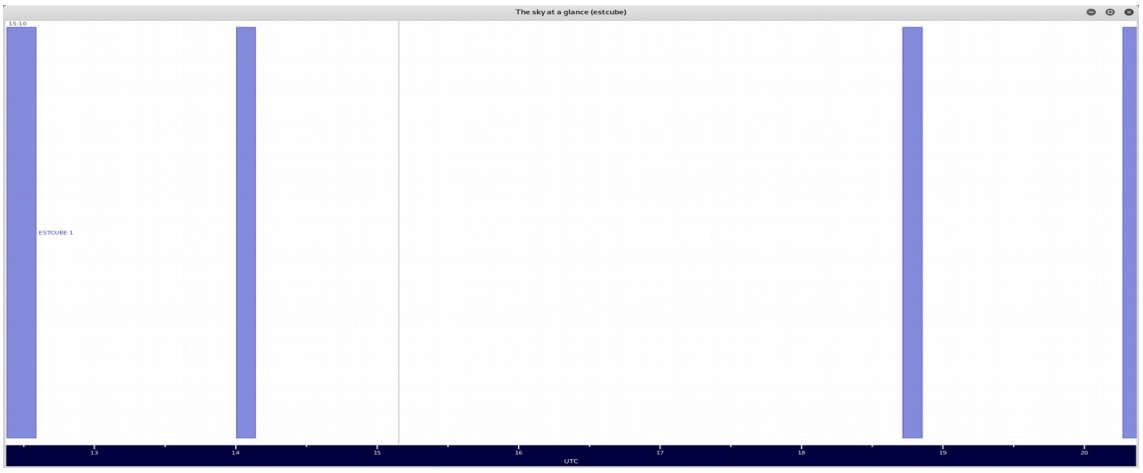
1 https://junit.org

Antud koodijupi testimisel otseselt mittevajalike väliste sõltuvuste eemaldamiseks on kasutatud junitiga ühilduvat raamistikku nimega Mockito[1].

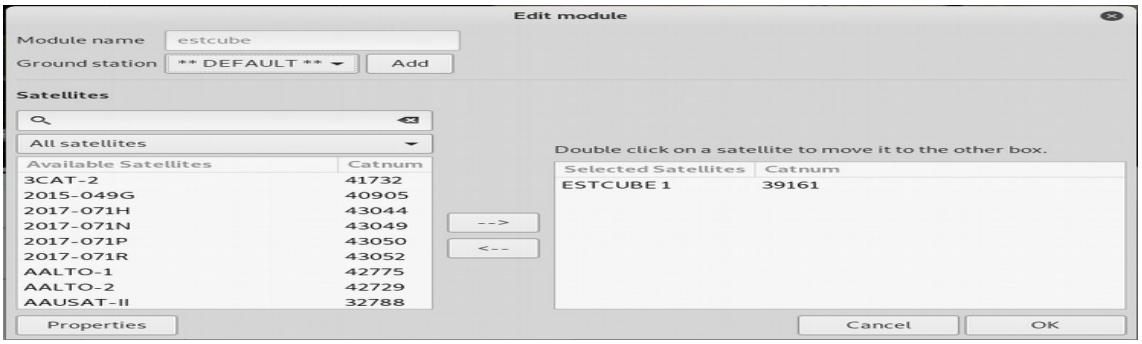
6.3 Saadud andmete verifitseerimine

Loodud teenuselt saadud andmete õigsuse kontrollimiseks on kasutatud nii Estcube-1 satelliidi kohta internetist leitud informatsiooni otsimist kui ka Gpredict[2]-nimelist programmi.

Gpredict võimaldab soovitud satelliitide kohta käivat informatsiooni reaalajas jälgida. Muuhulgas kuvab Gpredict satelliidi kohta näiteks asimuudi nurga kraadides, kõrguse merepinnast, asukoha ja kiiruse soovitud kuupäeval ning kellaajal. Järgnevalt toob töö autor välja paar pilti Gpredict programmist. [8]



Joonis 17. Kuvatõmmis satelliidi ülelendude aegadest



Joonis 18. Kuvatõmmis satelliidi valimise aknast

1 https://site.mockito.org
2 http://gpredict.oz9aec.net

7 Kokkuvõte

Lõputöö eesmärgiks oli luua missioonijuhtimistarkvarale lisafunktsionaalsus, mille abil oleks võimalik sätestada piltide tegemiseks vajalikke parameetreid ja nende abil leida satelliidi ülelende. Selle saavutamiseks analüüsiti ning võeti arvesse praeguste kasutajate soove. Analüüsi põhjal pandi kirja nõuded, mis said ka täidetud.

Töö raames loodi REST teenus, millelt saab pärida satelliidi kohta käivaid andmeid. Teenus loodi Spring Booti raamistikule ning suhtleb Pythoni failis oleva pyEphemi teegiga läbi sokkeli. Lisaks sai loodud Angular 6 raamistikul põhinev pilditöötlusklient, mille abil saab sätestada piltide tegemiseks vajalikke parameetreid ning neid kasutajale kuvada.

Loodud lisafunktsionaalsust on plaanis veel edasi täiustada, integreerides seda ühe teise lõputööga, et sätestatud parameetrid edukalt satelliidini ning tehtud pildid satelliidilt loodud pilditöötluskliendini jõuaksid.

Töö autor sai lõputööd kirjutades ennast kurssi viia erinevate astronoomias ja lennunduses kasutavate terminitega. Lisaks sai autor end proovile panna ka java-pythoni-vahelise ühenduse loomisel ja trigonomeetrias.

8 Kasutatud kirjandus

[1] „TTÜ100 Satelliidi tutvustus,“ [Võrgumaterjal]. Available: https://www.ttu.ee/projektid/mektory-est/satelliidiprogramm-4/satelliidiprogramm/. [Kasutatud 17 Mai 2019].

[2] S. Romanov, *Kuupsateliidi missioonijuhtimistarkvara arhitektuur*, Tallinn: Tallinna Tehnikaülikool, 2017.

[3] „pyephem . PyPI,” [Võrgumaterjal]. Available: https://pypi.org/project/pyephem/. [Kasutatud 16 Mai 2019].

[4] „*Converts WGS84 coordinates to ECEF cartesian coordinates*,” [Võrgumaterjal]. Available: https://gist.github.com/soggier/7406b230d486df466f6f. [Kasutatud 17 Mai 2019].

[5] V. Krõšin, *Asukohapõhiste veebisündmuste algoritmid (Twitteri andmete näitel)*, Tartu: Tartu Ülikool, 2014.

[6] „*PostgreSQL function for haversine distance calculation, in miles*,” [Võrgumaterjal]. Available: https://gist.github.com/carlzulauf/1724506. [Kasutatud 15 Mai 2019].

[7] „*What is Jython?*,” [Võrgumaterjal]. Available: https://www.jython.org/archive/21/docs/whatis.html. [Kasutatud 14 Mai 2019].

[8] „*Gpredict: Free, Real-Time Satellite Tracking and Orbit Prediction Software*,” [Võrgumaterjal]. Available: http://gpredict.oz9aec.net/. [Kasutatud 17 Mai 2019].

Lisa 1 – Teenuse poolt tagastatav informatsioon


```
[
    {
        "id": 210,
        "paramsId": 25,
        "riseTime": "2019/4/26 07:41:22",
        "riseAzimuth": "32:13:55.4",
        "maximumAltitudeTime": "2019/4/26 07:45:42",
        "maximumAltitude": "5:10:22.0",
        "setTime": "2019/4/26 07:49:59",
        "setAzimuth": "111:48:56.7",
        "pitchAngles": [
            {
                "id": 1589,
                "flyOversId": 83,
                "subLat": 75.4296107405999976,
                "subLong": 81.9248884147999945,
                "elevation": 659705.8125,
                "altitudeAngle": -0.0129076793204999998,
                "azimuthAngle": 29.2152940614999999,
                "currentDateTime": "2019/4/30 07:55:27",
                "pitchAngle": "23:59:35.64"
            },
            ....
        ]
    },
....
]
```

Lisa 2 – Koolide poolt tehtav päring teenuse pihta

```
{
	"latitude":59.39,
	"longitude":24.66,
	"timeOfDay": 1,
	"school":true
}
```

Lisa 3 – Android seadmete poolt tehtav päring teenuse pihta

```
{
	"satelliteName":”Estcube”,
	"latitude":59.39,
	"longitude":24.66,
	"mobileTracker":true
}
```

Lisa 4 – Pilditöötluskliendi poolt tehtav päring teenuse pihta

```
{
	"satelliteName"="Estcube",
	"latitude"=59.39,
	"longitude"=24.66,
	"minSlayAngle"=1,
	"howOften"=1,
	"startDate"=”Tue Apr 30 03:00:00 EEST 2019”,
	"howManyAtTime"=1,
	"userPriority"=1,
	"channel"=1,
	"timeOfDay"=1,
	"maxCloudContent"=1,
	"school"=false
}
```

Lisa 5 – PyEphemi vaatlusobjekti loomine

```
while 1:
    connection, address = s.accept()
    message = connection.recv(1024)

    d = json.loads(message)

    result["params_id"] = d["params_id"]
    result["fly_overs"] = []
    updatedata(tlezip, "https://www.tle.info/data/TLE.ZIP")

    tlelist = {}
    list = open(tlefile, "r").readlines()
    lines = []
    for i in range(len(list)):
        if (str(d["satellite_name"].upper()) in list[i]):
            lines.append(list[i])
            lines.append(list[i + 1])
            lines.append(list[i + 2])
            break

    mectory = ephem.Observer()
    mectory.lon, mectory.lat = str(d["longitude"]), str(d["latitude"])
    mectory.date = ephem.Date(str(d["start_date"]))
    nextPass(mectory, lines)
    connection.send(json.dumps(result))
    connection.close()
```

Lisa 6 – Sissetulnud päringu analüüs

```
def nextPass(site, tle):
    satellite = ephem.readtle(tle[0], tle[1], tle[2])

    if not d["school"]:
        if (str(d["end_date"]) == ""):
            end_date = ephem.Date(site.date + day)
            if not d["mobile_tracker"]:
                while site.date < end_date:
                    getData(site, satellite, hour)
            else:
                getData(site, satellite, hour)

        else:
            end_date = ephem.Date(str(d["end_date"])) + day
            while site.date < end_date:
                interval = hour
                if (str(d["period"]) == "1"):
                    interval = hour
                elif (str(d["period"]) == "2"):
                    interval = day
                elif (str(d["period"]) == "3"):
                    interval = week
                elif (str(d["period"]) == "4"):
                    interval = month
                elif (str(d["period"]) == "5"):
                    interval = year
                getData(site, satellite, interval)

    else:
        end_date = site.date + 15 * day
        while site.date < end_date:
            getData(site, satellite, hour)
```

Lisa 7 - Satelliidi andmete pärimine

```
def getData(site, satellite, interval):
    rise_time, rise_time_azimuth_angle, max_altitude_angle_time, max_altitude_angle, set_time, set_time_azimuth_angle = site.next_pass(satellite)

    pitch_angles = []
    site.date = ephem.Date(str(rise_time))
    while site.date < ephem.Date(str(set_time)):
        satellite.compute(site)
        pitch_angles.append({"elevation": str(satellite.elevation), "sublong": str(satellite.sublong / ephem.degree),
                             "sublat": str(satellite.sublat / ephem.degree), "alt":str(satellite.alt / ephem.degree),
                             "az":str(satellite.az / ephem.degree),
                             "current_date_time": str(site.date)})
        site.date = site.date + 30 * second

    site.date = ephem.Date(str(max_altitude_angle_time))
    satellite.compute(site)

    sunrise=site.previous_rising(ephem.Sun())
    sunset=site.next_setting(ephem.Sun())
    result["fly_overs"].append({"rise_time":str(rise_time),
                                "rise_time_azimuth_angle":str(rise_time_azimuth_angle),
                                "max_altitude_angle_time":str(max_altitude_angle_time),
                                "max_altitude_angle":str(max_altitude_angle),
                                "set_time": str(set_time),
                                "set_time_azimuth_angle": str(set_time_azimuth_angle),
                                "sunrise": str(sunrise),
                                "sunset": str(sunset),
                                "pitch_angles": pitch_angles})
    site.date = site.date + interval
```